MOTOROLA

Wireless Communications Division
Wireless Subscriber Sytem Group

WMSG/WSD Neptune LTE Baseband IC Specification
Version 1.5

June 26, 2003

WMSG/WSD Neptune LTE IC Baseband Specification

Motorola Triton Low Cost Architechture Program

Preliminary Specification

Austin Wireless Design Center
6501 W. William Cannon Drive
Austin, TX 78727

Control No: **Issued To:** **Issued By:**

Organization and Non-Disclosure No:

Contents

Chapter 1
Neptune LTE IC Summary (ICSum)

1.1 Overview . . . 1-2
1.2 Neptune LTE IC Features . . . 1-2
1.3 Neptune LTE IC Module List . . . 1-3
1.4 Neptune LTE Functional Summary . . . 1-4
1.4.1 DSP . . . 1-4
1.4.2 DSP Memory . . . 1-4
1.4.2.1 DSP BIST (DBIST) . . . 1-5
1.4.3 DSP Program Paging . . . 1-5
1.4.4 DSP Peripherals . . . 1-5
1.4.4.1 Baseband Port Module (BBP) . . . 1-5
1.4.4.2 Layer1 Encryption Module (LEM) . . . 1-5
1.4.4.3 DSP Timer Module (DTIMER) . . . 1-5
1.4.4.4 Serial Audio Codec Port (SAP) . . . 1-5
1.4.4.5 Direct Memory Access Controller (DMA) . . . 1-6
1.4.4.6 DMA Arbitration Module (DMA_ARB) . . . 1-6
1.4.4.7 Data RAM Arbiter (DRA) . . . 1-6
1.4.4.8 Viterbi Accelerator (VIAC) . . . 1-6
1.4.4.9 DSP Special Interrupt Handler (DSIH) . . . 1-6
1.4.4.10 DSP On Chip Emulator Debug Module (OnCE) . . . 1-6
1.4.4.11 DSP Debug Module (PDDM) . . . 1-6
1.4.4.12 Voice Codec (VOCOD) . . . 1-7
1.4.4.13 DSP Clock Generator (DCLKG) . . . 1-7
1.4.5 Mixed Signal Modules . . . 1-7
1.4.5.1 Receive Coprocessor (RxCPROC) . . . 1-7
1.4.5.2 Receiver Analog Front End (RxAFE) . . . 1-7
1.4.5.3 Fast Digital DC Adapt (DCADAPT) . . . 1-7
1.4.5.4 Receiver Saturation Detection and SDM Dither Generation (RxSDG) . . . 1-7
1.4.5.5 Transmitter (TX) . . . 1-8
1.4.5.6 Rx/Tx Synthesizer (TRSYNT) . . . 1-8
1.4.5.7 Power Amplifier Control (PAC) . . . 1-8
1.4.5.8 Regulators (REGUL) . . . 1-8
1.4.5.9 Tuning Circuit (TUNEC) . . . 1-8
1.4.5.10 Reference PLL (REFPLL) . . . 1-8
1.4.5.11 General Purpose ADC (GPADC) . . . 1-8
1.4.5.12 Crystal Oscillator (CROSC) . . . 1-9
1.4.5.13 Analog Test Interface (ANATEST) . . . 1-9
1.4.6 Special Cells . . . 1-9
1.4.6.1 IC Identification Module (IIM) . . . 1-9

1.4.6.2 Clock Monitor (CMON) . . . 1-9
1.4.6.3 Power On Reset (POR) . . . 1-9
1.4.6.4 System JTAG Controller (SJC) . . . 1-10
1.4.7 ARM7 MCU . . . 1-10
1.4.8 MCU Memory . . . 1-10
1.4.9 MCU Peripherals . . . 1-10
1.4.9.1 ARM7 Platform Core Arbiter (CARB) . . . 1-10
1.4.9.2 ARM External Interface Module (AEIM) . . . 1-11
1.4.9.3 ARM IPBus Interface (AIPI) . . . 1-11
1.4.9.4 Alternate Master Arbiter (AMARB) . . . 1-11
1.4.9.5 Multiple Queue Serial Peripheral Interface (MQSPI) . . . 1-11
1.4.9.6 ARM7TDMI-S Interrupt Controller (AITC) . . . 1-11
1.4.9.7 SIM Interface Modifications (SIM) . . . 1-11
1.4.9.8 Universal Asynchronous Receiver Transmitter (UART) . . . 1-12
1.4.9.9 Deep Sleep Module (DSM) . . . 1-12
1.4.9.10 Watchdog Timer (WDOG) . . . 1-12
1.4.9.11 General Packet Radio Service Encryption Module (GEM) . . . 1-12
1.4.9.12 Keypad Port (KPP) . . . 1-12
1.4.9.13 Enhanced General Purpose Timer (EGPT) . . . 1-12
1.4.9.14 Test Control Module (TCM) . . . 1-13
1.4.9.15 Scan Test Controller (STC) . . . 1-13
1.4.9.16 MCU Watchpoint (AWPT) . . . 1-13
1.4.9.17 USB Digital Phase Locked Loop (REFPLL) . . . 1-13
1.4.9.18 Real Time Reference (RTR) . . . 1-13
1.4.9.19 Real Time Clock (RTC) . . . 1-14
1.4.9.20 Display Memory Access Controller (DMAC) . . . 1-14
1.4.9.21 Clock Control Module (CCM) . . . 1-14
1.4.9.22 External Interrupt Module (INT) . . . 1-14
1.4.9.23 Analog to Digital Interface (A2DIGL) . . . 1-14
1.4.10 Shared Peripherals . . . 1-14
1.4.10.1 Universal Serial Bus Module (USB) . . . 1-15
1.4.10.2 General Purpose I/O (GPIO) . . . 1-15
1.4.10.3 MCU / DSP Interface (MDI) . . . 1-15
1.4.10.4 Layer 1 Timer (L1T) . . . 1-15
1.5 Block Diagram . . . 1-16

Chapter 2
Neptune LTE IC Pin Description (PINS)

2.1 Pin Assignment Listing . . . 2-2
2.2 Detail Pin Description . . . 2-29
2.2.1 AEIM Signals . . . 2-29
2.2.2 Real Time Trace Signals . . . 2-31
2.2.3 Layer 1 Timer Signals . . . 2-32
2.2.4 A2DIGL Mixed Signal Interface . . . 2-33
2.2.5 External Interrupt Signals . . . 2-34
2.2.6 SAP Signals . . . 2-36
2.2.7 GPIO Signals . . . 2-37

2.2.8 SIM Signals. . . . 2-38
2.2.9 Display Memory Access Controller Signals. . . . 2-39
2.2.10 Keypad Port Signals 2-40
2.2.11 Real Time Clock Signals 2-41
2.2.12 Clock Control Signals. . . . 2-41
2.2.13 RF Interface Signals 2-42
2.2.14 MQSPI Signals 2-42
2.2.15 JTAG Signals 2-44
2.2.16 Miscellaneous Control Signals 2-46
2.2.17 USB Signals 2-46
2.2.18 RX and TX Synthesizers. . . . 2-47
2.2.19 UART Signals. . . . 2-48
2.2.20 PA Control 2-49
2.2.21 RX Data Converters 2-50
2.2.22 Audio Codec 2-50
2.2.23 General Purpose A/D 2-51
2.2.24 Power and Ground 2-52
2.2.25 General Purpose Timer 2-53
2.2.26 DSP Debug and Trace. . . . 2-54
2.2.27 DSP Triple Timer 2-55
2.2.28 One Wire Interface 2-55
2.3 Programmable I/O Drive Strength 2-55
2.4 Power Supply Description 2-57
2.5 GPIO Port Mux Assignments 2-59
2.5.1 Port A Mux Assignments 2-60
2.5.2 Port B Mux Assignments 2-61
2.5.3 Port C Mux Assignments 2-62
2.5.4 Port D Mux Assignments 2-63
2.5.5 Port E Mux Assignments 2-64

Chapter 3
Clocks, Low Power Control and Reset (CLKRST)

3.1 Clock Sources 3-2
3.2 MCU Clock Generation 3-5
3.3 DSP CLKGEN (DCLKG) - Block Diagram and Control. . . . 3-7
3.3.1 Low Power Divider (LPD) 3-9
3.3.2 Clock Select 3-10
3.3.3 Divide by 2 3-10
3.3.4 DSP Operating Frequency 3-10
3.3.5 DSP Peripherals Clock Sources 3-10
3.4 Low Power Modes 3-11
3.4.1 Introduction. . . . 3-11
3.4.2 MCU Low Power Modes 3-11
3.4.3 DSP Low Power Modes 3-11
3.4.4 General Low Power Features 3-12
3.4.4.1 REF_DPLL Shut Down 3-12
3.4.4.2 Crystal Oscillator (CROSC) Output Driver Shut Down. . . . 3-12

3.4.5 MCU Low Power Modes operation . . . 3-12
3.4.5.1 Entering Low Power . . . 3-12
3.4.5.2 Exiting Low Power . . . 3-12
3.4.5.3 Alternate Bus Master During Low Power . . . 3-12
3.5 Reset . . . 3-13
3.5.1 Physical Reset Pin(s) . . . 3-13
3.5.2 Other Reset sources . . . 3-13
3.5.3 Reset function/sequence . . . 3-13
3.5.3.1 MCU Reset . . . 3-14
3.5.3.2 DSP Reset . . . 3-14
3.5.3.3 REF_DPLL operation after reset . . . 3-14

Chapter 4
USB Digital Phase Lock Loop (DPLL)

Chapter 5
Clock Control Module (CCM)

5.1 Overview . . . 5-2
5.2 Register Summary . . . 5-9
5.3 Detailed Register Descriptions . . . 5-10
5.3.1 Clock and PLL Registers . . . 5-11
5.4 DPLL Control . . . 5-26
5.4.1 x_CTL, x_OP, x_MFD Register Update . . . 5-26
5.4.2 x_MFN Register Update . . . 5-27
5.4.3 DPLL Reset Logic . . . 5-27
5.5 Clock Control . . . 5-28
5.5.1 Clock Cleaner - Deep Sleep Mode . . . 5-28
5.5.2 Clock Muxes . . . 5-30
5.5.3 Test Mode . . . 5-30
5.5.4 Clock Dividers . . . 5-31
5.6 Clock Characteristics . . . 5-32
5.7 Low Power Mode . . . 5-32
5.7.1 Entering a low power mode . . . 5-33
5.7.2 Restoring Clocks while in low-power mode . . . 5-33
5.7.3 Exiting a low power mode . . . 5-34
5.8 CKO(H) Selection . . . 5-35
5.9 Reset Logic . . . 5-37
5.9.1 Synchronized System Reset Deassertion . . . 5-39
5.10 Synchronized PAT_REF generation . . . 5-40
5.11 Bus Interface . . . 5-40
5.12 CKIL Clock Monitor (32.768 kHz Nominal) . . . 5-40
5.13 CCM Desense Circuit . . . 5-41

Chapter 6
Chip Configuration and Memory Maps (MEMMAP)

6.1 Introduction . . . 6-3
6.2 DSP Memory Space . . . 6-4
6.2.1 DSP Peripheral I/O Space . . . 6-4
6.2.2 DSP56600 Patch Jump Targets . . . 6-9
6.2.3 DSP Memory Space . . . 6-9
6.2.3.1 DSP Program Memory Space . . . 6-9
6.2.3.2 DSP X and Y Memory Spaces . . . 6-10
6.2.4 Detailed Register Descriptions for DSP Core . . . 6-11
6.2.5 DSP Interrupt Controller . . . 6-22
6.2.6 DSP56600 DMA Interface . . . 6-28
6.2.6.1 DMA to Memory Subsystem Interface . . . 6-28
6.2.6.2 DMA (s) to Peripheral Subsystem Interface . . . 6-28
6.3 MCU Memory Space . . . 6-30
6.3.1 On-Chip MCU ROM . . . 6-30
6.3.2 On-Chip MCU RAM . . . 6-31
6.3.3 MCU External Memory . . . 6-32
6.3.4 MCU Peripherals . . . 6-32
6.3.4.1 MDPI . . . 6-33
6.3.4.2 AIPI . . . 6-33
6.3.4.3 AHB . . . 6-33
6.3.4.4 MDI . . . 6-33
6.3.4.5 AEIM . . . 6-33
6.3.5 MCU Memory Map . . . 6-33
6.3.6 MCU Peripheral Interrupts . . . 6-63
6.4 Mixed-Signal Modules Interconnections . . . 6-64
6.5 MAGIC Serial Communication Protocol Enable on BBP pins . . . 6-68
6.6 DSP Debug Configuration . . . 6-70
6.7 Debug Event Assertion Timing . . . 6-71
6.8 IO Pad Drive Programming . . . 6-72
6.9 USB Module Configuration . . . 6-78
6.10 UART2 Module Configuration . . . 6-78
6.11 SCC Debug Signal Configuration . . . 6-78

Chapter 7
Synthesizable ONYXU (SONYXU)

7.1 Introduction . . . 7-1
7.1.1 High Performance DSP56600 Core . . . 7-2
7.1.2 DSP Core Low Power support . . . 7-2
7.2 Core Registers . . . 7-2
7.3 Instructions . . . 7-2
7.3.1 Instruction Types . . . 7-2
7.3.2 Addressing Modes . . . 7-2
7.3.3 Instruction Descriptions . . . 7-2

7.3.4 Opcode Map . . . 7-2
7.4 Instruction Execution . . . 7-3
7.4.1 Normal Instruction Execution . . . 7-3
7.4.2 Execution sequence . . . 7-3
7.4.3 Changes of flow . . . 7-3
7.4.4 Instruction Timing . . . 7-3
7.5 Exception Processing . . . 7-3
7.6 Core Interface . . . 7-3
7.6.1 Core Interface Overview . . . 7-4
7.7 SONYXU Module Pin List . . . 7-5
7.8 Signal Descriptions . . . 7-9
7.8.1 Program Memory Interface Signals . . . 7-9
7.8.1.1 Program Memory Read (apmrpm) . . . 7-9
7.8.1.2 Program Memory Write (apwr) . . . 7-9
7.8.1.3 Internal Program Memory Enable (bpim_b) . . . 7-9
7.8.1.4 Program Memory Read Data Hold (bmemca) . . . 7-9
7.8.1.5 Program Memory Address Bus (pab[15:0]) . . . 7-9
7.8.1.6 Program Memory Write Data Bus (pdb_wr[23:0]) . . . 7-9
7.8.1.7 Program Memory Read Data Bus (pdb_rd[23:0]) . . . 7-9
7.8.2 External Memory Interface (Port A) Signals . . . 7-10
7.8.2.1 Address Trace (bat_b) . . . 7-10
7.8.2.2 Data Output Enable (bdatoe) . . . 7-10
7.8.2.3 Chip Select (bmcs_b) . . . 7-10
7.8.2.4 Read Strobe (brd_b) . . . 7-10
7.8.2.5 Write Strobe (bwr_b) . . . 7-10
7.8.2.6 Port A Address Bus (bab[15:0]) . . . 7-10
7.8.2.7 Port A Write Data Bus (bdio_wr[23:0]) . . . 7-10
7.8.2.8 Port A Read Data Bus (bdio_rd[23:0]) . . . 7-10
7.8.3 Data Memory Interface Signals . . . 7-10
7.8.3.1 X Memory Read (arxrd) . . . 7-10
7.8.3.2 X Memory Write (arxwr) . . . 7-10
7.8.3.3 X Memory Address Bus (xab[15:0]) . . . 7-10
7.8.3.4 X Memory Write Data Bus (xdb_wr[15:0]) . . . 7-11
7.8.3.5 X Memory Read Data Bus (xdb_rd[15:0]) . . . 7-11
7.8.3.6 Y Memory Read (aryrd) . . . 7-11
7.8.3.7 Y Memory Write (arywr) . . . 7-11
7.8.3.8 Y Memory Address Bus (yab[15:0]) . . . 7-11
7.8.3.9 Y Memory Write Data Bus (ydb_wr[15:0]) . . . 7-11
7.8.3.10 Y Memory Read Data Bus (ydb_rd[15:0]) . . . 7-11
7.8.4 Clock and Reset Interface Signals . . . 7-11
7.8.4.1 System Clock (gclkw) . . . 7-11
7.8.4.2 Core Reset (irs_b) . . . 7-11
7.8.4.3 Core Reset due to Voltage Difference (io_jdish) . . . 7-11
7.8.5 Core Interrupt Interface Signals . . . 7-12
7.8.5.1 Interrupt Request A (irqmda_b) . . . 7-12
7.8.5.2 Interrupt Request B (irqmdb_b) . . . 7-12
7.8.5.3 Interrupt Request C (irqmdc_b) . . . 7-12

7.8.5.4 Interrupt Request D (irqmdd_b) . . . 7-12
7.8.5.5 Non-Maskable Interrupt Request (ipinmi_b) . . . 7-12
7.8.6 Test Interface Signals . . . 7-12
7.8.6.1 Scan Mode (scan_mode) . . . 7-12
7.8.6.2 Scan Mode for TCK Logic (scan_mode_tck) . . . 7-12
7.8.6.3 Scan Mode for Bus Logic (scan_mode_bus) . . . 7-13
7.8.6.4 Scan Enable (scan_en) . . . 7-13
7.8.6.5 Test Reset (test_reset) . . . 7-13
7.8.6.6 Test Reset for TCK Logic (test_reset_tck) . . . 7-13
7.8.6.7 Test Reset for Bus Logic (test_reset_bus) . . . 7-13
7.8.6.8 Test Clock (test_clk) . . . 7-13
7.8.6.9 Scan Divergence Mode (tcm_scan_divergence) . . . 7-13
7.8.6.10 Scan Divergence In (scan_sonyxu_div_in) . . . 7-13
7.8.6.11 Scan Data In (scan_sonyxu_sdi_1 .. scan_sonyxu_sdi_27) . . . 7-13
7.8.6.12 Scan Data In Bypass (scan_sonyxu_sdi_1_bypass .. scan_sonyxu_sdi_27_bypass) . . . 7-13
7.8.6.13 Scan Data Out (sonyxu_sdo_1 .. sonyxu_sdo_27) . . . 7-14
7.8.7 Synthesizable Peripheral Module Bus (SPMB) Signals . . . 7-14
7.8.7.1 I/O Space Access (mcacc) . . . 7-14
7.8.7.2 X I/O Space Read (mcselx_rd) . . . 7-14
7.8.7.3 X I/O Space Write (mcselx_wr) . . . 7-14
7.8.7.4 Y I/O Space Read (mcsely_rd) . . . 7-14
7.8.7.4.1 Y I/O Space Write (mcsely_wr) . . . 7-14
7.8.7.5 I/O Space Read Address Bus (mcab_rd[6:0]) . . . 7-14
7.8.7.6 I/O Space Write Address Bus (mcab_wr[6:0]) . . . 7-14
7.8.7.7 I/O Space Write Data Bus (gdb_wr[15:0]) . . . 7-14
7.8.7.8 I/O Space Read Data Bus (gdb_rd[15:0]) . . . 7-14
7.8.7.9 Peripheral Interrupt Vector Read (pivrd) . . . 7-14
7.8.7.10 Peripheral Interrupt Acknowledge (pivack) . . . 7-15
7.8.7.11 Peripheral NMI Vector Select (pinmivse) . . . 7-15
7.8.7.12 Peripheral Interrupt Vector Selects (pivsse[7:0]) . . . 7-15
7.8.7.13 Peripheral Non-Maskable Interrupt Request (mnmirq_b) . . . 7-15
7.8.7.14 Peripheral Interrupt Requests (mirq_b[7:0]) . . . 7-15
7.8.7.15 Peripheral Interrupt Vector Bus (vab_rd[7:1]) . . . 7-15
7.8.7.16 Peripheral Software Reset (pires) . . . 7-15
7.8.7.17 System Hardware Reset (res) . . . 7-15
7.8.7.18 Core Stall Phase 0 (mcdis0) . . . 7-15
7.8.7.19 Core Stall Phase 1 (mcdis1) . . . 7-15
7.8.7.20 Core Debug Mode (ndbgen1) . . . 7-16
7.8.7.21 Core Going to Stop (bcstop) . . . 7-16
7.8.8 Configuration Interface Signals . . . 7-16
7.8.8.1 Disable OnCE (nsec) . . . 7-16
7.8.8.2 OMR Enable (omr_en[15:0]) . . . 7-16
7.8.8.3 Program Space Internal/External Boundary (pmem_bdry[7:0]) . . . 7-16
7.8.8.4 IDR Value (bid[15:0]) . . . 7-16
7.8.8.5 JTAG ID (nidvia[31:0]) . . . 7-16
7.8.8.6 RESET1 Configuration (paviares1) . . . 7-16

7.8.8.7 RESET2 Configuration (paviares2) . . . 7-16
7.8.8.8 RESET3 Configuration (paviares3) . . . 7-17
7.8.8.9 RESET3 Select (reset3_sel[1:0]) . . . 7-17
7.8.9 JTAG/OnCE Interface Signals . . . 7-17
7.8.9.1 Debug Event (idreq_b) . . . 7-17
7.8.9.2 Debug Acknowledge (nack) . . . 7-17
7.8.9.3 Debug Acknowledge — Active Low (nack_b) . . . 7-17
7.8.9.4 Disable Inputs (nodis_b) . . . 7-17
7.8.9.5 Tristate Outputs (ntrist) . . . 7-17
7.8.9.6 System Reset due to JTAG Instruction (nmode2) . . . 7-17
7.8.9.7 JTAG DEBUG_REQUEST Instruction (njdreq_b) . . . 7-17
7.8.9.8 General Test Mode (ngtm) . . . 7-18
7.8.9.9 Serial Test Mode (nsertst) . . . 7-18
7.8.9.10 Clock Test Mode (nctst) . . . 7-18
7.8.9.11 JTAG Boundary Scan Register Output (ibsro) . . . 7-18
7.8.9.12 JTAG Boundary Scan Register Control (nclockdr) . . . 7-18
7.8.9.13 JTAG Shift-DR Control (nshiftdr) . . . 7-18
7.8.9.14 JTAG Update-DR Strobe (nupdatedr) . . . 7-18
7.8.9.15 JTAG/OnCE Test Clock (itck) . . . 7-18
7.8.9.16 JTAG/OnCE Test Mode Select (itms) . . . 7-18
7.8.9.17 JTAG/OnCE Test Data In (itdi) . . . 7-18
7.8.9.18 JTAG/OnCE Test Reset (itrst_b) . . . 7-18
7.8.9.19 Power-On Reset (coscen_b) . . . 7-18
7.8.9.20 JTAG/OnCE Test Data Out (ntdo_p) . . . 7-19
7.8.9.21 JTAG/OnCE Test Data Output Enable (ntdo_ena) . . . 7-19
7.8.10 Clock Control Interface Signals . . . 7-19
7.8.10.1 Processing Wait (pwait) . . . 7-19
7.8.10.2 Exit Wait due to Interrupt Request (pint) . . . 7-19
7.8.10.3 Exit Wait due to Debug Request (ndreq) . . . 7-19
7.8.10.4 Processing Stop (pstop) . . . 7-19
7.8.10.5 PLL Enable Initial Value (tpinit_b) . . . 7-19
7.8.10.6 Clock Wait (cwait) . . . 7-19
7.8.10.7 Clock Disable for Wait (clk_dis) . . . 7-19
7.8.11 Miscellaneous Outputs Signals . . . 7-19
7.8.11.1 System Hardware Reset — Active Low (res_b) . . . 7-19
7.8.11.2 OMR Bit 6 (paomr6) . . . 7-20
7.8.11.3 OMR Bit 13 (paomr13) . . . 7-20
7.9 DSP Address Trace Timing . . . 7-20

Chapter 8
DSP Memory (DSPMEM)

8.1 Overview . . . 8-1
8.1.1 Modes of Operation . . . 8-1
8.2 Functional Description . . . 8-2
8.2.1 Program Memory Operation . . . 8-3
8.2.2 X And Y Data Memory Operation . . . 8-3
8.2.2.1 Dual Port RAM Block . . . 8-3

8.2.2.2 Interleaved RAM Block . . . 8-5
8.2.2.3 ROM Block . . . 8-6
8.3 Initialization/Application Information . . . 8-6

Chapter 9
DSP Clock Generator (DCLKG_LOGIC)

9.1 Introduction . . . 9-2
9.2 DCLKG_LOGIC Module Pin List . . . 9-3
9.3 DCLKG_LOGIC Internal Blocks . . . 9-6
9.3.1 Internal Blocks Functionality . . . 9-8
9.3.1.1 DCLKG_LOGIC . . . 9-8
9.3.1.2 clkgen . . . 9-8
9.4 Modes of Operation . . . 9-8
9.5 Programming Model . . . 9-11
9.5.1 Detailed Register Descriptions . . . 9-11
9.6 Restrictions . . . 9-14

Chapter 10
DSP Special Interrupt Handler (DSIH)

10.1 Introduction . . . 10-1
10.2 Pin Description . . . 10-2
10.3 Functional Description . . . 10-5
10.4 Timing Diagram . . . 10-5

Chapter 11
Power on Reset (POR)

11.1 General Description . . . 11-1
11.2 Functional Description . . . 11-1
11.2.1 POR Module Pin List . . . 11-2

Chapter 12
Synthesizable PMBIF (SMPBIF)

12.1 Overview . . . 12-1
12.2 New Custom Module Structure . . . 12-1
12.3 Synthesisable Peripheral Module Bus(S-PMB) . . . 12-2
12.3.1 S-PMB Signals Description . . . 12-3
12.4 S-PMB Interface . . . 12-4
12.4.1 SPMBIF Internal Clock Logic . . . 12-4
12.4.2 Address Decoder . . . 12-5
12.4.2.1 Write Control Signals . . . 12-5
12.4.2.2 Read Control Signals . . . 12-7
12.4.3 Data Buffers . . . 12-8
12.4.3.1 Core Data Bus Read Buffer . . . 12-8
12.4.4 The Interrupt Interface . . . 12-9
12.4.4.1 The Interrupt Request . . . 12-9

12.4.4.2 The Vector Address Bus . . . 12-10
12.4.5 The Scan Interface . . . 12-11
12.4.5.1 The Scan Mode (scan_mode) Signal . . . 12-12
12.4.5.2 The Scan Enable Pin (scan_en) . . . 12-12
12.4.5.3 The Scan Clock Source (scan_clk) . . . 12-13
12.4.6 The SCAN_RES Signal . . . 12-13
12.5 Directly Connected Signals . . . 12-14
12.5.1 The DMA Request Signal . . . 12-14
12.5.2 The Core Write Bus (gdb_wr) . . . 12-14
12.5.3 The DMA Write Bus (mddb_wr) . . . 12-15
12.5.4 The BCSTOP Signal . . . 12-15
12.5.5 The PIRES Signal . . . 12-15
12.5.6 The RES Signal . . . 12-15
12.5.7 Shared Memory Signals . . . 12-15
12.6 S-PMB Interface Timing Diagrams . . . 12-15
12.6.1 Core Write Access . . . 12-16
12.6.2 Core Read Access . . . 12-16
12.6.3 DMA Write Access . . . 12-17
12.6.4 DMA Read Access . . . 12-17
12.6.5 Shared Memory Write Access . . . 12-18
12.6.6 Shared Memory Read Access . . . 12-18
12.6.7 Interrupts . . . 12-18
12.6.8 The DMA Request . . . 12-19
12.6.9 The NDBGCTL Signal . . . 12-21
12.7 The Customer Specific Logic Module . . . 12-22
12.8 Customer Specific Logic Hardware Design . . . 12-24
12.8.1 General Restrictions . . . 12-24
12.8.2 Clock Signals . . . 12-24
12.8.2.1 Rising and Falling Edge . . . 12-24
12.8.2.2 Low Power Considerations . . . 12-24
12.9 Test Mode Considerations . . . 12-25
12.9.1 Asynchronous Resets . . . 12-25
12.10 Verilog Code Examples . . . 12-25
12.10.1 Read and Write Registers through PMB . . . 12-25
12.10.1.1 Write Accessible Registers . . . 12-26
12.10.1.2 Read Accessible Registers . . . 12-26
12.10.1.3 DMA Accessible Registers . . . 12-28
12.10.2 Flag Logic Management . . . 12-29
12.10.2.1 Switching Flags Using Access Enable Signals . . . 12-29
12.10.2.2 Sticky Status Bit Management . . . 12-29
12.10.3 Interrupt Logic . . . 12-29
12.10.4 Clocks . . . 12-30
12.10.4.1 The Clock Logic . . . 12-31
12.10.4.2 Clock Disable Logic . . . 12-31
12.10.5 Shared Memory Logic . . . 12-31

Chapter 13
Layer 1 Encryption Module (LEM)

13.1 Module Overview . . . 13-1
13.1.1 General Description . . . 13-1
13.1.1.1 A5 Ciphering . . . 13-1
13.1.1.2 FIRE Encode/Decode . . . 13-2
13.1.2 Connectivity . . . 13-2
13.2 LEM Module Pin List . . . 13-3
13.3 Encryption Module Architecture . . . 13-5
13.4 LEM Block Description . . . 13-6
13.4.1 Control Block . . . 13-6
13.4.1.1 FIRE Encode FSM Branch . . . 13-9
13.4.1.2 FIRE Decode FSM Branch . . . 13-10
13.4.1.3 Cipher Block Generation FSM Branch . . . 13-13
13.4.2 Register Block . . . 13-14
13.4.2.1 CNTRL Register . . . 13-16
13.4.2.2 STAT Register . . . 13-16
13.4.3 Cipher Block . . . 13-17
13.4.3.1 Description . . . 13-17
13.4.4 FED Block (Fire Encode/Decode Block) . . . 13-18
13.4.4.1 Description . . . 13-18
13.4.4.2 FIRE Encode Process . . . 13-19
13.4.4.3 FIRE Decode Process . . . 13-19
13.5 Programming Model . . . 13-20
13.5.1 Cipher Block Programming . . . 13-20
13.5.1.1 Cipher Block Programming Flow Diagram . . . 13-22
13.5.2 FED Block Programming . . . 13-24
13.5.2.1 FED Block Programming Flow Diagram . . . 13-26
13.5.2.2 Low Power Mode Behavior . . . 13-28
13.6 LEM Register Summary . . . 13-28
13.6.1 Register Summary . . . 13-28
13.6.2 Register Description . . . 13-29

Chapter 14
Base Band Port (BBP)

14.1 BBP changes from RSI . . . 14-1
14.2 Introduction . . . 14-2
14.2.1 BBP Module Pin List . . . 14-3
14.3 BBP Data and Control Pins . . . 14-7
14.3.1 Serial Transmit Data Pin (STDB) . . . 14-7
14.3.2 Serial Receive Data Pin (SRDB) . . . 14-7
14.3.3 Serial Clock (SCKB) . . . 14-7
14.3.4 Serial Control Pin (SC0B) . . . 14-8
14.3.5 Serial Control Pin (SC1B) . . . 14-8
14.3.6 Serial Control Pin (SC2B) . . . 14-9
14.4 BBP PROGRAMMING MODEL . . . 14-9

14.4.1 Register Summary . . . 14-9
14.4.2 Detailed Register Descriptions . . . 14-10
14.4.3 BBP Receive Shift Register . . . 14-32
14.4.4 BBP Transmit Shift Register . . . 14-32
14.5 Operating Modes . . . 14-32
14.5.1 BBP After Reset . . . 14-32
14.5.2 BBP Initialization . . . 14-32
14.5.3 BBP Exceptions . . . 14-33
14.5.4 Operating Modes – Normal, Network, and On-Demand . . . 14-34
14.5.4.1 Normal/Network/On-Demand Mode Selection . . . 14-34
14.5.4.2 Synchronous/Asynchronous Operating Modes . . . 14-34
14.5.4.3 Frame Sync Selection . . . 14-35
14.5.4.4 Shift Direction Selection . . . 14-35
14.5.4.5 Counters Operating Modes . . . 14-36
14.5.5 Flags . . . 14-36
14.6 BBP MAGIC Communication . . . 14-36
14.7 Neptune LTE Specific Requirement . . . 14-37

Chapter 15
DSP Timer Module (Dtimer)

15.1 Introduction . . . 15-1
15.2 Architecture . . . 15-2
15.2.1 DSP Timer Module Block Diagram . . . 15-2
15.2.2 Timer Block Diagram . . . 15-2
15.2.3 Pin Diagram . . . 15-3
15.3 DSP Timer Module Register Summary . . . 15-6
15.3.1 Detailed Register Descriptions . . . 15-7
15.4 Timer Functional Description (Modes of Operation) . . . 15-19
15.4.1 Timer Modes . . . 15-19
15.4.1.1 Timer mode, internal clock, no output (mode 0) . . . 15-19
15.4.1.2 Timer mode, internal clock, output pulse enable (mode 1) . . . 15-19
15.4.1.3 Timer mode, internal clock, output toggle enable (mode 2) . . . 15-20
15.4.1.4 Timer mode, external clock, event counter (mode 3) . . . 15-20
15.4.2 Measurement Modes . . . 15-20
15.4.2.1 Pulse width measurement mode (mode 4) . . . 15-20
15.4.2.2 Period measurement mode (mode 5) . . . 15-21
15.4.2.3 Capture mode (mode 6) . . . 15-21
15.4.2.4 Measurement modes exactness . . . 15-21
15.4.3 PWM Modes . . . 15-21
15.4.3.1 Pulse Width Modulation mode, internal clock, output toggle enable (mode 7) . . . 15-21
15.4.4 Watchdog Modes . . . 15-22
15.4.4.1 Watchdog mode, internal clock, output pulse enable (mode 9) . . . 15-22
15.4.4.2 Watchdog mode, internal clock, output toggle enable (mode 10) . . . 15-22
15.4.5 Reserved Modes . . . 15-23
15.4.6 Special Cases . . . 15-23
15.4.6.1 Timer behavior during WAIT and STOP . . . 15-23

Chapter 16
Serial Audio Codec Port (SAP)

16.1 SAP changes from RSI 16-2
16.1.1 Binary Rate Multiplier (BRM) 16-2
16.1.2 General Purpose Timer 16-2
16.2 Introduction 16-2
16.3 SAP Module Pin List 16-3
16.3.1 SAP Pin Diagram 16-6
16.3.1.1 SAP Connectivity Diagram 16-7
16.4 SAP Data and Control Pins 16-7
16.4.1 Serial Transmit Data Pin (STDA) 16-7
16.4.2 Serial Receive Data Pin (SRDA) 16-8
16.4.3 Serial Clock (SCKA) 16-8
16.4.4 Serial Control Pin (SC0A) 16-8
16.4.5 Serial Control Pin (SC1A) 16-9
16.4.6 Serial Control Pin (SC2A) 16-9
16.5 SAP Programming Model 16-10
16.5.1 Register Summary 16-10
16.5.2 Detailed Register Descriptions 16-11
16.5.3 SAP Receive Shift Register 16-36
16.5.4 SAP Transmit Shift Register 16-36
16.6 Operating Modes 16-37
16.6.1 SAP After Reset 16-37
16.6.2 SAP Initialization 16-37
16.6.3 SAP Exceptions 16-38
16.6.4 Operating Modes – Normal, Network, and On-Demand 16-38
16.6.4.1 Normal/Network/On-Demand Mode Selection 16-38
16.6.4.2 Synchronous/Asynchronous Operating Modes 16-39
16.6.4.3 Frame Sync Selection 16-39
16.6.4.4 Shift Direction Selection 16-40
16.6.4.5 Timer Operating Mode 16-40
16.6.5 Flags 16-40
16.7 Appendix: 16-41

Chapter 17
DSP Peripherals DMA (DMA)

17.1 DMA Module Pin List 17-4
17.2 DSP Peripherals DMA Architecture 17-7
17.3 Programming Model 17-8
17.3.1 Register Summary 17-8
17.3.2 Detailed Register Descriptions 17-9
17.3.3 DMA Buffer Counter (DBCN) 17-20
17.3.4 DMA Time Out Counter (DTOCN) 17-21
17.3.5 DMA Data Buffer (DDB) 17-21
17.4 Description of the Operation 17-22
17.4.1 DMA Initialization 17-22

17.4.2 Operating DMA with BBP and SAP . . . 17-22
17.4.3 Peripheral to DSP-memory data transfer . . . 17-22
17.4.4 DSP-memory to Peripheral data transfer . . . 17-23
17.4.5 DMA processing cycle times . . . 17-24
17.4.6 DSP Low Power Modes . . . 17-24
17.5 Programmers Notes . . . 17-24

Chapter 18
DMA Arbitration (DMA_ARB)

18.1 Overview . . . 18-1
18.1.1 Interrupt Arbitration between DMA Controllers . . . 18-3

Chapter 19
Viterbi Accelerator (VIAC)

19.1 Introduction . . . 19-2
19.2 Key Features . . . 19-3
19.3 Module Interface . . . 19-4
19.4 VIAC Pin Description . . . 19-5
19.5 Performance analysis . . . 19-6
19.5.1 VIAC's Organization . . . 19-7
19.5.2 ACS - Add Compare Select . . . 19-8
19.5.3 WED - Window Error Detection . . . 19-8
19.5.4 VIAC's pipeline . . . 19-9
19.5.4.1 VIAC's pipeline in lockstep mode . . . 19-9
19.5.4.2 VIAC's pipeline in independent mode . . . 19-11
19.5.5 Path Metric RAM access using VPMAR FIFO . . . 19-14
19.5.6 The Viterbi butterfly implementation . . . 19-14
19.5.6.1 Handling overflow in path metrics . . . 19-15
19.5.7 DMA buffers organization . . . 19-16
19.6 Programming Model . . . 19-26
19.6.1 Viterbi Accelerator (VIAC) Register Summary . . . 19-26
19.6.2 Detailed Register Descriptions . . . 19-27
19.6.3 VIAC DMA Output Channel Base Address (VDOBA) . . . 19-46
19.6.4 VIAC DMA Output Channel Current Address (VDOCA) . . . 19-47
19.7 Operating Modes . . . 19-47
19.7.1 General description . . . 19-47
19.7.2 VIAC states . . . 19-47
19.7.3 Trellis processing in equalization . . . 19-50
19.7.3.1 Preparing the VIAC for trellis main loop . . . 19-50
19.7.3.2 trellis main loop . . . 19-51
19.7.3.3 Trellis Completion . . . 19-52
19.7.3.3.1 Notifying the DSP . . . 19-52
19.7.3.3.2 Reading Path Metric RAM . . . 19-52
19.7.4 Trellis processing in convolutional decoding . . . 19-52
19.7.4.1 Preparing the VIAC for trellis main loop . . . 19-52
19.7.4.2 trellis main loop . . . 19-54

19.7.4.3 Trellis Completion 19-55
19.7.4.3.1 Notifying the DSP 19-55
19.7.4.3.2 Reading Path Metric RAM 19-55
19.7.4.3.3 Reading WED 19-55
19.7.5 VIAC's interrupts 19-55

Chapter 20
DSP Debug Module (PDDM)

20.1 Introduction 20-2
20.2 PDDM Module Pin List 20-2
20.3 DSP Memory Paging 20-4
20.4 Data Watchpoint (DW) 20-4
20.5 Program Address Trace (PAT) 20-5
20.6 Program Watchpoint (PW) 20-6
20.6.1 Program Watchpoint Controls 20-6
20.7 PDDM Patch Logic 20-6
20.8 Programming Model 20-6
20.8.1 Register Summary 20-6
20.8.2 Detailed Register Descriptions 20-8
20.9 DDM Interrupt 20-23
20.10 Restrictions 20-23
20.10.1 Using Read-Modify-Write instructions (bset, bclr) 20-23
20.10.2 Fetches From Program Memory Matching 20-24
20.10.3 Paged Memory Access Delay after PCR Write 20-24
20.10.4 Program Watch Point Delay After DCSR Write 20-24
20.10.5 Paging Bits Delay After OMR Write 20-24
20.10.6 Changing The EOMS Bits in DCSR 20-24
20.10.7 Changing PACMPR 20-24
20.10.8 Patch Generation Restriction 20-24
20.11 Mismatch at chip periphery (in Indy) 20-25

Chapter 21
Universal Serial Bus Interface (USB)

21.1 Description 21-2
21.2 Module Overview 21-2
21.2.1 Features 21-2
21.2.2 Background 21-2
21.3 USB Module Pin List 21-6
21.4 Transceiver Module Pin List 21-10
21.5 USB Module Operation 21-10
21.6 Packet Fields 21-12
21.6.1 Sync Pattern 21-12
21.6.2 Packet Identifier Field 21-13
21.6.3 Address Field (ADDR) 21-13
21.6.4 Endpoint Field (ENDP) 21-13
21.6.5 Cyclic Redundancy Check (CRC) 21-13

21.6.6 End Of Packet (EOP) . . . 21-15
21.7 Signaling Description . . . 21-16
21.7.1 Data Encoding/Decoding . . . 21-16
21.7.2 Bit Stuffing . . . 21-17
21.7.3 Reset Signaling . . . 21-19
21.7.4 Suspend / Resume . . . 21-20
21.7.5 Supported USB Device Speeds . . . 21-20
21.7.5.1 Low Speed Device . . . 21-20
21.7.5.2 Full Speed Device . . . 21-21
21.7.5.3 High Speed Device . . . 21-21
21.8 Hardware/Software Interaction . . . 21-21
21.9 Memory Maps . . . 21-22
21.9.1 Buffer Connections . . . 21-22
21.10 USB Control Registers . . . 21-24
21.10.1 Detailed Register Descriptions . . . 21-26

Chapter 22
Universal Asynchronous Receiver/Transmitter (UART)

22.1 Features . . . 22-2
22.2 UART Module Interface Signals . . . 22-3
22.2.1 UART Module Pin List . . . 22-4
22.3 UART Memory Map . . . 22-6
22.4 Interrupts . . . 22-8
22.5 General UART Definitions . . . 22-9
22.6 RTS - UART Request To Send . . . 22-10
22.6.1 RTS Edge Triggered Interrupt . . . 22-10
22.7 DTR - Data Terminal Ready . . . 22-11
22.7.1 DTR Edge Triggered Interrupt . . . 22-11
22.8 DSR - Data Set Ready . . . 22-12
22.9 DCD - Data Carrier Detect . . . 22-12
22.10 RI - Ring Indicator . . . 22-12
22.11 CTS - Clear To Send . . . 22-12
22.11.1 Programmable CTS de-assertion . . . 22-12
22.12 TXD - UART Transmit . . . 22-13
22.13 RXD - UART Receive . . . 22-13
22.14 Peripheral Bus Interface . . . 22-14
22.15 Sub-block Description . . . 22-14
22.16 Transmitter . . . 22-14
22.17 Transmitter FIFO Empty Interrupt Suppression . . . 22-14
22.18 Receiver . . . 22-16
22.18.1 Idle Line Detect . . . 22-16
22.18.1.1 Idle Condition Detect Configuration . . . 22-16
22.18.2 Receiver Wake Up . . . 22-17
22.18.3 Receiving a Break Condition . . . 22-17
22.18.4 Vote Logic . . . 22-17
22.19 Binary Rate Multiplier (BRM) . . . 22-18
22.19.1 Baud Rate Automatic Detection Logic . . . 22-19

22.19.1.1 Baud Rate Automatic Detection Protocol . . . 22-20
22.19.2 Escape Sequence Detection . . . 22-21
22.20 Infrared Interface . . . 22-22
22.21 UART Register Summary . . . 22-23
22.21.1 Detailed Register Descriptions . . . 22-25
22.22 UART Operation In Low-Power System States . . . 22-54
22.23 UART Operation In System Debug State . . . 22-55
22.24 Verification . . . 22-56
22.24.1 Registers . . . 22-56
22.24.2 Transmitter Functionality . . . 22-56
22.24.2.1 Transmission Modes . . . 22-57
22.24.2.2 Baud Rate Variation . . . 22-58
22.24.2.3 Automatic Baud Rate Detection Mode . . . 22-58
22.24.3 Receiver Functionality . . . 22-58
22.24.3.1 Reception Modes . . . 22-60
22.24.3.2 Baud Rate Variation . . . 22-60
22.24.3.3 Automatic Baud Rate Detection Mode . . . 22-60
22.24.4 Serial Infrared Functionality . . . 22-60
22.24.5 Vote Logic Functionality . . . 22-61
22.24.6 Test Register . . . 22-61
22.24.7 Low Power Mode Test . . . 22-61

Chapter 23
General Purpose Input/Output (GPIO)

23.1 GPIO Introduction . . . 23-2
23.2 GPIO Features . . . 23-2
23.3 GPIO Block Diagram . . . 23-3
23.4 GPIO Functional Description . . . 23-3
23.4.1 External Interrupt Control . . . 23-3
23.4.2 Alternate Output Functionality . . . 23-4
23.4.3 Alternate Input Functionality . . . 23-4
23.4.4 Internal Interrupt Visibility . . . 23-4
23.4.5 DSP Address Trace Mode and DSP Master Mode . . . 23-4
23.4.6 Shared Test Connections . . . 23-5
23.5 GPIO Register Summary . . . 23-5
23.6 Port A Functional Description . . . 23-9
23.7 Port B Functional Description . . . 23-9
23.8 Port C Functional Description . . . 23-9
23.9 Port D Functional Description . . . 23-9
23.10 Port E Functional Description . . . 23-9
23.11 GPIO Special Functions . . . 23-10
23.12 Primary Selection for Multiple Inputs . . . 23-11
23.12.1 Multiple SAP SCKA Connections . . . 23-11
23.12.2 Multiple SAP SC2A Connections . . . 23-12
23.12.3 Multiple SAP SRDA Inputs . . . 23-13
23.12.4 Multiple INT INT4 Inputs . . . 23-14
23.12.5 Multiple INT INT5 Inputs . . . 23-15

23.12.6 Multiple UART1 DTR1 Inputs . . . 23-16
23.12.7 Multiple UART1 RXD1 Inputs . . . 23-17
23.12.8 Multiple UART1 RTS1 Inputs . . . 23-19
23.12.9 Multiple MQSPI MISOB Inputs . . . 23-21
23.13 Internal Interrupt Visibility Table . . . 23-21
23.14 Port Diagrams . . . 23-25
23.14.1 Port A Block Diagrams . . . 23-25
23.14.1.1 Port A, B, C, D and E Pin Logic Diagrams . . . 23-29
23.14.2 Port C Block Diagrams . . . 23-64
23.14.3 Port D Block Diagrams . . . 23-68
23.14.4 Port E Block Diagrams . . . 23-72
23.15 Detailed Register Descriptions (Port A-E) . . . 23-76
23.15.1 Port A Register Description . . . 23-77
23.15.2 Port B-E Register Description . . . 23-95
23.15.3 Other Registers Description . . . 23-117
23.16 GPIO Test Mode Functions . . . 23-119
23.16.1 Bus Master Mode . . . 23-119
23.16.2 Scan Mode . . . 23-119
23.16.3 Bist Mode . . . 23-119
23.16.4 DSP Master Mode . . . 23-119
23.16.5 DSP Address Trace Mode . . . 23-119
23.17 GPIO Test Mode Muxing . . . 23-119
23.18 GPIO BIST Mode Muxing: . . . 23-122
23.19 GPIO Module Pin List . . . 23-126
23.20 Appendix . . . 23-136

Chapter 24
Clock Monitor (CMON)

24.1 System Clock Monitor (CMON, 26 MHz Typical) . . . 24-2
24.1.1 General Description . . . 24-2
24.1.2 Input and Output Description . . . 24-3
24.2 CMON Module Pin List . . . 24-3
24.2.1 Test Modes . . . 24-4

Chapter 25
Voice Codec (VOCOD)

25.1 Introduction . . . 25-1
25.2 Voice Codec Pin Description . . . 25-2
25.2.1 Microphone Inputs (ADC_P and ADC_N) . . . 25-4
25.2.2 Speaker Outputs (DAC_P and DAC_N) . . . 25-4
25.2.3 Power Supply and Reference Signals . . . 25-4
25.2.3.1 AVDD (REG_BYP_CODEC) . . . 25-4
25.2.3.2 VAG_CODEC . . . 25-4
25.2.3.3 VAG_OUT_CODEC . . . 25-4
25.2.3.4 REF_CODEC_P and REF_CODEC_N . . . 25-4
25.2.3.5 RCM (VSS_CODEC) . . . 25-4

25.2.3.6 LVDD . . . 25-4
25.2.3.7 REGUL_TXRX_VOCOD_IBIAS . . . 25-4
25.2.3.8 SGND . . . 25-5
25.2.4 Analog Test signals . . . 25-5
25.2.4.1 ATST_P and ATST_N . . . 25-5
25.2.4.2 ANATEST_VOC[1:0] . . . 25-5
25.2.5 Clock signal - CODEC_CLK . . . 25-5
25.2.6 DSP Interface . . . 25-5
25.2.7 DVOCOD Module Pin List . . . 25-5
25.2.8 AVOCOD Module Pin List . . . 25-7
25.3 Voice Codec Programming Model . . . 25-8
25.3.1 Register Summary . . . 25-9
25.3.2 Detailed Register Descriptions . . . 25-9
25.3.3 Voice Codec Interrupts . . . 25-18
25.4 Voice Codec Design Description . . . 25-19
25.4.1 Voice Codec ADC Operation . . . 25-19
25.4.1.1 ADC Sigma-Delta Modulator . . . 25-20
25.4.1.2 Decimation Filter . . . 25-22
25.4.1.3 Digital Gain Compensator . . . 25-26
25.4.1.4 IIR Low-Pass Filter . . . 25-27
25.4.1.5 IIR High-Pass Filter . . . 25-31
25.4.2 Voice Codec DAC Operation . . . 25-36
25.4.2.1 IIR High-Pass Filter . . . 25-37
25.4.2.2 IIR Low-Pass Filter . . . 25-37
25.4.2.3 Digital Gain Compensator . . . 25-39
25.4.2.4 Interpolation Filter . . . 25-42
25.4.2.5 DAC Sigma-Delta Modulator . . . 25-43
25.4.2.6 Dynamic Matching . . . 25-45
25.4.2.7 DAC Analog Part . . . 25-47
25.5 Voice Codec Electrical Specifications . . . 25-47
25.5.1 Voice Codec ADC Specifications . . . 25-48
25.5.2 Voice Codec DAC Specifications . . . 25-52
25.6 Voice Codec Testing . . . 25-55
25.6.1 Characterization Tests . . . 25-55
25.6.2 Production Tests . . . 25-55
25.6.2.1 Digital Parts Tests . . . 25-55
25.6.2.2 Analog Parts Tests . . . 25-55
25.6.2.3 Output to ANATEST module . . . 25-55
25.6.3 Backup plan . . . 25-55
25.7 REFERENCES . . . 25-55

Chapter 26
Receiver Analog Front End (RxAFE)

26.1 Introduction . . . 26-2
26.2 RxAFE Module Pin List . . . 26-3
26.3 RxAFE Test Modes . . . 26-4
26.4 TXRX Voltage Regulator . . . 26-6

26.5 RxAFE VAG Buffer . . . 26-6
26.6 RxAFE Reference Voltage Generator . . . 26-7
26.7 Baseband Receive I/Q A/Ds . . . 26-8
26.7.1 Specifications . . . 26-8
26.7.2 Timing . . . 26-9
26.7.3 Dither Input . . . 26-10
26.8 Baseband Receive I/Q A/Ds Block Diagram . . . 26-10

Chapter 27
Receive Coprocessor (RxCPROC)

27.1 Introduction . . . 27-2
27.2 RxCPROC Pin List . . . 27-3
27.3 RxCPROC architecture . . . 27-6
27.3.1 Receiver signal processing chain . . . 27-6
27.3.1.1 Noise cancellation block . . . 27-8
27.3.1.2 SINC filter and first down sampling . . . 27-8
27.3.1.3 Droop compensation FIR filter . . . 27-8
27.3.1.4 Anti-aliasing FIR filter . . . 27-9
27.3.1.5 Second down sampling . . . 27-10
27.3.1.6 High pass filter . . . 27-10
27.3.1.6.1 Non DCR mode . . . 27-10
27.3.1.6.2 DCR mode . . . 27-11
27.3.1.7 Complex Quadrature Mixer with Gain and Phase correction . . . 27-13
27.3.1.8 Programmable selectivity filter . . . 27-13
27.3.1.9 Digital Local Oscillator . . . 27-15
27.4 SPI interface . . . 27-16
27.5 BBP interface . . . 27-24
27.5.1 RXI, RXQ Output formats . . . 27-24
27.5.1.1 First GSM format . . . 27-25
27.5.1.2 Second GSM format . . . 27-26
27.5.1.3 Third GSM format . . . 27-26
27.6 Power On Off operation . . . 27-28
27.7 Timings examples . . . 27-28
27.8 RxCPROC test . . . 27-32

Chapter 28
Digital Fast DC Adapt (DCADAPT)

28.1 Introduction . . . 28-2
28.1.1 Overview . . . 28-2
28.1.2 Features . . . 28-3
28.2 Input/Output Pins . . . 28-4
28.2.1 DCADAPT Module Pin List . . . 28-5
28.3 Block Diagram . . . 28-7
28.3.1 Interface to Algae . . . 28-7
28.4 Fast DCADAPT Algorithm . . . 28-9
28.5 SPI Interface . . . 28-10

28.6 Test 28-10

Chapter 29
Receiver Saturation Detection and SDM Dither Generation (RxSDG)

29.1 Introduction. 29-2
29.2 Input and Output Description 29-4
29.2.1 RxSDG Module Pin List. 29-4
29.3 Receiver Saturation Detection Function 29-6
29.4 Receiver Sigma-Delta Modulator Dither Generation 29-11
29.5 Testability 29-14

Chapter 30
Transmit (TX)

30.1 Introduction. 30-3
30.2 TX Module Pin List 30-6
30.3 FN_Mode, DP_Mode 30-11
30.4 Module Operation (Digital Modulation). 30-12
30.4.1 Normal GSM Mode 30-12
30.4.2 Edge Mode 30-12
30.4.3 GSM Dual_port Mode 30-12
30.4.4 Predistortion GSM Mode:. 30-13
30.4.5 PCS (1800MHz) and DCS(1900 MHz) Mode 30-17
30.5 Dual Port D/A Chain(Analog) 30-17
30.5.1 Introduction. 30-17
30.5.2 Dual_Port Port Description. 30-18
30.5.3 Dual-Port Specifications 30-19
30.5.4 Dual_Port Functional Testing. 30-20
30.6 TRSYNT -TX & RX Synthesizer (Analog) 30-22
30.6.1 Introduction. 30-22
30.6.2 Power Supply Specification 30-24
30.6.3 Main Divider Specification. 30-25
30.6.4 FRAC-N Architecture. 30-28
30.6.5 Max. Division Ratios with (P/P+1): 30-28
30.6.6 Phase Detector and Charge Pump Specification. 30-29
30.6.7 ANATEST 30-33
30.7 CROSC- 26 MHz Reference Oscillator (analog) 30-34
30.7.1 Introduction. 30-34
30.7.2 Block Description 30-34
30.7.3 Start-Up. 30-35
30.7.3.1 Summary of Programming Algorithm 30-36
30.7.4 Power down Modes 30-37
30.7.5 Test Modes 30-37
30.7.6 Specifications 30-37
30.7.7 Input and Output Description 30-39

Chapter 31
Power Amplifier Control (PAC)

31.1 Overview . . . 31-2
31.2 PAC Module Pin List . . . 31-5
31.3 Digital PAC . . . 31-12
31.3.1 PAC Reset . . . 31-12
31.3.2 Battery Save . . . 31-12
31.3.3 Low Battery Detection . . . 31-13
31.3.4 PAC Clocks And Timing Diagrams . . . 31-13
31.3.5 Ramp Down . . . 31-15
31.3.6 GSM Look Up Tables . . . 31-16
31.3.7 Activity Detector . . . 31-16
31.3.8 SAT Detector And Waveform Generator . . . 31-17
31.3.9 Proportional and Integral Control . . . 31-18
31.4 DMCS Counters And Hold-Off Timers . . . 31-20
31.5 PAC Analog Blocks . . . 31-20
31.5.1 ADC_CTL . . . 31-20
31.5.2 PAC_ANA Analog Blocks . . . 31-20
31.5.3 PACII Power Detector . . . 31-21
31.5.4 PAC Input AMP, Comparator, and ADC . . . 31-23
31.5.5 Difference Buffer . . . 31-23
31.5.6 Activity Detect . . . 31-24
31.5.7 Low Pass Filter . . . 31-25
31.5.8 PAC 10-bit ADC . . . 31-25
31.5.9 Voltage Offset Requirements for Input Conditioner and PAC ADC . . . 31-26
31.5.10 PAC D/A Chain . . . 31-27
31.6 Analog Test . . . 31-30
31.7 System Operation . . . 31-32
31.7.1 GSM FN Operation . . . 31-32

Chapter 32
Regulators (REGUL)

32.1 Linear Regulator Characteristics . . . 32-2
32.1.1 Test Signal Descriptions . . . 32-2
32.1.2 1.575 V Regulator-regul_core . . . 32-6
32.1.3 PSRR Improvement Regulators . . . 32-8
32.1.4 1.575 V Synthesizer Regulator . . . 32-11
32.1.5 Master Bias Generator . . . 32-13

Chapter 33
Tuning Circuit (TUNEC)

33.1 Introduction . . . 33-1
33.2 TUNEC Pin Description . . . 33-2
33.2.1 TUNEC Module Pin List . . . 33-3
33.2.2 Input Pins . . . 33-4

33.2.3 Output Pins . . . 33-5
33.2.4 Test Pins . . . 33-5
33.2.5 Scan Pins . . . 33-6
33.3 Functional Description . . . 33-6
33.4 Timing Diagrams . . . 33-9

Chapter 34
Digital Phase Lock Loop (DPLL)

34.1 Introduction . . . 34-1
34.2 DPLL Specifications . . . 34-3
34.3 Pins Description . . . 34-5
34.4 Functional description . . . 34-7
34.5 DPLL linear analysis . . . 34-12
34.6 DPLL interface signals . . . 34-15
34.6.1 DPLL reset . . . 34-15
34.6.2 Static control signals . . . 34-17
34.6.2.1 Predivision Factor Minus One (CPD[3:0]) . . . 34-17
34.6.2.2 Multiplication Factor Integer Part (CMFI[3:0]) . . . 34-17
34.6.2.3 Multiplication Factor Fractional Part Denominator(CMFD[23:0]) . . . 34-18
34.6.2.4 Reference Clock Polarity (CRCP) . . . 34-18
34.6.2.5 Phase Lock Mode (CPLM) . . . 34-18
34.6.2.6 BRM Order (CBRMO) . . . 34-18
34.6.3 Clocks and other dynamic signals . . . 34-18
34.6.3.1 Multiplication Factor Fractional Part Numerator (CMFN[23:0]) . . . 34-18
34.6.3.2 Reference Clock (CRCKD) . . . 34-19
34.6.3.3 Double Clock (DPDCK) . . . 34-19
34.6.3.4 Gated Double Clock (DPGDCK) and Double Clock Enable (CDCKEN) . . . 34-20
34.6.3.5 Post-Divider Synchronization (DPPDSYN) . . . 34-20
34.6.3.6 Equivalent Delay Input (DPC) and Output (CKEQ) signal . . . 34-20
34.6.3.7 Lock Ready Flag (DPLRF) . . . 34-20
34.6.4 Test mode signals . . . 34-21

Chapter 35
General Purpose ADC (GPADC)

35.1 General Purpose ADC Block Description . . . 35-2
35.2 General Purpose ADC Input/Output Description . . . 35-3
35.2.1 GPADC Module Pin List . . . 35-3
35.2.2 Testing Pins . . . 35-5
35.3 General Purpose ADC Electrical Specification . . . 35-7
35.4 General Purpose ADC Timing Diagram . . . 35-8

Chapter 36
Mixed-Signal Control Interface (A2DIGL)

36.1 Introduction . . . 36-3
36.2 A2DIGL Block Diagram . . . 36-3

36.3 A2DIGL Module Pin List . . . 36-5
36.4 Mixed Signal Interfaces Summary . . . 36-14
36.4.1 A2DIGL New Modules Summary . . . 36-15
36.4.1.1 RC Tuning Element Control Interface . . . 36-15
36.4.1.1.1 Tuning Element Control Registers . . . 36-17
36.4.1.1.2 Tuning Element Control Detailed Register Descriptions . . . 36-17
36.4.1.2 CROSC Control Interface . . . 36-19
36.4.1.2.1 CROSC SPI bits . . . 36-19
36.4.1.2.2 CROSC Registers . . . 36-20
36.4.1.2.3 CROSC Detailed Register Descriptions . . . 36-20
36.4.1.3 Regulators Control . . . 36-22
36.4.1.4 Tracking Clock Control . . . 36-22
36.4.1.5 General Purpose ADC Control . . . 36-22
36.4.2 Mixed Signal Interface Re-Use Summary . . . 36-23
36.4.2.1 MAGIC-LV Module Re-use . . . 36-23
36.4.2.2 Tomahawk Module Re-use . . . 36-23
36.4.2.3 GCAP3 Module Re-use . . . 36-23
36.5 A2DIGL Control . . . 36-23
36.5.1 Receive and Transmit Modules Interface Control . . . 36-23
36.5.1.1 Support for bypass and control modes . . . 36-24
36.5.1.2 SPI Signal Interface Description . . . 36-27
36.5.1.3 SPI Decoder Requirements . . . 36-27
36.5.1.4 SPI Decoder Operation Description . . . 36-27
36.5.1.5 Receive and Transmit Timing Sequences and Algae SPI Multiplex Operation . . . 36-28
36.5.1.5.1 Receive VLIF mode timing sequences . . . 36-28
36.5.1.5.2 Detailed DC offset correction and SPI mux timing for VLIF . . . 36-35
36.5.1.5.3 Receive DCR mode timing sequence . . . 36-36
36.5.1.5.4 Detailed DC offset correction and SPI mux timing for DCR . . . 36-38
36.5.1.5.5 Transmit Timing Sequence . . . 36-39
36.5.1.6 SPI bits and signals requiring additional logic . . . 36-44
36.5.1.6.1 ADAPT_EN . . . 36-44
36.5.1.6.2 SL_TIMER[3:0] bits . . . 36-45
36.5.1.6.3 DCA_TIMER[3:0] bits . . . 36-45
36.5.1.6.4 FastDigitalAdapt signal . . . 36-45
36.5.1.6.5 EndFastAdapt signal . . . 36-45
36.5.1.6.6 RX_EN_OUT signal . . . 36-45
36.5.1.6.7 Battery Save bits and signals . . . 36-45
36.5.1.6.8 GAINADxa[6:0], GAINADxb[6:0], GainLUPADR, GCSTEP . . . 36-46
36.5.1.6.9 PHASEADxa, PHASEADxb, PhLUPADR[1:0], PHASEADSEL . . . 36-46
36.5.1.6.10 FastBiasCharge signal . . . 36-46
36.5.1.6.11 Saturation detect reset (DET_RESET) . . . 36-46
36.5.1.6.12 Dither reset (DITH_RESET) . . . 36-47
36.5.1.6.13 Regulators Control . . . 36-47
36.5.1.7 SPI Interface Address Decoding . . . 36-48
36.5.1.8 Receive and Transmit Modules Interface Register Summary . . . 36-48
36.5.1.9 Receive and Transmit Module Interface Detailed Register Descriptions . . . 36-49

36.5.1.10 SPI Interface Bit Groups . . . 36-54
36.5.2 A2DIGL and General Purpose ADC Interface . . . 36-74
36.5.2.1 Block Diagram . . . 36-74
36.5.2.2 GPADC Operation . . . 36-74
36.5.2.2.1 Normal Mode of Operation . . . 36-74
36.5.2.2.2 Deep Sleep Mode . . . 36-79
36.5.2.2.3 Test Mode . . . 36-80
36.5.2.3 RTL Sychronization Issues . . . 36-81
36.5.2.4 GPADC Interface Register Summary . . . 36-82
36.5.2.5 GPADC Detailed Register Descriptions . . . 36-83
36.5.3 A2DIGL Test Configuration in BIST Mode . . . 36-95

Chapter 37
MCU-DSP Interface (MDI)

37.1 Introduction . . . 37-4
37.1.1 MDI Module Pin List . . . 37-6
37.2 DSP Side Memory Mapping . . . 37-8
37.2.1 MCU Side Memory Mapping and Access Type Support . . . 37-9
37.2.2 Programmer's Model . . . 37-12
37.2.3 MDI Register Summary . . . 37-15
37.2.4 Detailed Register Descriptions . . . 37-15
37.2.5 Interrupt Sources and Priorities . . . 37-40
37.2.6 Implementing Messaging Protocols . . . 37-41
37.2.7 MCU Freeing the DSP from a Dead-lock Condition . . . 37-43
37.3 Low-Power Mode Behavior, Clocks and Synchronization . . . 37-43
37.3.1 MCU Low-Power Modes . . . 37-43
37.3.2 DSP Low-Power Modes . . . 37-44
37.3.3 The Shared Memory in DSP Stop Mode . . . 37-45
37.4 MDI Reset . . . 37-45
37.5 Simultaneous Access of the DSP and MCU to the Memory . . . 37-46
37.6 MDI Access and Event Update Timing . . . 37-47
37.6.1 Access to the Shared Memory from the DSP Side . . . 37-47
37.6.2 Access to the Shared Memory from the MCU Side . . . 37-47
37.6.2.1 MDI Timing diagram for shared memory Read and Write . . . 37-49
37.6.3 Event Update Timing . . . 37-51
37.7 Software Restrictions Summary . . . 37-52
37.7.1 General Restrictions . . . 37-52
37.7.1.1 MCU and DSP Frequency Ratio Dependency . . . 37-52
37.7.2 Write-After-Write to a Transmit Register . . . 37-52
37.7.2.1 Read-After-Read from a Receive Register . . . 37-52
37.7.2.2 Status Check Before Entering Stop . . . 37-52
37.7.3 Restrictions at the DSP Side . . . 37-52
37.7.3.1 Setting General Interrupt Requests (DGIR0,1) . . . 37-52
37.7.3.2 Proper use of IRQC, IRQD . . . 37-52
37.7.3.3 DSP Pipeline Effects on Registers . . . 37-53
37.7.3.4 Continuous 1-Cycle Accesses to the Shared Memory . . . 37-53
37.7.3.5 TRNM bit restriction while coming out of stop . . . 37-53

37.7.4 Restrictions at the MCU Side . . . 37-53
37.7.4.1 Byte-Wide Writes to the MDI Shared Memory . . . 37-53
37.7.4.2 Writes to MCVR While the MC bit is Set . . . 37-53
37.7.4.3 Setting the DWS bit While it is Set . . . 37-53
37.7.4.4 L1 Timer DSP Interrupt Override . . . 37-53
37.7.4.5 MDI Reset (MDIR, DHR) Restrictions: . . . 37-53

Chapter 38
ARM7 Platform (ARMPLAT)

38.1 Introduction . . . 38-3
38.2 Performance Characteristics . . . 38-3
38.3 ARM7 Platform Submodules . . . 38-3
38.3.1 ARM7TDMI-S (A7S) . . . 38-3
38.3.2 Core Bus Arbitration (CARB) . . . 38-3
38.3.3 R-AHB Mux (AHBMUX) . . . 38-5
38.3.4 ARM External Interface Module (AEIM) . . . 38-5
38.3.5 ARM IPBus Interface (AIPI) . . . 38-5
38.3.6 MCU to Dual Port RAM Peripheral Interface (MDPI) . . . 38-5
38.3.7 ARM Interrupt Controller (AITC) . . . 38-5
38.3.8 ARM Watchpoint Module (AWPT) . . . 38-5
38.3.9 Memory Controller (MEM_CTL) . . . 38-5
38.4 Platform Bus Support . . . 38-6
38.4.1 Internal Bus (R-AHB) . . . 38-6
38.4.2 R-AHB Slave Signal List . . . 38-6
38.4.3 R-AHB / AHB Bus Differences . . . 38-8
38.4.3.1 Unsupported AHB Transactions . . . 38-8
38.4.3.2 AHB / R-AHB Bus Arbitration Differences . . . 38-8
38.4.3.3 HLOCK . . . 38-8
38.4.3.4 R-AHB Delayed Cycle Termination . . . 38-8
38.4.3.5 Lower Order Address Bits . . . 38-10
38.4.3.6 IDLE - SEQ Cycles . . . 38-10
38.4.3.7 Two- Cycle AWPT Aborts . . . 38-10
38.4.3.8 BUSY Cycles . . . 38-10
38.4.4 External Buses . . . 38-11
38.4.5 Memory Size & Protection . . . 38-11
38.4.5.1 Memory Size & Protection Summary . . . 38-12
38.4.5.2 RAM . . . 38-12
38.4.5.3 ROM . . . 38-13
38.4.5.4 ROM Protection . . . 38-13
38.4.5.5 Memory Select Sizes . . . 38-14
38.4.5.6 Effects of Memory Map Space . . . 38-15
38.4.5.7 BIST Sizes . . . 38-16
38.4.5.8 Examples . . . 38-20
38.4.5.9 Another Level - Adding ROM Protection . . . 38-22
38.4.5.10 Alternate Bus Master Interface . . . 38-27
38.4.5.10.1 R-AHB Bus Arbitration Signal List . . . 38-27
38.4.5.11 EIM Bus . . . 38-28

38.4.5.12 IP Bus . . . 38-28
38.4.5.13 MDPI Buses . . . 38-28
38.4.5.14 Shared Memory Bus . . . 38-28
38.5 Endian Configuration of ARM7TDMI . . . 38-29
38.5.1 Big Endian Operation . . . 38-29
38.5.2 Little Endian Operation . . . 38-30
38.6 JTAG . . . 38-31
38.6.1 JTAG Clocking and Synchronization . . . 38-31
38.6.2 The ARM7 DBGTRST_B Input . . . 38-31
38.6.3 The ARM7 DBGTDO Output . . . 38-31
38.7 Debug Mode Considerations . . . 38-32
38.7.1 The ARM7 Debug Enable (DBGEN) . . . 38-32
38.7.2 The ARM7 Debug Request (DBGRQ) . . . 38-32
38.7.3 MCU Debug Acknowledge . . . 38-32
38.7.3.1 DBGACK . . . 38-32
38.7.3.2 MCUACK_B . . . 38-32
38.7.4 Breakpoints (DBGBREAK) . . . 38-32
38.7.5 Debug Mode Behavior . . . 38-33
38.7.5.1 Alternate Bus Master Considerations . . . 38-33
38.8 Low Power Modes . . . 38-34
38.8.1 CCM_LPMODE[1:0] . . . 38-34
38.8.2 AP_WAKEUP_B . . . 38-34
38.9 AHBMUX Module . . . 38-35
38.9.1 Alternate Bus Master Interface to R-AHB . . . 38-35
38.9.2 R-AHB Centralized Address Decoding . . . 38-36
38.9.2.1 Internal / External Boot Select . . . 38-36
38.9.2.2 Module Select Outputs (hsel_) . . . 38-36
38.9.3 Special R-AHB Decoding . . . 38-37
38.9.3.1 Holes in the Memory Map (hsel_hole) . . . 38-37
38.9.3.1.1 ROM Holes . . . 38-37
38.9.3.1.2 RAM . . . 38-38
38.9.3.2 HSEL_PROT . . . 38-38
38.9.4 R-AHB State Trackers . . . 38-38
38.9.5 Read Data Muxing . . . 38-39
38.9.5.1 Read Data Over-Rides . . . 38-39
38.9.6 HREADY . . . 38-39
38.9.7 RESP0 (Abort) . . . 38-39
38.9.8 HTRANS[0] . . . 38-39
38.9.9 Reset, IDLE, and BUSY State HREADY Assertion . . . 38-40
38.9.10 AHBMUX_DBGRQ . . . 38-40
38.9.11 AHBMUX_DBGEN . . . 38-41
38.10 Memory Map . . . 38-41
38.11 Scan Wrapper and DFT . . . 38-41
38.12 ARM7 Platform Pin List . . . 38-42
38.13 Electrical Specifications . . . 38-48
38.13.1 External Timing References . . . 38-48
38.13.2 R-AHB Timing . . . 38-49

38.13.3 Alternate Bus Master Interface Timing 38-52
38.13.4 ARM7TDMI-S Internal Timing 38-56
38.13.5 ARM7TDMI-S External Timing 38-58

Chapter 39
ARM7TDMI-S (ARM7)

Chapter 40
ARM7 Platform Core Arbiter (CARB)

40.1 Introduction. 40-2
40.2 Core Arbiter Pin Description 40-3
40.3 External Arbiter Programming Model 40-4
40.4 Arbiter Operation 40-5
40.4.1 Alternate Master Bus Handoff 40-5
40.4.1.1 HCLK 40-6
40.4.1.2 HRESET_B. 40-6
40.4.1.3 HBUSREQ 40-6
40.4.1.4 HGRANT 40-6
40.4.1.5 HRESP0 40-6
40.4.1.6 HREADY 40-6
40.4.1.7 LOCK 40-6
40.4.1.8 HRDATA[31:0] 40-6
40.4.1.9 TRANS[1:0] 40-6
40.4.1.10 AITC_BLK_ARB. 40-6
40.4.1.11 CARB_REG_EN 40-7
40.4.1.12 CARB_ADDR_SEL[1:0] 40-7
40.4.1.13 CARB_DATA_SEL 40-7
40.4.1.14 ALT_RDATA[31:0] 40-7
40.4.1.15 CLKEN 40-8
40.4.1.16 CARB_ABORT 40-8
40.4.1.17 ALT_MASTER[2:0]. 40-8
40.4.1.18 HMASTER[2:0] 40-8
40.4.1.19 CARB_FORCE_NONSEQ 40-9
40.4.2 Low Power Mode Operation. 40-9
40.4.3 Operation during DEBUG Mode 40-9
40.4.4 Differences between AHB Bus Arbitration and CARB 40-9

Chapter 41
R-AHB IP Interface (AIPI)

41.1 Overview. 41-1
41.1.1 Highlights 41-1
41.1.2 General 41-2
41.2 AIPI 1.0 Interface Signals. 41-5
41.2.1 R-AHB Master Interface signals. 41-5
41.2.1.1 Address Bus (haddr[16:0]) 41-5
41.2.1.2 Write Data Bus (hwdata[31:0]). 41-5

41.2.1.3 Read Data Bus (aipi_hrdata[31:0]) . . . 41-5
41.2.1.4 Transfer Type (htrans1) . . . 41-5
41.2.1.5 Transfer Direction (hwrite) . . . 41-6
41.2.1.6 Protection Control (hprot1) . . . 41-6
41.2.1.7 Transfer Size (hsize[1:0]) . . . 41-6
41.2.1.8 AIPI Transfer Done (aipi_hready_out) . . . 41-6
41.2.1.9 Transfer Done In (hready_in) . . . 41-6
41.2.1.10 Transfer Response (aipi_hresp0) . . . 41-6
41.2.1.11 CCM Low Power Mode(ccm_lpmode[1:0]) . . . 41-6
41.2.1.12 Debug Mode (dbgack) . . . 41-7
41.2.2 IPbus v2.0 Interface Signals . . . 41-7
41.2.2.1 IP Address Bus (ips_addr[11:0]) . . . 41-7
41.2.2.2 IP Read Data Bus (ips_rdata[31:0]) . . . 41-7
41.2.2.3 IP write data bus . . . 41-7
41.2.2.4 ips_module_en[31:1] . . . 41-7
41.2.2.5 ips_rwb . . . 41-8
41.2.2.6 ips_byte_31_24, ips_byte_23_16, ips_byte_15_8, ips_byte_7_0 . . . 41-8
41.2.2.7 ips_xfr_wait . . . 41-8
41.2.2.8 ips_xfr_err . . . 41-8
41.2.2.9 ips_supervisor_access . . . 41-8
41.2.2.10 ipg_debug . . . 41-8
41.2.2.11 ipg_stop . . . 41-8
41.2.2.12 ipg_wait . . . 41-9
41.2.2.13 ipg_doze . . . 41-9
41.2.2.14 ipg_hard_async_reset_b . . . 41-9
41.2.2.15 ipg_clk/ips_clk . . . 41-9
41.3 AIPI Pin List . . . 41-10
41.4 AIPI Registers . . . 41-14
41.4.1 Peripheral Size Registers[1:0] . . . 41-14
41.4.2 Peripheral Access Register . . . 41-15
41.5 Interface Timings . . . 41-15
41.5.1 Read Cycles . . . 41-15
41.5.2 Write Cycles . . . 41-16
41.5.3 Aborted Cycles . . . 41-16
41.6 AIPI AC Timing Information . . . 41-29

Chapter 42
Alternate Master Arbiter (AMARB)

42.1 Introduction . . . 42-2
42.2 AMARB Module Pin List . . . 42-4
42.3 AMARB Programming Model . . . 42-7
42.3.1 Register Summary . . . 42-7
42.3.2 Detailed Register Descriptions . . . 42-8
42.4 Arbiter Operation . . . 42-16
42.4.1 Alternate Master Bus Handoff . . . 42-16
42.4.2 Low Power Mode Operation . . . 42-17
42.4.3 Bus Handoff Interval Control . . . 42-17

42.4.4 Bus Control Monitoring 42-25
42.5 Arbitration Protocol 42-25

Chapter 43
ARM7TDMI-S Interrupt Controller (AITC)

43.1 Introduction. 43-2
43.2 AITC Module Pin List 43-5
43.3 Interrupt Controller Programming Model. 43-8
43.3.1 Register Summary. 43-8
43.3.2 Detailed Register Descriptions 43-9
43.4 ARM7TDMI-S Interrupt Controller Operation 43-32
43.4.1 ARM7TDMI-S Prioritization of Exception Sources 43-32
43.4.2 AITC Prioritization of Interrupt Sources 43-32
43.4.3 Assigning and Enabling Interrupt Sources 43-32
43.4.4 Enabling Interrupts Sources 43-33
43.4.5 Controlling Bus Arbitration with AITC 43-33
43.4.6 Typical Interrupt Entry Sequences 43-34
43.4.7 Writing Reentrant Normal Interrupt Routines 43-35

Chapter 44
ARM7TDMI-S Watch Point Module (AWPT)

44.1 Introduction. 44-2
44.1.1 Opcode Patching 44-3
44.1.1.1 Software Controlled Opcode Patching Mode 44-3
44.1.1.2 Program Counter (PC) Relative Opcode Patching Mode 44-3
44.1.1.3 Stack Pointer (SP) Relative Opcode Patching Mode 44-4
44.1.2 Data Fixing 44-5
44.1.3 ARM7TDMI-S Break Pointing 44-5
44.2 AWPT Signals 44-5
44.3 Watch Point Module Programming Model. 44-8
44.3.1 Register Summary. 44-8
44.3.2 Detailed Register Descriptions 44-9
44.4 ARM7TDMI-S Watch Point Module Operation 44-19
44.4.1 Typical AWPT Opcode Patching 44-19
44.4.1.1 Typical Entry to Opcode Patching Routine 44-19
44.4.1.2 Typical Exit from Opcode Patching Routine 44-19
44.4.2 AWPT Event Priority 44-20
44.4.3 Typical Software Context Switching and AWPT. 44-20
44.4.4 Alternate Masters and AWPT. 44-20
44.4.4.1 Low Power Mode and AWPT 44-20
44.4.5 Normal Memory ABORTs and AWPT 44-21

Chapter 45
MCU Memories (MCUMEM)

45.1 Introduction. 45-1

45.2 Memory Controller (MCTL) . . . 45-3
45.2.1 AHB/Memory Interface . . . 45-3
45.2.2 Block Diagram . . . 45-4
45.2.3 MCUMEM Module Pin List . . . 45-5
45.2.4 Wave Diagram . . . 45-9
45.3 MCTL Memory mapped registers . . . 45-10
45.3.1 Register Descriptions . . . 45-10
45.3.1.1 MCTL fuse sense 512K/256K RAM (MFSLRAM) . . . 45-10
45.3.1.2 Reserved Register . . . 45-10
45.3.1.3 MCTL fuse test 512K/256K RAM (MFTLRAM) . . . 45-10
45.3.1.4 Reserved Register . . . 45-11
45.3.1.5 MCTL test mode (MCTLTM) . . . 45-11
45.3.1.6 MCTL Precharge MCTPHG . . . 45-11
45.3.1.7 MCTL Write Val setting 512K RAM (WVALLRM) . . . 45-12
45.3.1.8 Reserved Register . . . 45-12
45.3.1.9 MCTL Read Val setting 512K/256K RAM (RVALLRM) . . . 45-12
45.3.1.10 Reserved Register . . . 45-12
45.3.1.11 MCTL Read Val setting1 1792K ROM (RVALROM) . . . 45-13
45.3.1.12 MCTL Read Val setting2 1792K RAM (RVAL2ROM) . . . 45-13
45.3.1.13 Reserved Register . . . 45-13
45.3.1.14 Reserved Register . . . 45-13
45.3.1.15 Reserved Register . . . 45-14
45.3.1.16 Reserved Register . . . 45-14
45.3.1.17 MCTL Fuse sense2 512K/256K RAM (MFSLRAM1) . . . 45-14
45.3.1.18 MCTL Fuse sense2 512K/256K RAM (MFSLRAM2) . . . 45-14
45.3.1.19 MCTL Fuse sense3 512K/256K RAM (MFSLRAM3) . . . 45-15
45.3.1.20 MCTL Fuse sense4 512K/256K RAM (MFSLRAM4) . . . 45-15
45.3.1.21 MCTL tscan en 512K/256K RAM (MTSCLRAM) . . . 45-15
45.3.1.22 Memory Map . . . 45-16
45.4 MCU RAM (MRAM) . . . 45-18
45.4.1 Block Diagram . . . 45-18
45.4.2 Pin Description . . . 45-19
45.4.3 Wave Diagram . . . 45-20
45.4.4 AC Operating Conditions and Chacteristics . . . 45-20
45.5 MCU ROM (MROM) . . . 45-22
45.5.1 Block Diagram . . . 45-22
45.5.2 Pin Description . . . 45-23
45.5.3 Wave Diagram . . . 45-24
45.5.4 AC Operating Conditions and Chacteristics . . . 45-24

Chapter 46
IP-Bus Mux (IPMUX)

46.1 Introduction . . . 46-2
46.2 IPMux Module Pin List . . . 46-3
46.3 IP-Bus Mux Operation . . . 46-10

Chapter 47
IC Identification Module (IIM)

47.1 Introduction . . . 47-1
47.1.1 IIM Module Pin List . . . 47-2
47.2 IIM Module Block Diagram . . . 47-2
47.3 Register Summary . . . 47-4
47.3.1 Detailed Register Descriptions . . . 47-4

Chapter 48
ARM External Interface Module (AEIM)

48.1 Overview . . . 48-1
48.2 AEIM Module Pin List . . . 48-2
48.3 Typical AEIM System Connections . . . 48-4
48.4 AEIM Functionality . . . 48-7
48.4.1 Chip Select Outputs . . . 48-7
48.4.1.1 CS0 - Chip Select 0 . . . 48-7
48.4.1.2 CS1 - CS4, CS5- Chip Select 1 - Chip Select 5 . . . 48-7
48.4.1.3 Programmable Output Generation . . . 48-8
48.4.2 Programmable Transfer Duration . . . 48-8
48.4.3 Protected Addressing . . . 48-8
48.4.4 Configurable Bus Sizing . . . 48-9
48.4.5 External Boot ROM Control . . . 48-10
48.4.6 Enable Byte Outputs . . . 48-11
48.4.6.1 Enable Byte Control . . . 48-11
48.4.7 Output Enable Outputs . . . 48-11
48.4.8 Burst (Synchronous) Mode . . . 48-11
48.4.8.1 Burst (Synchronous) Mode Operation . . . 48-12
48.4.8.2 Pulse Mode Operation . . . 48-12
48.4.8.3 Quasi-Pipelined Mode . . . 48-13
48.4.9 Bus Watchdog Operation . . . 48-14
48.4.10 Error Conditions . . . 48-14
48.4.11 Show Cycles . . . 48-15
48.5 Real Time Trace Support . . . 48-15
48.6 AEIM Programming Model . . . 48-18
48.6.1 Register Summary . . . 48-19
48.6.2 Quick Reference To CS Control Bit Fields . . . 48-21
48.6.3 Detailed Register Descriptions . . . 48-23
48.7 External Bus Timing Diagrams . . . 48-41

Chapter 49
Multiple Queue Serial Peripheral Interface (MQSPI)

49.1 Introduction . . . 49-2
49.1.1 Basic Features . . . 49-2
49.1.2 Extended Features . . . 49-2
49.1.2.1 Dual Independently Functioning SPIs . . . 49-3

49.1.2.2 32 Configurable Control Queues . . . 49-3
49.1.2.3 64 Programmable Control Data Registers (32 Mode and 32 Pointer Register) 49-3
49.1.2.4 Four FIFO Queues - a High and Low Priority Queue for Each SPI . . . 49-3
49.1.2.5 32 One-Cold Trigger Signals From the Layer 1 Timer . . . 49-3
49.1.2.6 MCU Controlled Trigger Register . . . 49-3
49.1.2.7 Two Interrupt Lines . . . 49-3
49.1.2.8 Programmable Data Change/Latch Phase . . . 49-4
49.2 PORT INFORMATION . . . 49-4
49.2.1 MQSPI Module Pin List . . . 49-5
49.3 MQSPI ARCHITECTURE . . . 49-6
49.3.1 MCU Peripheral Bus Interface . . . 49-6
49.3.2 SPI1, SPI2 . . . 49-8
49.3.3 Chip Select (CS) . . . 49-8
49.3.4 Clock Gen . . . 49-8
49.3.4.1 Transfer Operation . . . 49-8
49.3.4.2 SPI Transfer Setup . . . 49-9
49.3.4.3 SPI Transmit (Write) Operation . . . 49-13
49.3.4.4 SPI Receive (Read) Operation . . . 49-13
49.3.4.5 SPI Receive Operation Timing Considerations . . . 49-13
49.3.5 SPI Control . . . 49-17
49.3.6 MQSPI Registers . . . 49-17
49.3.7 FIFO Queues . . . 49-17
49.3.8 Trig Route . . . 49-17
49.3.9 RAM Arbitration . . . 49-17
49.3.10 Burst and Multiple Message Modes . . . 49-18
49.3.10.1 Multiple Messages . . . 49-18
49.3.10.2 Burst Transfers . . . 49-18
49.3.11 Serial Display Function . . . 49-19
49.3.12 Register Summary . . . 49-21
49.3.13 Detailed Register Descriptions . . . 49-25
49.3.14 RAM . . . 49-26
49.3.14.1 I/O Data Storage RAM . . . 49-29
49.3.14.2 Control Data RAM . . . 49-29
49.3.14.3 Register Descriptions . . . 49-33
49.3.14.4 Interrupt Handling . . . 49-97

Chapter 50
Layer 1 Timer (L1T)

50.1 Port Definitions . . . 50-6
50.1.1 L1T Module Pin List . . . 50-7
50.2 MCU Memory Map . . . 50-10
50.3 Timing Block . . . 50-14
50.3.1 Prescaler (TB_PRE) . . . 50-15
50.3.2 Reference Quarter Bit Counter (RQBC) . . . 50-15
50.3.3 Modulo Adder and Rollover Detector Block . . . 50-16
50.3.4 Channel Frame Counter (CFC) . . . 50-17
50.3.5 Reference Frame Counter (RFC) . . . 50-17

50.3.6 Reference Absolute Frame Number (RAFN) . . . 50-18
50.3.7 Channel Frame Number (CFN) . . . 50-18
50.3.8 Periodic Interrupt Timer (PIT) . . . 50-19
50.3.9 System Clock Divider . . . 50-20
50.3.10 Debug Mode . . . 50-20
50.4 Event Generator . . . 50-20
50.4.1 Event Table . . . 50-20
50.4.2 Frame Tables . . . 50-20
50.4.3 Macro Tables . . . 50-24
50.4.4 Parameter Table . . . 50-26
50.4.5 Frame Table Control Unit . . . 50-26
50.4.6 Macro Table Control Units . . . 50-27
50.4.7 Event Decoder . . . 50-27
50.4.8 Address Generation Unit (AGU) . . . 50-27
50.4.9 QBCO Handler . . . 50-28
50.5 Pin Control Unit . . . 50-30
50.6 Error Detector . . . 50-31
50.7 MCU Interrupt Generation . . . 50-35
50.8 DSP Interrupt Handler . . . 50-36
50.9 Bus Interfaces . . . 50-37
50.9.1 Invalid Addresses . . . 50-38
50.10 Reset, Sleep, and Wakeup Logic . . . 50-38
50.10.1 Reset . . . 50-38
50.10.2 Wakeup . . . 50-39
50.11 Synchronized Wakeup . . . 50-40
50.12 Programmable Timeout . . . 50-40
50.12.1 Sleep . . . 50-40
50.13 Event Codes . . . 50-41
50.14 MCU Memory Map Definitions . . . 50-49
50.14.1 Frame Table RAM . . . 50-49
50.14.2 Macro Table RAM . . . 50-52
50.14.3 Parameter Table RAM . . . 50-54
50.14.4 Register Summary . . . 50-57
50.14.5 Detailed Register Descriptions . . . 50-60
50.15 DSP Memory Map . . . 50-131
50.15.1 Detailed Register Descriptions . . . 50-131
50.16 Appendix . . . 50-138
50.16.1 Example Traffic Channel Timing . . . 50-138
50.16.2 Example Traffic Channel Frame/Macro Tables . . . 50-141
50.16.2.1 Frame Timing Overview . . . 50-142
50.16.2.2 Downlink Frame Timing . . . 50-143
50.16.3 Uplink Frame Timing . . . 50-144
50.16.4 Power Measurement Frame Timing . . . 50-144
50.16.4.1 Frame Table 0 (Odd Frames) . . . 50-145
50.16.4.2 Frame Table 1 (Even Frames) . . . 50-145
50.16.4.3 TCH Receive Macro Table (Macro Table A) . . . 50-145
50.16.4.4 TCH Transmit Macro Table (Macro Table B) . . . 50-146

50.16.4.5 TCH Power Measurement Macro Table (Macro Table C) 50-147
50.16.4.6 Delay Registers 50-149
50.16.4.7 Loop Registers 50-149
50.16.4.8 Parameter Table 50-149
50.17 References 50-150
50.18 Glossary 50-151
50.19 RTL Design Hierarchy 50-152
50.20 Clock Gating 50-155

Chapter 51
Smartcard Interface Module (SIM)

51.1 Errata History 51-3
51.1.1 Patriot RAM1 Errata 51-3
51.1.2 Patriot ROM1 (CDR3) Errata 51-5
51.1.3 Neptune LT (LCA) Errata 51-5
51.2 Enhancements 51-5
51.2.1 New Features 51-5
51.2.1.1 Patriot RAM1 51-5
51.2.1.2 Patriot ROM1 51-6
51.2.1.3 Patriot Hip6W 51-7
51.2.1.4 Patriot Indy 51-7
51.2.2 Neptune LTE 51-7
51.2.3 Hardware Bug Fixes 51-8
51.2.4 SIM Performance Improvements 51-8
51.2.5 Register Interface Change History 51-9
51.2.5.1 Port Control Registers (PORTx_CNTL) 51-9
51.2.5.2 Setup Register (SETUP) 51-9
51.2.5.3 Control Register (CNTL) 51-9
51.2.5.4 Receive Threshold Register (RCV_THRESHOLD) 51-10
51.2.5.5 Transmit Status Register (XMT_STATUS) 51-10
51.2.5.6 Receive Status Register (RCV_STATUS) 51-11
51.2.5.7 Interrupt Mask Register (INT_MASK) 51-11
51.2.5.8 Transmit Threshold Register (XMT_THRESHOLD) 51-12
51.2.5.9 Guard Control Register (GUARD_CNTL) 51-12
51.2.5.10 Reset Control Register (RESET_CNTL) 51-12
51.2.5.11 Character Wait Counter Register (CHAR_WAIT) 51-12
51.2.5.12 General Purpose Counter Register (GPCNT) 51-12
51.2.5.13 Divisor Register (DIVISOR) 51-12
51.2.5.14 Receive Buffer Registers (PORTx_RCV_BUF) 51-13
51.2.5.15 Block Wait Time Registers (BWT_xSB) 51-13
51.2.5.16 (Accurate) Character Wait Time Register (CHAR_WAIT_ACC) 51-13
51.2.5.17 Block Guard Time Register (BGT) 51-13
51.3 Module Overview 51-13
51.3.1 SIM Bus Interface Overview 51-14
51.3.2 SIM Clock Generator Overview 51-15
51.3.3 SIM Transmitter Overview 51-15
51.3.4 SIM Receiver Overview 51-15

51.3.5 SIM Port Control Overview . . . 51-16
51.3.6 SIM General Purpose Counter Overview . . . 51-16
51.3.7 SIM LRC Block Overview . . . 51-17
51.3.8 SIM CRC Block Overview . . . 51-17
51.3.9 SIM ETU counter Block Overview . . . 51-17
51.4 Module Interface . . . 51-17
51.4.1 Port Description . . . 51-19
51.5 Module Architecture . . . 51-20
51.6 Functional Description . . . 51-23
51.6.1 IPBUS Gasket () . . . 51-23
51.6.2 SIM Bus Interface . . . 51-23
51.6.3 SIM Clock Generator . . . 51-25
51.6.3.1 Clock Tree Synthesis . . . 51-26
51.6.3.2 SCAN Test . . . 51-26
51.6.3.3 Baud Clock Generation . . . 51-26
51.6.3.4 Transmitter Clock Generation . . . 51-27
51.6.3.5 Receiver Clock Generation . . . 51-27
51.6.3.6 Port Control Clock Generation . . . 51-27
51.6.3.7 Low Power Mode Clock Control . . . 51-27
51.6.4 SIM Transmitter . . . 51-27
51.6.4.1 Transmit State Machine . . . 51-28
51.6.4.2 Transmit Shift Register . . . 51-29
51.6.4.3 Transmit FIFO . . . 51-29
51.6.4.4 Transmit Guard Time Generator . . . 51-30
51.6.4.5 Transmit NACK Generator . . . 51-30
51.6.4.6 Transmit Data Convention Logic . . . 51-31
51.6.5 SIM Receiver . . . 51-32
51.6.5.1 Receive State Machine . . . 51-32
51.6.5.1.1 Data Sampling/Voting . . . 51-34
51.6.5.1.2 Start Bit Detection . . . 51-34
51.6.5.1.3 Parity Error Detection . . . 51-34
51.6.5.1.4 Framing Error Detection . . . 51-34
51.6.5.1.5 NACK Detection . . . 51-35
51.6.5.1.6 Initial Character Detection . . . 51-35
51.6.5.2 Receive FIFO . . . 51-36
51.6.5.2.1 Overrun Detection . . . 51-36
51.6.6 SIM Port Control . . . 51-37
51.6.6.1 SIM Card Interface . . . 51-37
51.6.6.2 SIM Card Presence Detect . . . 51-37
51.6.6.3 SIM Card Automatic Power Down . . . 51-38
51.6.6.4 SIM Card Forced Power Down . . . 51-38
51.6.7 SIM General Purpose Counter . . . 51-38
51.6.8 SIM LRC Block . . . 51-39
51.6.9 SIM CRC Block . . . 51-39
51.6.10 SIM T=0 & T=1 features . . . 51-41
51.6.10.1 Work Waiting Time / Character Wait Time . . . 51-41
51.6.10.2 Block Time . . . 51-41

51.6.10.2.1 Block Guard Time (T=1 SIM cards only) 51-41
51.6.10.2.2 Block Wait Time 51-42
51.6.10.3 Register Interface 51-42
51.6.10.4 Backward Software Compatibility 51-42
51.6.10.5 ETU Counter Architecture 51-43
51.6.10.5.1 ETU counter State machine 51-43
51.7 Module Interrupts 51-44
51.8 Register Map 51-46
51.8.1 Detailed Register Descriptions 51-47
51.9 Programming Model 51-81
51.9.1 Configuring the SIM for Operation 51-81
51.9.1.1 Configuring the SIM Receiver 51-82
51.9.1.2 Configuring the SIM Transmitter 51-83
51.9.1.3 Configuring the SIM General Purpose Counter 51-83
51.9.1.4 Configuring the SIM Linear Redundancy Check (LRC) Block 51-84
51.9.1.5 Configuring the SIM Cyclic Redundancy Check (CRC) Block 51-84
51.9.2 Using the SIM Receiver 51-84
51.9.2.1 Receive Parity Errors and Parity NACK Generation 51-85
51.9.2.2 Receive Frame Errors 51-86
51.9.2.3 Receive Overrun Errors and Overrun NACK Generation 51-86
51.9.2.4 Using Initial Character Mode and Resulting Receive Data Formats 51-86
51.9.2.4.1 Initial Character Mode Programming Notes 51-87
51.9.2.5 Automatic Receiver Mode 51-87
51.9.2.6 Using the SIM Receiver with “T=1” SIM Cards 51-87
51.9.3 Using the SIM Transmitter 51-88
51.9.3.1 Transmit Data Formats 51-89
51.9.3.2 Transmit NACKs 51-89
51.9.3.3 Transmit Guard Time 51-89
51.9.3.4 Using the SIM Transmit with “T=1” SIM Cards 51-90
51.9.3.5 Suggested “T=1” Compliant Programming Model 51-90
51.9.3.5.1 Answer To Reset (ATR) Detection 51-90
51.9.3.5.2 Programming Considerations for Geldkate Cards 51-92
51.9.3.5.3 Programming Considerations for T=0 SIM Cards 51-93
51.9.3.5.4 Programming Considerations for T=1 SIM Cards 51-94

Chapter 52
Deep Sleep Module (DSM)

52.1 Radio Platform Architecture 52-2
52.2 Interaction with other MCU Peripherals 52-4
52.2.1 Clocks 52-4
52.2.1.1 Clock Cleaner 52-4
52.2.1.2 Gearshifting 52-4
52.2.1.3 Simultaneous use of PAT_REF and CKIL 52-4
52.2.2 Interrupt Handler 52-4
52.2.3 Layer 1 Timer 52-4
52.3 Port Information 52-5
52.4 Deep Sleep Module Architecture 52-8

52.4.1 Bus Interface . . . 52-8
52.4.2 Counter Block . . . 52-10
52.4.3 State Machine . . . 52-12
52.4.3.1 Port Definitions . . . 52-12
52.4.3.2 State Summary . . . 52-12
52.5 Sleep Sequence . . . 52-17
52.5.1 Drift Measurement . . . 52-17
52.5.2 Setup for Sleep . . . 52-17
52.5.3 Commit to Deep Sleep . . . 52-18
52.5.4 Enter Deep Sleep and Gearshift . . . 52-18
52.5.5 Layer 1 Timer Re-calibration . . . 52-20
52.6 Wakeup Sequence . . . 52-21
52.6.1 Response to External Interrupts . . . 52-21
52.6.2 Wakeup via Counter Timeout . . . 52-23
52.6.3 Crystal Warm Up Complete . . . 52-23
52.6.4 Initialize VCO . . . 52-24
52.6.5 Reference DPLL Lock . . . 52-24
52.6.6 Layer 1 Timer Restart . . . 52-24
52.7 Memory Map . . . 52-25
52.8 Detailed Register Descriptions . . . 52-27
52.8.1 Preprogrammed Absolute-Time Parameters . . . 52-31
52.8.2 Derived Absolute-time Parameters . . . 52-35
52.8.3 Control and Status Registers . . . 52-37
52.8.4 Delta-time Parameters . . . 52-49
52.8.5 Word Counter Contents . . . 52-51
52.8.6 MAGIC Minimum Standby time bugfix . . . 52-53
52.9 Appendix . . . 52-55
52.9.1 Timing Relationships . . . 52-55
52.9.2 Crystal Requirements . . . 52-55
52.9.3 Measurement Time Tables . . . 52-56
52.9.4 MATLAB Code for Calculating Measurement Time Tables . . . 52-57

Chapter 53
Watchdog Timer Module (WDOG)

53.1 General Overview . . . 53-1
53.1.1 WDOG Module Pin List . . . 53-1
53.2 Watchdog Timer Operation . . . 53-3
53.2.1 Timing Specifications . . . 53-3
53.2.2 Watchdog During Reset . . . 53-3
53.2.3 Watchdog After Reset . . . 53-3
53.2.3.1 Initial Load . . . 53-3
53.2.3.2 Countdown . . . 53-3
53.2.3.3 Reload . . . 53-3
53.2.3.4 Time-out . . . 53-4
53.2.4 Low Power and DBUG Modes . . . 53-4
53.2.4.1 WAIT and DOZE and STOP Mode . . . 53-4
53.2.4.2 DBUG Mode . . . 53-4

53.2.5 State Machine . . . 53-6
53.3 Watchdog Reset Control . . . 53-7
53.3.1 Reset Sources . . . 53-7
53.3.2 WDOG_B Operation . . . 53-7
53.4 Clock Monitor . . . 53-8
53.5 Watchdog Programming Model . . . 53-9
53.5.1 Detailed Register Descriptions . . . 53-10

Chapter 54
GPRS Encryption Module (GEM)

54.1 Module Overview . . . 54-2
54.2 GEM Functional Description . . . 54-3
54.2.1 GPRS Processing Functional Description . . . 54-3
54.2.1.1 Frame Check Sequence (FCS) Generation/Verification Process . . . 54-3
54.2.1.2 GPRS Encryption/Decryption Process . . . 54-4
54.2.1.3 Flow of GRPS Processing . . . 54-5
54.2.1.4 Calculating GPRS Processing Time\Percentage Bus Cycles Stolen . . . 54-8
54.2.2 Using the GEM for Simple DMA Operations . . . 54-9
54.2.2.1 Calculating DMA Processing Time . . . 54-9
54.2.3 Aborting, Resetting, and Stopping GEM Processing . . . 54-9
54.2.4 GEM Clock Usage and Power Management . . . 54-10
54.3 GEM Data Interface . . . 54-10
54.3.1 Controlling the GEM Data Interface . . . 54-10
54.3.2 Parameters for GPRS LLC Frame Location in RAM . . . 54-10
54.3.3 Miscellaneous Data Interface Controls . . . 54-12
54.3.3.1 Controlling the Size of Data Accesses . . . 54-12
54.3.3.2 Controlling the Spacing of Data Accesses . . . 54-13
54.3.3.3 Having Separate Read and Write Pointers . . . 54-13
54.3.3.4 Automatic Linked List Processing . . . 54-14
54.4 Summary of Changes between the Neptune and Neptune LTS Parts . . . 54-15
54.4.1 Additional processing modes added . . . 54-15
54.4.2 Additional data accessing features added . . . 54-15
54.4.3 Modifications Made Going from Patriot to Neptune . . . 54-16
54.5 Programmer's Model . . . 54-16
54.5.1 Programming for GPRS frame processing . . . 54-16
54.5.2 Programming for Simple DMA Mode Processing . . . 54-17
54.6 Register Map . . . 54-17
54.6.1 Detailed Register Descriptions . . . 54-19
54.6.2 Reference Documents . . . 54-33

Chapter 55
Enhanced General Purpose Timers (EGPT)

55.1 OVERVIEW . . . 55-1
55.2 EGPT ARCHITECTURE . . . 55-2
55.3 GPT AND PWM OPERATION . . . 55-6
55.4 EPIT Operation . . . 55-7

55.4.1 EPIT as a “Set-and-Forget” Timer . . . 55-8
55.4.2 EPIT as a “Free-Running” Timer . . . 55-8
55.4.3 EPIT in MCU LOW POWER and DEBUG MODES . . . 55-9
55.4.3.1 EPIT in Low Power Modes . . . 55-9
55.4.3.2 EPIT in DEBUG Mode . . . 55-9
55.5 EGPT Programming Model . . . 55-10
55.5.1 EGPT Register Summary . . . 55-10
55.5.2 Detailed Register Descriptions . . . 55-11
55.6 Pinout . . . 55-33
55.7 Pin Diagram . . . 55-34

Chapter 56
Real Time Reference (RTR)

56.1 RTR Overview . . . 56-2
56.1.1 RTR Module Pin List . . . 56-3
56.2 Functional Description . . . 56-4
56.3 Module Architecture . . . 56-5
56.4 Module Interrupts . . . 56-7
56.5 RTR Clocking and power management . . . 56-7
56.6 Register Summary . . . 56-7
56.6.1 Detailed Register Descriptions . . . 56-8

Chapter 57
Real Time Clock (RTC)

57.1 Introduction . . . 57-2
57.2 RTC Pin Description . . . 57-3
57.2.1 Input Pins . . . 57-3
57.2.2 Output Pins . . . 57-3
57.2.3 IP Bus . . . 57-3
57.3 RTC Programming Model . . . 57-5
57.3.1 Register Summary . . . 57-5
57.3.2 Detailed Register Descriptions . . . 57-5
57.4 RTC Operation . . . 57-11
57.4.1 RTC Prescaler . . . 57-11
57.4.2 RTC Counters . . . 57-11
57.4.3 RTC Alarm & Interrupt Generation . . . 57-11
57.5 Power Cut Timer . . . 57-12
57.5.1 PCRAM Backup Register Bank . . . 57-12
57.5.2 Signal Interface . . . 57-13
57.5.3 Real Time Clock Trim Calculations . . . 57-13
57.5.3.1 Obtainable Accuracy of Trim . . . 57-13
57.5.3.2 Obtaining the Trim Value . . . 57-13

Chapter 58
Keypad Port (KPP)

58.1 Introduction . . . 58-2
58.2 Keypad Port Pin Description . . . 58-3
58.2.1 Input Pins . . . 58-3
58.2.2 Output Pins . . . 58-3
58.2.3 KPP Module Pin List . . . 58-3
58.2.4 KPP PIN DIAGRAMS . . . 58-5
58.3 Keypad Port Programming Model . . . 58-6
58.3.1 Register Summary . . . 58-6
58.3.2 Detailed Register Descriptions . . . 58-6
58.4 Keypad Operation . . . 58-11
58.4.1 Keypad Matrix Construction . . . 58-11
58.4.2 Keypad Port Configuration . . . 58-11
58.4.3 Keypad Matrix Scanning . . . 58-11
58.4.4 Keypad Standby . . . 58-11
58.4.5 Glitch Suppression on Keypad Inputs . . . 58-12
58.4.6 Multiple Key Closures . . . 58-12
58.4.7 Typical Keypad Configuration and Scanning Sequence . . . 58-13

Chapter 59
Display Memory Access Controller (DMAC)

59.1 Overview . . . 59-1
59.2 DMAC Module Pin List . . . 59-3
59.3 Functional Description . . . 59-4
59.3.1 Accessing the LCD Controller . . . 59-4
59.3.1.1 Automatic DMAC Transfers . . . 59-5
59.3.1.1.1 Automatic Display Data Transfers (AUTOMODE[1:0]=10) . . . 59-5
59.3.1.1.2 Automatic Command Data Transfers (AUTOMODE[1:0]=00) . . . 59-12
59.3.1.1.3 Automatic Command Data Transfers (AUTOMODE[1:0]=01) . . . 59-13
59.3.1.2 Direct IPAccess . . . 59-13
59.3.2 Aborting DMAC Transfers . . . 59-13
59.3.3 Low Power Mode Operation . . . 59-14
59.4 DMAC Register Summary . . . 59-14
59.5 DMAC Register Descriptions . . . 59-15
59.6 LCD Controller Interface . . . 59-33
59.6.1 Serial Interface . . . 59-33
59.6.2 Parallel Interface . . . 59-35
59.7 LCD Clock Configuration . . . 59-37
59.8 R-AHB Interface and DMAC FIFOs . . . 59-37

Chapter 60
Hash Acceleration Module (HAC)

60.1 Module Overview . . . 60-2
60.2 Functional Description . . . 60-2

60.2.1 HAC Usage . . . 60-2
60.2.2 Reading hash result from HAC . . . 60-3
60.2.3 HAC Data Interface . . . 60-3
60.2.3.1 Calculating HAC Processing Time and Percentage Bus Cycles Stolen . . . 60-5
60.2.3.2 Aborting, Resetting, and Stopping HAC Processing . . . 60-5
60.2.4 HAC Clock Usage and Power Management . . . 60-6
60.2.5 HAC disable via LASER fuse . . . 60-6
60.2.6 Running HAC with various burst memory configurations . . . 60-6
60.3 Programmer's Model . . . 60-6
60.4 Register Map . . . 60-7
60.4.1 Detailed Register Descriptions . . . 60-8

Chapter 61
Data RAM Arbiter Module (DRA)

61.1 Introduction . . . 61-2
61.2 DRA Block Diagram . . . 61-2
61.3 DRA Pin List . . . 61-3
61.4 Arbitration Scheme . . . 61-4
61.5 Low Power Mode . . . 61-4

Chapter 62
Memory Separation Unit (MSU)

62.1 Introduction . . . 62-2
62.2 Operation . . . 62-2
62.2.1 Disabling the MSU . . . 62-4
62.2.1.1 Complete and Total Hardware Disable . . . 62-4
62.2.1.2 Disabling Abort Generation . . . 62-4
62.2.2 Enabling the MSU . . . 62-5
62.2.3 The msu_prot_violation Output . . . 62-5
62.2.4 Unassigned Memory Area Behavior . . . 62-5
62.2.5 Privileged Mode Operation (Automatic Power-down) . . . 62-5
62.3 MSU Registers . . . 62-6
62.3.1 Detailed Register Descriptions . . . 62-7
62.3.2 ID Register . . . 62-8
62.3.3 Set Enable Register . . . 62-9
62.3.4 Clear Enable Register . . . 62-10
62.3.5 Fault Address Register . . . 62-11
62.3.6 Fault Status Register . . . 62-12
62.3.7 Area Enable Register . . . 62-14
62.3.8 Area Control Registers . . . 62-15
62.3.8.1 Discussion of MSU Address Compare Logic . . . 62-17
62.4 MSU Caveats . . . 62-18
62.5 Memory Map Example . . . 62-18

Chapter 63
Security Controller (SCC)

63.1 Introduction . . . 63-1
63.2 External Pins . . . 63-2
63.3 Secure RAM . . . 63-7
63.3.1 Secure RAM Overview . . . 63-7
63.3.1.1 Use of the Secure RAM . . . 63-8
63.3.1.2 Use of the Secure RAM in a non-secure application . . . 63-8
63.3.1.3 Security Monitor . . . 63-8
63.3.2 Block Diagram . . . 63-9
63.3.2.1 Memory Controller . . . 63-10
63.3.2.2 Key Encryption Module . . . 63-11
63.3.2.3 Bus Interface . . . 63-11
63.3.3 Internal Signal Pins . . . 63-11
63.3.4 Register Summary . . . 63-14
63.3.5 Detailed Register Descriptions . . . 63-15
63.3.5.1 RED_START . . . 63-16
63.3.5.2 BLACK_START . . . 63-17
63.3.5.3 Length . . . 63-18
63.3.5.4 Control . . . 63-19
63.3.5.5 Status . . . 63-20
63.3.5.6 Error Status . . . 63-22
63.3.5.7 Interrupt Control . . . 63-24
63.3.5.8 Configuration . . . 63-25
63.3.5.9 Initialization Vector . . . 63-26
63.3.5.10 RED_MEM . . . 63-27
63.3.5.11 BLACK_MEM . . . 63-27
63.3.5.12 Initialization Vector . . . 63-27
63.3.6 Functional Details . . . 63-27
63.3.6.1 DMA Controller . . . 63-27
63.3.6.2 Zeroization . . . 63-27
63.3.6.3 Blocking Access to Memory . . . 63-28
63.3.6.4 Laser ID . . . 63-28
63.3.6.5 Error Code for the Laser ID . . . 63-30
63.3.6.6 Generating the Code Bits . . . 63-32
63.3.6.7 Checking the Code Bits . . . 63-33
63.4 Security Monitor . . . 63-34
63.4.1 Monitor Components . . . 63-35
63.4.1.1 Secure State Controller . . . 63-36
63.4.1.2 Security Policy . . . 63-40
63.4.1.3 Algorithm Integrity Checker (AIC) . . . 63-42
63.4.1.3.1 Algorithm Sequence Checker (ASC) . . . 63-43
63.4.1.3.2 Bit Bank . . . 63-44
63.4.1.3.3 Plaintext/Ciphertext Comparator . . . 63-46
63.4.1.3.4 AIC Example . . . 63-47
63.4.1.4 Timer . . . 63-47
63.4.1.5 Debug Detector . . . 63-48

63.4.2 Use in a Non-Secure Application 63-48
63.4.3 Internal Signals Pins 63-49
63.4.4 Register Definitions 63-51
63.4.4.1 Secure State Controller Registers 63-53
63.4.4.1.1 Status (R/W, Supervisor) 63-53
63.4.4.1.2 Command (R/W, Supervisor) 63-55
63.4.4.2 Algorithm Sequence Checker Registers 63-56
63.4.4.2.1 SeqStart (R/W, Supervisor) 63-56
63.4.4.2.2 SeqEnd (R/W, Supervisor) 63-57
63.4.4.2.3 SeqChk (R/W, Supervisor/User) 63-58
63.4.4.3 Bit Bank Registers 63-59
63.4.4.3.1 BitCnt (R, Supervisor) 63-59
63.4.4.3.2 IncSize (R/W, Supervisor) 63-60
63.4.4.3.3 BBDec (W, Supervisor/User) 63-61
63.4.4.4 PT/CT Registers 63-62
63.4.4.4.1 CompSize (R/W, Supervisor) 63-62
63.4.4.4.2 PTChk (R/W, Supervisor/User) 63-63
63.4.4.4.3 CTChk (R/W, Supervisor/User) 63-64
63.4.4.5 Timer Registers 63-65
63.4.4.5.1 TimerIV (R/W, Supervisor) 63-65
63.4.4.5.2 TimerCtl (R/W, Supervisor) 63-66
63.4.4.5.3 Timer (R, Supervisor) 63-67
63.4.4.6 Debug Detector Register 63-68
63.4.4.6.1 DD Status (R/W, Supervisor) 63-68
63.5 DFT and Scan Testing 63-69
63.5.1 Sensitive Information 63-69
63.5.2 Scan Testing 63-69
63.5.3 Scan Test Controller 63-70
63.5.4 Testing the secret key test logic 63-70
63.6 Security Features 63-70

Chapter 64
Security Laser ID Module (SLID)

64.1 Description 64-1
64.1.1 SLID Module Pin List 64-2

Chapter 65
MCU to Dual-Port-Ram Peripheral Interface (MDPI)

65.1 Overview 65-1
65.2 Bus Interface State Machine Block 65-2
65.3 MDPI Multiplexor Block (MMB) 65-6

Chapter 66
One - Wire Interface (OWIRE)

66.1 Introduction . . . 66-1
66.2 Peripheral Architecture . . . 66-2
66.3 Port Definitions . . . 66-3
66.4 Functional Description . . . 66-4
66.4.1 Low Power Modes . . . 66-4
66.4.2 Reset Sequence with Reset Pulse Presence Pulse . . . 66-4
66.4.3 Write 0 . . . 66-5
66.4.4 Write 1/Read Data . . . 66-5
66.4.5 Program Pulse . . . 66-6
66.5 Registers . . . 66-6
66.5.1 Register Summary . . . 66-6
66.5.2 Detailed Register Descriptions . . . 66-7
66.5.2.1 Control Register . . . 66-8
66.5.2.2 Time Divider Register . . . 66-9
66.5.2.3 Reset Register . . . 66-10
66.6 Addendum . . . 66-10
66.7 Reference Documents . . . 66-14

Chapter 67
Test Control Module (TCM)

67.1 Overview . . . 67-3
67.1.1 Features . . . 67-3
67.2 Block Diagram . . . 67-4
67.3 Description . . . 67-6
67.3.1 Alternate Master Test (AMT) . . . 67-6
67.3.2 Bus Mux (BMM) . . . 67-6
67.3.2.1 Bus Mux . . . 67-6
67.3.3 Test Control Register (TCR) . . . 67-6
67.3.4 Miscellaneous Test Control Register (MTCR) . . . 67-7
67.3.5 Test Modes . . . 67-7
67.3.6 MCU BIST Operation . . . 67-9
67.3.6.1 MCU Bist operation using MBIST Register . . . 67-9
67.3.6.2 MCU Bist operation using Test Pins . . . 67-10
67.3.6.3 Example of Both Methods . . . 67-10
67.3.7 DSP BIST Operation . . . 67-10
67.4 Non Scan Tests . . . 67-10
67.4.1 ARM7 Trace Mode . . . 67-11
67.4.2 ARM7 Functional . . . 67-11
67.4.3 DSP56600 Test Mode . . . 67-11
67.5 Register Map . . . 67-12
67.6 Detailed Register Descriptions . . . 67-14
67.6.1 Register Description of Incoming Bist signals . . . 67-24
67.7 TCM Pin List . . . 67-26
67.8 Clocks During Test Modes . . . 67-30

67.8.1 Clocks During AMT Mode . . . 67-30
67.8.2 Clocks During MBIST Mode . . . 67-30
67.9 Alternate Master Test Module (AMT) . . . 67-30
67.9.1 Overview . . . 67-30
67.9.2 Features . . . 67-30
67.9.3 Block Diagrams . . . 67-32
67.9.4 AMT Register Map . . . 67-32
67.9.5 Functional Description . . . 67-33
67.9.5.1 R-AHB Arbitration . . . 67-33
67.9.5.2 AMT Mode . . . 67-33
67.9.5.3 I/O Registers . . . 67-39
67.9.6 External Pin Descriptions . . . 67-39
67.9.7 Modes of Operation . . . 67-39
67.9.8 Interrupt Operation . . . 67-39
67.9.9 AMT Module Pin List . . . 67-40

Chapter 68
Scan Test Controller (STC)

68.1 Module Overview . . . 68-1
68.1.1 General Description . . . 68-2
68.2 Scan Control Unit . . . 68-3
68.2.1 Neptune LTE Test Modes . . . 68-4
68.2.2 Scan Test Modes . . . 68-5
68.2.3 DSP Master Mode . . . 68-5
68.2.4 MBIST Test Modes . . . 68-6
68.2.5 Scan Operations . . . 68-6
68.2.5.1 Scan Test Clock and Test Reset . . . 68-6
68.2.5.2 Chip In Scan Mode . . . 68-6
68.2.5.3 Platform Internal Scan Test . . . 68-6
68.2.5.4 Platform External Scan Test . . . 68-7
68.2.5.5 SOG Scan Test . . . 68-7
68.2.5.6 Scan Divergence Mode and IDDQ Test . . . 68-7
68.2.5.7 Small Embedded Memory Macrotest Modes . . . 68-7
68.2.5.7.1 Small Memory Macrotest mode 0 . . . 68-7
68.2.5.7.2 Small Memory Macrotest mode 1 . . . 68-8
68.3 Scan Chain Router . . . 68-11
68.3.1 Scan Chains from Platforms and SOG to Be Routed . . . 68-11
68.3.2 Scan Input Router . . . 68-12
68.3.3 Scan Output Router . . . 68-13
68.4 Scan Reset Logic Additions . . . 68-21
68.5 STC Module Pin List . . . 68-22

Chapter 69
MCU External Interrupts (INT)

69.1 Block Diagram . . . 69-2
69.2 Ports List . . . 69-2

69.3 Pins Description 69-3
69.4 Programming Model 69-4
69.4.1 Register Summary 69-4
69.4.2 Detailed Register Descriptions 69-5

Chapter 70
Built-In Self Test for RAM and ROM (BIST)

70.1 Introduction 70-2
70.2 BIST Module Pin List 70-3
70.2.1 Input Pins 70-3
70.2.2 Output Pins 70-4
70.3 BIST Operation 70-7
70.3.1 BIST Disabled 70-7
70.3.2 Production Test Mode - Stop on First Fail 70-7
70.3.3 Debug Test Mode - Continue on Fail 70-8
70.3.4 ROM MISR Data Out Mode 70-9
70.3.5 Bitmap Mode 70-10
70.4 RAM Fault Model and Test Algorithm 70-10
70.4.1 RAM Fault Model 70-10
70.4.2 RAM Test Algorithm 70-12
70.4.2.1 Algorithm For Late Write Buffered RAM 70-12
70.4.3 Data Retention 70-13
70.5 ROM Fault Model and Test Algorithm 70-13
70.5.1 ROM Fault Model 70-13
70.5.2 ROM Test Algorithm 70-14
70.6 Integration of Multiple BIST Modules 70-14
70.6.1 Assumptions and Guidelines for BIST Integration 70-14

Chapter 71
System JTAG Controller (SJC)

71.1 Introduction 71-2
71.2 Neptune JTAG OVERVIEW 71-4
71.2.1 PAD SIGNALS 71-4
71.2.1.1 Test Clock (TCK) 71-4
71.2.1.2 Return Test Clock (RTCK) 71-4
71.2.1.3 Test Mode Select (TMS) 71-4
71.2.1.4 Test Data Input (TDI) 71-4
71.2.1.5 Test Data Out (TDO) 71-4
71.2.1.6 Test Reset (TRST) 71-4
71.2.1.7 SJC_MOD 71-4
71.2.2 SJC Pin List 71-5
71.2.3 ARM Synchronization 71-8
71.2.4 SJC Modes 71-9
71.2.4.1 Onyx Mode (SJC_MOD = 0) 71-9
71.2.4.2 Enabling ARM-Onyx JTAG Concatenation 71-10

71.2.4.3 ARM Mode (SJC_MOD = 1) . . . 71-11
71.2.5 DSP JTAG Overview . . . 71-12
71.2.5.1 TAP CONTROLLER . . . 71-13
71.2.5.2 INSTRUCTION REGISTER . . . 71-13
71.2.5.3 IDCODE . . . 71-16
71.2.5.4 HIGHZ . . . 71-16
71.2.5.5 ENABLE_DSP_ONCE . . . 71-16
71.2.5.6 ENABLE_MCU . . . 71-17
71.2.5.7 DSP_DEBUG_REQUEST . . . 71-17
71.2.5.8 LOAD_ABISTCR . . . 71-17
71.2.5.9 LOAD_JTCR . . . 71-17
71.2.5.10 BYPASS . . . 71-18
71.2.6 JTAG Test Control Register (JTCR) . . . 71-19
71.2.7 Debug Mode . . . 71-20
71.3 Examples of JTAG instructions . . . 71-22
71.3.1 Example of Entering ARM-Onyx serial chain via JTAG Control . . . 71-22
71.3.2 Example of Releasing the MCU and DSP from Debug Modes . . . 71-23
71.4 Serial ARM-ONYX TDI-TDO Waveform . . . 71-26
71.5 Neptune JTAG Restrictions . . . 71-27
71.6 Issues . . . 71-27

Chapter 72
Analog Test Module (ANATEST)

72.1 Overview . . . 72-1
72.2 Analog Signal Selection and Routing (Primary Function) . . . 72-2
72.3 Digital Signal Selection and Routing (Secondary Function) . . . 72-3
72.4 Analog Muxes . . . 72-3
72.4.1 Analog Mux1 . . . 72-5
72.4.2 ANALOG Mux2 . . . 72-5
72.5 Digital Muxes . . . 72-6
72.5.1 Functional Mode . . . 72-6
72.5.2 BIST Mode . . . 72-6
72.6 Level Shifter . . . 72-6
72.7 Register Summary . . . 72-7
72.8 Detailed Register Descriptions . . . 72-8
72.9 Pin List . . . 72-12

Chapter 73
Input/Output Pad (PADIO)

73.1 Introduction . . . 73-2
73.2 Functional Requirements . . . 73-2
73.3 PADIO Module Pin List . . . 73-4
73.4 Electrical Requirements . . . 73-5
73.4.1 DC Characteristics . . . 73-6
73.4.2 AC Characteristics . . . 73-7
73.5 Layout Considerations . . . 73-8

73.6 Chip Integration-Level Connections . . . 73-8
73.6.1 Enable Selectors . . . 73-8
73.6.2 Input Type Selector . . . 73-10

Chapter 74
Standard ESD Protection Networks (PADRING)

74.1 Introduction . . . 74-1
74.2 Pad Drive Strength . . . 74-11
74.2.1 User Drive Strength Control . . . 74-11
74.2.2 Test Drive Strength Control . . . 74-11
74.3 Pad Cells . . . 74-11
74.4 Power Supplies . . . 74-12
74.4.1 Power Down Modes . . . 74-13

Chapter 75
Neptune LTE Electrical Requirements (ELEC)

75.1 DC Operating Conditions . . . 75-2
75.2 Power Up Sequence . . . 75-6
75.3 AC Characteristics . . . 75-8
75.3.1 Crystal Oscillator Characteristics . . . 75-8
75.3.2 Reference Digital Phase Locked Loop Characteristics . . . 75-8
75.3.3 DSP Internal Clocks . . . 75-12
75.3.4 Clock Monitor Module Electrical Characteristics . . . 75-13
75.3.5 Reset, Mode Select, and Interrupt Timing . . . 75-14
75.3.6 AEIM and Emulation Port Timing . . . 75-16
75.3.7 Subscriber Interface Module Timings . . . 75-22
75.3.8 MQSPI Timing Specifications . . . 75-23
75.3.9 USB Timing Specifications . . . 75-26
75.3.10 BBP and SAP Timing Specifications . . . 75-28
75.3.11 DMAC Timing Specifications . . . 75-35
75.3.12 JTAG Port . . . 75-36
75.4 Analog Module Electrical Specifications . . . 75-38
75.4.1 Receiver Analog Front-End Electrical Specifications . . . 75-38
75.4.2 Transmitter & Receiver Synthesizer Electrical Specifications . . . 75-38
75.4.3 Transmitter Dual Port D/A Electrical Specifications . . . 75-38
75.4.4 Power Amplifier Control Electrical Specifications . . . 75-38
75.4.5 Voice Coder Electrical Specifications . . . 75-38
75.4.6 General Purpose A/D Converter Electrical Specifications . . . 75-38

Chapter 76
Mechanical Specifications (PKG)

76.1 MAPBGA Packages . . . 76-1
76.2 280 MAPBGA 13X13mm 0.65mm pitch for Production . . . 76-2
76.2.1 280 MAPBGA Package Diagrams (13 x 13 mm) . . . 76-3
76.2.2 280 MAPBGA Production Package Pinout . . . 76-5

76.3 280 MAPBGA 13X13mm 0.65mm pitch for Development. 76-9
76.3.1 280 MAPBGA Development Package Pinout 76-10

Appendix A
Neptune Bootloader (ABOOT)

A.1 Introduction. A-1
A.2 MCU Boot Sequence A-1
A.2.1 Pin Configuration at Reset A-1
A.2.2 MCU Boot Mode Descriptions A-2
A.2.3 PCS MCU Boot Mode A-3
A.2.4 SPS MCU Boot Modes A-3
A.2.4.1 SPS MCU Burnin Test Mode A-3
A.2.4.2 SPS MCU Normal Boot Mode A-3
A.2.4.2.1 Preamble Search A-3
A.2.4.2.2 Destination Address A-4
A.2.4.2.3 Download Size A-4
A.2.4.2.4 Baud Rate Update A-4
A.2.4.2.5 Checksum A-4
A.2.4.3 SPS MCU Rx/Tx External Control Boot Mode A-5
A.3 DSP Boot Sequence A-5
A.3.1 Pin Configuration at Reset A-5
A.3.2 SPS Test Mode Descriptions A-6
A.3.2.1 Burnin Mode A-6
A.3.2.2 Ext_word Mode A-7
A.3.2.3 Typ2 Mode A-7
A.3.2.4 Ext_byte Mode A-7
A.3.2.5 Wait Mode A-7
A.3.2.6 Serial Mode. A-8
A.3.2.7 Stop Mode. A-8
A.3.2.8 DSP Bootloader Protocol B and C Specifications A-8
A.3.2.8.1 Protocol B: Shared Memory Boot A-8
A.3.2.8.2 Protocol C: Messaging Unit Boot. A-9
A.3.3 DSP Bootloader Protocol A Specification A-9
A.3.3.1 Message Formats A-9
A.3.3.1.1 Short Message Format A-10
A.3.3.1.2 Long Message Format A-11
A.3.3.1.3 XYP Fields A-12
A.3.3.2 Message Structure. A-13
A.3.3.2.1 memory_write.request A-13
A.3.3.2.2 memory_write.response A-14
A.3.3.2.3 memory_read.request A-14
A.3.3.2.4 memory_read.response A-15
A.3.3.2.5 memory_check.request A-16
A.3.3.2.6 memory_check.response. A-17
A.3.3.2.7 start_application.request A-17
A.3.3.2.8 use_pll.request A-17
A.3.3.2.9 use_pll.response A-18

A.3.3.2.10 burnin.request . . . A-18
A.3.3.2.11 invalid_opcode.response . . . A-18
A.3.3.3 Protocol A Usage Notes . . . A-19
A.3.3.3.1 Accessing 24 bit Program Memory . . . A-19
A.3.3.3.2 Shared Memory Size . . . A-19
A.3.3.3.3 DSP Message Reception . . . A-19
A.3.3.3.4 Long Reply Message Address . . . A-19
A.4 Complete Pin Allocation Summary . . . A-19

Appendix B
Design for Testabilty Methodology (DFT)

B.1 Introduction . . . B-1
B.2 Chip Level Overview . . . B-2
B.2.1 Design Flow for Module Scan Insertion . . . B-3
B.2.2 General ATPG Flow . . . B-4
B.2.3 Getting SDI and SDO Inputs In and Out of the Chip . . . B-5
B.2.4 Legency Module Naming Convention for Test Signal Ports . . . B-6
B.2.5 Testing Cores in Neptune . . . B-7
B.2.6 Bus Interface for Peripherals . . . B-7
B.2.7 Maximum Scan Chain Length . . . B-8
B.2.8 Clock & Reset Control During Scan Testing . . . B-9
B.2.9 Testability and Test Coverage . . . B-9
B.2.9.1 Bus Contention . . . B-9
B.2.9.2 Unknown State Signals . . . B-10
B.2.9.3 Unobservable Signals . . . B-10
B.2.9.4 Configuring the IO Pins During Test Mode . . . B-10
B.2.10 MCU PIG/MIG Interface . . . B-10
B.2.11 MCU TAN-line Interface to Test Control Module (TCM) . . . B-12
B.2.12 BIST/Memory Test . . . B-12
B.2.13 PIG Test . . . B-13
B.3 Chip Scan Control . . . B-14
B.3.1 Clock Control for Scan Testing of Legacy Modules . . . B-14
B.3.2 New Module Clocking Methodology . . . B-17
B.3.3 Test Register Block . . . B-17
B.3.4 External Clocks/Divided Clocks . . . B-18
B.4 Module Scan Requirements . . . B-19
B.4.1 Module RESET Control for Legacy Modules . . . B-19
B.4.2 Module RESET Control for New Modules . . . B-20
B.5 Removal of the Scan Data Output Tri-State Elements . . . B-20
B.5.1 Module SDO Output Signal(s) . . . B-21
B.5.2 Module Scan Enable . . . B-22
B.5.3 Module Tri-State Enables for Functional Signals . . . B-22
B.5.4 Summary of all the Scan Control in a DSP or MCU Peripheral . . . B-23
B.6 RTL Coding Examples . . . B-26
B.7 Neptune DFT Core Scan Requirements . . . B-28
B.7.1 Core Requirements . . . B-28
B.7.1.1 Software (versions must be synchronized with Neptune requirements) . . . B-28

B.7.1.2 Design Requirements for Integration . . . B-28
B.7.1.3 Test Modes . . . B-28
B.7.1.4 Deliverables . . . B-29
B.7.2 S-OnyxU 56600S Specifics . . . B-30
B.7.2.1 Testing DSP Core in Neptune . . . B-30
B.7.2.2 DSP Synthesizable PMBIF . . . B-30
B.7.3 ARM7TDMI-S . . . B-31
B.8 Analog and Mixed-Signal DFT . . . B-35
B.8.1 Introduction . . . B-35
B.8.2 Analog and Mixed-signal Modules . . . B-36
B.8.3 Analog Modules Requirements . . . B-37
B.8.4 Mixed-Signal Modules Requirements . . . B-37
B.8.5 I/O Pads Requirements . . . B-38
B.9 Introduction . . . B-39
B.9.1 Algorithm . . . B-39
B.9.2 Late Write Memory . . . B-40
B.9.3 ipt_test_membist_sel[1:0] . . . B-41
B.9.4 ipt_test_membist_invoke . . . B-41
B.9.5 ipt_test_membist_reset . . . B-41
B.9.6 ipt_test_membist_fail . . . B-41
B.9.7 ipt_test_membist_done . . . B-42
B.9.8 ipt_test_membist_bitmap_out . . . B-42
B.10 Neptune Testing in Additional to Scan . . . B-42
B.10.1 Functional Test Introduction . . . B-42
B.10.1.1 JTAG . . . B-42
B.10.1.2 Scan Controller (Test Control Module) . . . B-42
B.10.1.3 BIST . . . B-42
B.10.2 Functional Speed and Non-Scanned Logic Verification Vectors . . . B-43
B.10.2.1 Critical Speed Paths . . . B-43
B.10.2.2 Bus Specification . . . B-43
B.10.2.3 Non-Scanned Logic . . . B-43
B.10.3 IDDQ Scan Vectors . . . B-43
B.10.4 Analog Test Vectors . . . B-44
B.11 Failure Analysis . . . B-44
B.11.1 Full Scan . . . B-44
B.11.2 IDDQ . . . B-44
B.11.3 MEMORY BIST with Debugging Capability or at Full Speed Access . . . B-45
B.11.4 Scan Divergence . . . B-45
B.11.4.1 Entering Scan Mode Operation . . . B-45
B.11.4.2 Reconfiguring Scan Chains . . . B-45
B.11.4.3 Functional Clock Speeds Greater Than the Tester Frequency . . . B-46
B.11.4.4 Non-Destructive Memory During Scan Shift . . . B-47
B.11.5 BURNIN Scan Mode . . . B-47
B.11.6 FA Diodes . . . B-47
B.11.7 Functional Test Suite . . . B-47
B.11.8 Micro-Probing . . . B-47
B.11.9 E-BEAM Probing . . . B-47

B.11.10 Test Point Insertion B-47
B.12 Neptune Scan Guidelines and Rules B-47
B.12.1 General Comments about Rules and Guidelines B-47
B.12.2 Clock Rules B-48
B.12.3 RULE: No Gated Clocks Must Exist During Scan Shifting B-48
B.12.4 RULE: Define All Scan Ports B-48
B.12.5 RULE: No Internally Generated Clocks Used During Scan Testing B-49
B.12.6 GUIDELINE: Multiplexers Instead of 3-Statable Elements B-49
B.12.7 RULE: Testing 3 State Elements B-49
B.12.8 RULE: Testing Dynamic Logic B-49
B.12.9 GUIDELINE: Types of Storage Elements B-49
B.12.10 RULE: Combinational Feedback Paths B-50
B.12.11 GUIDELINE: Handling Memory Array Elements During Scan Testing B-50
B.12.12 GUIDELINE: Non-Scannable Analog, Core, and Embedded Memory Test Techniques B-51
B.12.13 GUIDELINE: Scan Enable Network B-51
B.12.14 RULE: SDI (Scan Data-In) Port Hook-Up During Scan Shifting B-51
B.12.15 RULE: Control Asynchronous Cells During Scan Testing B-51
B.12.16 GUIDELINE: No Multi-Cycle Paths During Scan Testing B-52
B.12.17 GUIDELINE: Scan Chain(S) Lengths B-52
B.12.18 GUIDELINE: Scan Chain Clocking B-52
B.12.19 GUIDELINE: Scan In and Scan Out Pad Data Frequency B-52
B.12.20 GUIDELINE: Control of 3 State Bus Enables B-52

Appendix C
MAP BGA Substrate (PKGROUTE)

C.1 Neptune Package Substrate Design Priorities C-1
C.2 Other Considerations C-2

List of Tables

Table 1-1 Neptune LTE DSP Memory Requirements 1-4
Table 1-2 MCU Memory Configurations 1-10
Table 2-1 Neptune LTE Pin Assignments. 2-2
Table 2-2 AEIM Signals 2-29
Table 2-3 Real Time Trace Signals. 2-31
Table 2-4 Layer 1 Timer Signals. 2-32
Table 2-5 A2DIGL Signals 2-33
Table 2-6 External Interrupt Signals 2-34
Table 2-7 SAP Signals 2-36
Table 2-8 GPIO Signals 2-37
Table 2-9 SIM Signals. 2-38
Table 2-10 DMAC Signals 2-39
Table 2-11 KPP Signals 2-40
Table 2-12 RTC Signals 2-41
Table 2-13 CCM Signals. 2-41
Table 2-14 RF Interface Signals 2-42
Table 2-15 MQSPI Signals 2-42
Table 2-16 JTAG Signals 2-44
Table 2-17 Miscellaneous Control Signals 2-46
Table 2-18 USB Signals 2-46
Table 2-19 RX and TX Synthesizer Signals 2-47
Table 2-20 UART Signals. 2-48
Table 2-21 PA Control Signals 2-49
Table 2-22 RX Data Converter Signals. 2-50
Table 2-23 Audio Codec Signals 2-50
Table 2-24 GPADC Signals 2-51
Table 2-25 Power and Ground Signals 2-52
Table 2-26 EGPT Signals 2-53
Table 2-27 DSP Debug and Trace Signals 2-54
Table 2-28 DSP Debug Page Visibility. 2-54
Table 2-29 DSP Triple Timer Signals. 2-55
Table 2-30 One Wire Interface Signals. 2-55
Table 2-31 I/O Drive Groups 2-55
Table 2-32 $\overline{\text{LBA}}$ Drive Strength Settings 2-57

Table 2-33 Power Supply Description 2-58
Table 2-34 Port A Mux Assignments 2-60
Table 2-35 Port B Mux Assignments 2-61
Table 2-36 Port C Mux Assignments 2-62
Table 2-37 Port D Mux Assignments 2-63
Table 2-38 Port E Mux Assignments 2-64
Table 3-1 MCU Peripheral Interfaces and Clocks 3-5
Table 3-2 PCTL1 Description 3-8
Table 3-3 DSP Module Clock Sources 3-10
Table 5-1 CCM Module Pin List 5-4
Table 5-2 CCM Register Summary 5-9
Table 5-3 DPLL_CTL Description 5-11
Table 5-4 DPLL_OP Description 5-13
Table 5-5 DPLL_MFN Description 5-14
Table 5-6 DPLL_MFD Description 5-15
Table 5-7 DFR Description 5-16
Table 5-8 CSR Description 5-17
Table 5-9 CMON Description 5-19
Table 5-10 LPMDR Description 5-21
Table 5-11 DPLL_MFN MINUS Description 5-22
Table 5-12 MFN PLUS register 5-23
Table 5-13 MFN TOG CNT Register 5-24
Table 5-14 DESENSE STAT Register 5-25
Table 5-15 Characteristics of CCM Clocks During System Events 5-32
Table 5-16 Reset Assertion Timings 5-38
Table 5-17 Reset Deassertion Timings 5-39
Table 6-1 DSP Physical Memory Configurations 6-3
Table 6-2 MCU Memory Configurations 6-4
Table 6-3 DSP X - I/O and Y - I/O Memory Map 6-4
Table 6-4 Patch Jump Targets 6-9
Table 6-5 DSP 56600 OMR Description (Continued) 6-12
Table 6-6 DSP 56600 IPRP Description 6-15
Table 6-7 DSP 56600 IPRC Description 6-18
Table 6-8 IDR Description 6-20
Table 6-9 JIDR Description 6-21
Table 6-10 DSP Interrupt Sources 6-22
Table 6-11 **Second Level Priority Ordering for DSP Interrupt Sources 6-25**
Table 6-12 DSP Interrupt Sources to the DSP56600 Core 6-28
Table 6-13 MCU Memory Space (MOD = HIGH on reset) 6-34

Table 6-14 MCU Interrupt Sources 6-63
Table 6-15 Pad Drive Groups 6-72
Table 6-16 LBA Drive Strength 6-73
Table 6-17 GPO Bit Descriptions 6-74
Table 7-1 SONYXU Module Pin List 7-5
Table 9-1 DCLKG_LOGIC Module Pin List 9-3
Table 9-2 DSP DPLL Register Summary 9-11
Table 9-3 PCTL0 Description 9-12
Table 9-4 PCTL1 Description 9-13
Table 10-1 DSIH Pin List 10-3
Table 10-2 10-5
Table 11-1 Power On Reset characteristics 11-2
Table 11-2 POR Module Pin List 11-2
Table 12-1 SPMBIF Pin List 12-3
Table 12-2 VAB Bits Partition vs. Number of Vectors 12-10
Table 12-3 Customer Specific Logic Signals 12-23
Table 13-1 LEM Module Pin List 13-3
Table 13-2 Control Block I/O List 13-6
Table 13-3 FIRE Encode Transition Mnemonic Definition 13-10
Table 13-4 FIRE Decode Transition Mnemonic Definition 13-11
Table 13-5 Cipher Mnemonic Definition 13-13
Table 13-6 Register Block I/O List 13-15
Table 13-7 Cipher Block I/O List 13-17
Table 13-8 FED Block I/O List 13-19
Table 13-9 Writing of Cipher Key, Kc, if IBR=0 13-20
Table 13-10 Writing of Cipher Key, Kc, if IBR=1 13-20
Table 13-11 Writing of COUNT if IBR=0 13-21
Table 13-12 Writing of COUNT if IBR=1 13-21
Table 13-13 BLOCK1 output if OBR=0 13-21
Table 13-14 BLOCK1 output if OBR=1 13-21
Table 13-15 BLOCK2 output if OBR=0 13-22
Table 13-16 BLOCK2 output if OBR=1 13-22
Table 13-17 Writing data to input for FED processing with IBR=0 13-24
Table 13-18 Writing data to input for FED processing with IBR=1 13-24
Table 13-19 FED output data if OBR=0 13-25
Table 13-20 FED output data if OBR=1 13-25
Table 13-21 LEM Register Summary Y:$FFA0- Y:$FFB1 13-28
Table 13-22 CNTRL Description 13-30
Table 13-23 STAT Description 13-32

Table 13-24 DATA Description 13-33
Table 14-1 BBP Module Pin List 14-3
Table 14-2 BBP Clock Sources 14-8
Table 14-3 BBP Register Summary 14-9
Table 14-4 RCRB Description 14-11
Table 14-5 TCRB Description 14-12
Table 14-6 CRAB Description 14-13
Table 14-7 CRBB Description 14-15
Table 14-8 Mode and Pin Definition Table 14-19
Table 14-9 CRCB Description 14-20
Table 14-10 SSISRB Description 14-23
Table 14-11 RXB Description 14-26
Table 14-12 TSRB Description 14-27
Table 14-13 TXB Description 14-28
Table 14-14 PDRB Description 14-29
Table 14-15 PDRB Description 14-30
Table 14-16 PCRB Description 14-31
Table 15-1 DTimer Pin Diagram 15-3
Table 15-2 DSP Timer Module Register Summary 15-6
Table 15-3 TPLR Description 15-8
Table 15-4 TPCR Description 15-9
Table 15-5 TCR Description 15-10
Table 15-6 TCPR Description 15-11
Table 15-7 TCSR Description 15-13
Table 16-1 SAP Module Pin List 16-3
Table 16-2 SAP Clock Sources 16-9
Table 16-3 Serial Audio Port Register Summary 16-10
Table 16-4 BCBRA Description 16-12
Table 16-5 Recommended Programming of BCARA and BCBRA 16-12
Table 16-6 BCARA Description 16-14
Table 16-7 TCRA Description 16-15
Table 16-8 TCLR Description 16-16
Table 16-9 CRAA Description 16-17
Table 16-10 CRBA Description 16-20
Table 16-11 Mode and Pin Definition Table 16-23
Table 16-12 CRCA Description 16-25
Table 16-13 SSISRA Description 16-28
Table 16-14 RXA Description 16-31
Table 16-15 TSRA Description 16-32

Table 16-16 TXA Description. 16-33
Table 16-17 PDRA Description 16-34
Table 16-18 PRRA Description 16-35
Table 16-19 PCRA Description 16-36
Table 17-1 DMA Module Pin List 17-4
Table 17-2 DMA1 Register Summary 17-8
Table 17-3 DMA2 Register Summary 17-8
Table 17-4 DCRA Description 17-10
Table 17-5 DCRB Description 17-12
Table 17-6 DBAR Description 17-17
Table 17-7 DACN Description 17-18
Table 17-8 DWCR Description. 17-19
Table 17-9 DBSR Description 17-20
Table 17-10 DTOR Description 17-21
Table 17-11 Number of DSP cycles per 16-bit data transfer 17-24
Table 18-1 DMA_ARB Pin List 18-6
Table 19-1 VIAC Module Pin List 19-5
Table 19-2 VIAC Register Summary 19-26
Table 19-3 VIDR Description. 19-28
Table 19-4 VBMR Description. 19-29
Table 19-5 VPTR Description. 19-31
Table 19-6 VODR Description 19-32
Table 19-7 VCSR Description 19-33
Table 19-8 VMR Description 19-37
Table 19-9 VTCR Description 19-39
Table 19-10 VWEDR Description 19-40
Table 19-11 VWADRR Description. 19-41
Table 19-12 VPMARA Description 19-42
Table 19-13 VPMARB Description 19-43
Table 19-14 VDIBA Description 19-44
Table 19-15 VDICA Description 19-45
Table 19-16 VDOBA Description. 19-46
Table 19-17 VDOCA Description. 19-47
Table 19-18 Differnces Between Hard VIAC & VIAC RTL 19-58
Table 20-1 PDDM Module Pin List 20-2
Table 20-2 DSP Program Memory Allocation 20-4
Table 20-3 External Paging Pins Decoding 20-5
Table 20-4 BANKSEL[1:0] 20-6
Table 20-5 PDDM Register Summary 20-7

Table 20-6 PATCH0 Description 20-9
Table 20-7 PATCH Description 20-10
Table 20-8 PCR Description 20-11
Table 20-9 DCSR Description 20-12
Table 20-10 PACMPR Description. 20-15
Table 20-11 YARRL Description 20-16
Table 20-12 YARRH Description. 20-17
Table 20-13 YDCMPR Description 20-18
Table 20-14 YDMR Description. 20-19
Table 20-15 XARRL Description 20-20
Table 20-16 XARRH Description. 20-21
Table 20-17 XDCMPR Description 20-22
Table 20-18 XDMR Description. 20-23
Table 21-1 USB Module Pin List 21-6
Table 21-2 Transceiver Module Pin List. 21-10
Table 21-3 Supported Packet Identifier 21-13
Table 21-4 USB CPU Control Summary 21-24
Table 21-5 USB DSP Control Summary. 21-25
Table 21-6 USBFAD Description. 21-27
Table 21-7 USBIER Description. 21-28
Table 21-8 USBINT Description 21-30
Table 21-9 USBRXCR Description 21-32
Table 21-10 USBTXCR Description 21-34
Table 21-11 USBE1CR Description 21-36
Table 21-12 USBE2CR Description 21-38
Table 21-13 USBE3CR Description 21-40
Table 21-14 USBE4CR Description 21-42
Table 21-15 USBE10CR Description 21-44
Table 21-16 USBCPUCR Description 21-46
Table 21-17 USBDINTE Description 21-48
Table 21-18 USBDIR Description 21-50
Table 21-19 USBSOF Description 21-52
Table 21-20 USBE5CR Description 21-53
Table 21-21 USBE6CR Description 21-55
Table 21-22 USBE7CR Description 21-57
Table 21-23 USBE8CR Description 21-59
Table 21-24 USBE9CR Description 21-61
Table 21-25 USBBPR Description 21-63
Table 21-26 USBDBR Description. 21-64

Table 21-27 USBDRAT Description 21-65
Table 21-28 USBE11CR Description 21-66
Table 22-1 UART Module Pin List. 22-4
Table 22-2 Primary UART Memory Map. 22-6
Table 22-3 Secondary UART Memory Map. 22-7
Table 22-4 Interrupts. 22-8
Table 22-5 RTS Edge Triggered Interrupt Truth Table 22-10
Table 22-6 DTR Edge Triggered Interrupt Truth Table 22-12
Table 22-7 IDLE Detection Truth Table. 22-16
Table 22-8 Majority Vote Results. 22-17
Table 22-9 Baud Rate Automatic Detection 22-19
Table 22-10 Highest Baud Rates. 22-21
Table 22-11 Escape Timer Scaling 22-21
Table 22-12 UART Register Summary. 22-23
Table 22-13 URXD Description 22-26
Table 22-14 UTXD Description 22-28
Table 22-15 UCR1 Description. 22-29
Table 22-16 UCR2 Description. 22-32
Table 22-17 UCR3 Description. 22-35
Table 22-18 UCR4 Description. 22-37
Table 22-19 UFCR Description 22-39
Table 22-20 USR1 Description 22-41
Table 22-21 USR2 Description 22-43
Table 22-22 UECS Description. 22-45
Table 22-23 UTIM Description. 22-46
Table 22-24 UBIR Description 22-47
Table 22-25 UBMR Description 22-48
Table 22-26 UBRC Description 22-49
Table 22-27 BIPRi Description. 22-50
Table 22-28 BMPRi Description. 22-51
Table 22-29 UTS Description 22-52
Table 22-30 UART Low Power State Operation 22-54
Table 22-31 Standard Baud Rates. 22-58
Table 23-1 GPIO Register Summary 23-5
Table 23-2 Internal Interrupt Table, Port B and Port E. 23-22
Table 23-3 PAConfReg0 Description 23-77
Table 23-4 PAConfReg1 Description 23-81
Table 23-5 PADDirReg Description 23-84
Table 23-6 PAIMaskReg Description 23-85

Table 23-7 PAIntCtrlReg Description. 23-86
Table 23-8 PAAItInReg Description. 23-87
Table 23-9 DSPPADataReg Description 23-93
Table 23-10 MCUPADataReg Description. 23-94
Table 23-11 PXConfReg0 Description 23-95
Table 23-12 PXConfReg1 Description 23-99
Table 23-13 PXDDirReg Description 23-102
Table 23-14 PXIMaskReg Description 23-103
Table 23-15 PXIntCtrlReg Description. 23-104
Table 23-16 PXAItInReg Description. 23-106
Table 23-17 IntSelReg Description. 23-112
Table 23-18 IntStatReg Description 23-114
Table 23-19 MCUPXDataReg Description. 23-115
Table 23-20 DSPPXDataReg Description 23-116
Table 23-21 MCUBGref Register Description 23-117
Table 23-22 DSPBGref Register Description 23-118
Table 23-23 Test Mode Muxing 23-119
Table 23-24 DSP BIST Mode Muxing Table 23-122
Table 23-25 Analog Test BIST Mode Muxing Table 23-124
Table 23-26 23-125
Table 23-27 MCU BIST Mode Muxing Table 23-126
Table 23-28 GPIO Module Pin List 23-126
Table 24-1 CMON Electrical Performance. 24-2
Table 24-2 CMON Module Pin List 24-3
Table 25-1 Voice Codec Sub-module list 25-3
Table 25-2 DVOCOD Module Pin List 25-5
Table 25-3 AVOCOD Module Pin List 25-7
Table 25-4 Voice Codec DSP Register Summary 25-9
Table 25-5 VCCR Description 25-10
Table 25-6 VCSR Description 25-12
Table 25-7 VCRX Description 25-14
Table 25-8 VCTX Description 25-15
Table 25-9 VCRX Description 25-16
Table 25-10 VCRX Description 25-17
Table 25-11 VCRX Description 25-18
Table 25-12 Voice Codec Input Clock Settings 25-19
Table 25-13 Decimation filter gain (a) 25-26
Table 25-14 Decimation filter gain (b) 25-26
Table 25-15 ADC signal path gain lineup (for 1.02 kHz signal) 25-27

Table 25-16 LPF Filter Coefficients 25-28
Table 25-17 LPF filter relative group delay 25-31
Table 25-18 HPF Filter Coefficients 25-32
Table 25-19 LPF filter relative group delay 25-36
Table 25-20 Interpolation filter gain (a) 25-39
Table 25-21 Interpolation filter gain (b) 25-40
Table 25-22 DAC signal path gain lineup (for 1.02 kHz signal) 25-41
Table 25-23 DAC signal path gain and interpolation ratio for midi mode(for 1.02 kHz signal) . 25-41
Table 25-24 Midi sampling rate and frequency deviation 25-43
Table 25-25 SD Modulator Quantization Levels 25-44
Table 25-26 Voice Codec General Specifications 25-48
Table 25-27 Voice Codec ADC Specifications 25-48
Table 25-28 Voice Codec DAC Specifications 25-52
Table 26-1 RxAFE Module Pin List 26-3
Table 26-2 RxAFE Test Mode RXAFE_AMUXP/N Output List 26-5
Table 26-3 TXRX Voltage Regulator Power Down Control 26-6
Table 26-4 RxAFE VAG Buffer Specifications 26-6
Table 26-5 RxAFE VAG Buffer Power Save Control 26-6
Table 26-6 RxAFE Reference Voltage Generator Specifications 26-7
Table 26-7 26-7
Table 26-8 Baseband Receive A/D Converters Specifications 26-8
Table 27-1 RxCPROC Pin List 27-3
Table 27-2 Anti-droop filter coefficients 27-8
Table 27-3 First anti-alias filter coefficients 27-9
Table 27-4 GSM/EDGE 27-10
Table 27-5 high bandwidth coefficients 27-12
Table 27-6 THP1 duration 27-12
Table 27-7 medium and low bandwidth coefficients 27-13
Table 27-8 THP2 duration 27-13
Table 27-9 Selectivity filter coefficients of the 15th order FIR 27-14
Table 27-10 Selectivity filter coefficient of the 14th order FIR 27-14
Table 27-11 27-16
Table 28-1 DCADAPT Module Pin List 28-5
Table 28-2 SPI bit description 28-8
Table 29-1 RxSDG Module Pin List 29-4
Table 29-2 Saturation Detection Signal Level Threshold Encoding Method 29-7
Table 29-3 Dither Amplitude Control Decode 29-11
Table 29-4 Digital DIther Disable Threshold Level Encode 29-12

Table 30-1 Nominal Conditions Current Drain. 30-5
Table 30-2 TX Module Pin List 30-6
Table 30-3 Port Description 30-18
Table 30-4 Dual-Port Specifications 30-19
Table 30-5 30-20
Table 30-6 Synthesizer Specific Pin Table 30-22
Table 30-7 Power Consumption 30-24
Table 30-8 30-26
Table 30-9 30-26
Table 30-10 30-27
Table 30-11 Charge Pump Current Output: 30-30
Table 30-12 Neptune Synthesizer Noise Requirement 30-31
Table 30-13 MAIN PLL SPI Bit Definition: 30-32
Table 30-14 30-33
Table 30-15 **osc26m_peak[1:0] Decoded Value**s. 30-36
Table 30-16 syn_osc26m Test Modes. 30-37
Table 30-17 Reference Oscillator Specifications 30-37
Table 30-18 Spurious Emission Requirement for the 26 MHz Oscillator 30-38
Table 30-19 Typical Crystal Specifications 30-38
Table 30-20 Inputs and Outputs Description 30-39
Table 31-1 PAC Module Pin List 31-5
Table 31-2 Clock Ratios 31-13
Table 31-3 Ramp Values. 31-16
Table 31-4 Ramp Look-up Table 31-18
Table 31-5 Path Gain of the Proportional Path and the Integral Path 31-20
Table 31-6 PACII Power Detect Specifications 31-22
Table 31-7 Operating Conditions 31-23
Table 31-8 PAC Input Conditioning Gain Settings 31-23
Table 31-9 PAC INPUT Specifications 31-24
Table 31-10 Analog Activity Detect Specifications 31-24
Table 31-11 PAC Input Filter Specifications 31-25
Table 31-12 ADC Specifications 31-25
Table 31-13 PAC Input & ADC Offset and Dynamic Range 31-26
Table 31-14 Specifications for the D/A Chain 31-28
Table 31-15 PAC-anatest TMUX selection 31-31
Table 31-16 PAC Loop Element Timing Delays 31-38
Table 31-17 Reference Clock 13MHz System Settings Example Values 31-41
Table 32-1 Core Regulator Test Bit Allocation 32-2
Table 32-2 TXRX Regulator Test Bit Allocation 32-3

Table 32-3 Codec Regulator Test Bit Allocation 32-3
Table 32-4 Synthesizer Regulator Test Bit Allocation 32-3
Table 32-5 Analog Control Mux1 Description for Regulators 32-4
Table 32-6 Analog Control Mux2 Descriptions 32-5
Table 32-7 Analog Data Register 2 Description for Regulators 32-6
Table 32-8 regul_core - Neptune 1.575V Regulator Specs 32-6
Table 32-9 Core Regulator Signal Descriptions 32-7
Table 32-10 regul_txrx - Neptune TX,RX, and PAC 2.5V Regulator 32-8
Table 32-11 TX,RX, PAC Regulator Signal Descriptions 32-9
Table 32-12 regul_codec - Neptune Audio Codec 2.5V Regulator 32-10
Table 32-13 Codec Regulator Signal Descriptions 32-11
Table 32-14 Triton Synthesizer Regulator Specs 32-12
Table 32-15 Synth Regulator Signal Descriptions 32-12
Table 32-16 Master Bias Signal Descriptions 32-13
Table 33-1 Pin Electrical Characteristics 33-2
Table 33-2 TUNEC Module Pin List 33-3
Table 33-3 TUNEC Input Signal Description 33-5
Table 33-4 TUNEC Output Pin Description 33-5
Table 33-5 TUNEC Test Pin Description 33-5
Table 33-6 TUNEC SCAN PINS 33-6
Table 34-1 DPLL Specifications 34-4
Table 34-2 DPLL external signal description 34-5
Table 34-3 The DPLL states 34-16
Table 35-1 GPADC Module Pin List 35-3
Table 35-2 GPADC Testing Truth Table 35-5
Table 35-3 GPADC ANA_CNTL and ANA_DATA2 Register 35-6
Table 35-4 Analog Mux1 GPADC Selection 35-6
Table 35-5 Analog Mux2 35-7
Table 35-6 GPADC Control/Status Register 35-7
Table 35-7 GPADC Specification 35-7
Table 35-8 GPADC 1LSB Table 35-8
Table 36-1 A2DIGL Module Pin List 36-5
Table 36-2 Tuning Element Control Register Summary 36-17
Table 36-3 TCODE Description 36-18
Table 36-4 TTEST Description 36-19
Table 36-5 MCU Peripheral Bus Register Summary 36-20
Table 36-6 OSC26M Description 36-21
Table 36-7 Receive and Transmit Modules Interface Register Summary 36-48
Table 36-8 PWRFLT Description 36-50

Table 36-9 PWRFLT Description . . . 36-51
Table 36-10 PWRDAC Description . . . 36-52
Table 36-11 MUXCTL Description . . . 36-53
Table 36-12 Group 1 TX and General Control Description . . . 36-54
Table 36-13 Group2 RX Control 1 Description . . . 36-56
Table 36-14 Group3 RX Control 2 Description . . . 36-59
Table 36-15 Group 4 Fields that are expected to change with mode change . . . 36-62
Table 36-16 Group 5 Fields expected to change with channel selection . . . 36-63
Table 36-17 Group 6 pre-distortion . . . 36-67
Table 36-18 Group 7 TX,PAC parameters which must be programmed before a TX burst 36-67
Table 36-19 Group 8 PAC parameters which must be programmed before a TX burst . . . 36-69
Table 36-20 Group 9 PAC Control . . . 36-70
Table 36-21 Group 10 PAC Control . . . 36-71
Table 36-22 Group 11 PAC Control . . . 36-72
Table 36-23 Group 12 ANATEST Control . . . 36-72
Table 36-24 General Purpose ADC Timing . . . 36-80
Table 36-25 GPADC Data Register Summary . . . 36-82
Table 36-26 GPADC Control/Status Register Summary . . . 36-82
Table 36-27 B+ Description . . . 36-84
Table 36-28 TEMP Description . . . 36-85
Table 36-29 AD1 Description . . . 36-86
Table 36-30 REF_REG Description . . . 36-87
Table 36-31 TXCPDIV2 Description . . . 36-88
Table 36-32 RXCPDIV2 Description . . . 36-89
Table 36-33 CTRL Description . . . 36-90
Table 36-34 WHIGH Description . . . 36-92
Table 36-35 WLOW Description . . . 36-93
Table 36-36 MTR Description . . . 36-94
Table 37-1 MDI Module Pin List . . . 37-6
Table 37-2 MDI Access Type Support at MCU Side . . . 37-11
Table 37-3 MCU-Side Signalling & Control Registers . . . 37-13
Table 37-4 DSP-Side Signalling & Control Registers . . . 37-13
Table 37-5 MDI Register Summary . . . 37-15
Table 37-6 MTR0 Description . . . 37-17
Table 37-8 . . . 37-17
Table 37-7 MTR1 Description . . . 37-18
Table 37-9 . . . 37-18
Table 37-8 MRR0 Description . . . 37-19
Table 37-10 . . . 37-19

Table 37-9 MRR1 Description 37-20
Table 37-11 37-20
Table 37-10 MSR Description 37-21
Table 37-11 MCVR Description 37-26
Table 37-12 MCR Description 37-28
Table 37-13 DTR0 Description 37-31
Table 37-12 37-31
Table 37-14 DTR1 Description 37-32
Table 37-13 37-32
Table 37-15 DRR0 Description 37-33
Table 37-14 37-33
Table 37-16 DRR1 Description 37-34
Table 37-15 37-34
Table 37-17 DSR Description 37-35
Table 37-18 DCR Description 37-39
Table 37-19 MCU Access Timing to the MDI in INDY (Neptune LT) Do not read for Neptune LTE37-48
Table 37-20 MCU Access Timing to the MDI (Number of wait states in terms of MCU clock) 37-49
Table 38-1 RAHB Signal List 38-7
Table 38-2 Significant Address Bits 38-10
Table 38-3 RAM Select 38-14
Table 38-4 ROM Select 38-14
Table 38-5 RAM BIST Sizes 38-16
Table 38-6 ROM BIST Sizes 38-17
Table 38-7 ROM Protection Select in Boot Internal (Normal) 38-22
Table 38-8 ROM Protection Select in Boot External 38-22
Table 38-9 R-AHB Bus Arbitration SIgnal List 38-27
Table 38-10 Big Endian Byte Write Enable Decoding 38-29
Table 38-11 Little Endian Byte Write Enable Decoding 38-30
Table 38-12 CCM_LPMODE Encoding 38-34
Table 38-13 alt_addr_sel Encoding 38-36
Table 38-14 MCU ROM Decoding for hsel_hole 38-37
Table 38-15 Decoding for hsel_prot 38-38
Table 38-16 ARM Platform Pin List 38-42
Table 38-17 R-AHB Timing Constraints 38-49
Table 38-18 R-AHB AC Timing Parameters 38-51
Table 38-19 Alternate Bus Master Interface Constraints 38-52
Table 38-20 Alternate Bus Master AC Timing Parameters 38-54

Table 38-21 ARM7TDMI-S Internal Timing Constraints 38-56
Table 38-22 ARM7TDMI-S External Timing Constraints 38-58
Table 40-1 ARM7 Core Arbiter Pin List 40-3
Table 40-2 CARB_ADDR_SEL Decode 40-7
Table 40-3 CARB_DATA_SEL Decode 40-7
Table 40-4 Mapping ALT_MASTER[2:0] to HMASTER[2:0] 40-8
Table 41-1 R-AHB to IPbus v2.0 Interface Operation (Big Endian Only) 41-4
Table 41-2 IPbus Peripheral Location Decode 41-5
Table 41-3 Transfer Size Encoding 41-6
Table 41-4 R-AHB Master Low Power Mode Encoding 41-6
Table 41-5 AIPI Pin List 41-10
Table 41-6 PSR Data Bus Size Encoding 41-15
Table 41-7 AIPI IPbus v2.0 AC Timing Parameters 41-30
Table 42-1 AMARB Module Pin List 42-4
Table 42-2 AMARB Register Summary 42-7
Table 42-3 Arbiter Control Register Description 42-9
Table 42-4 Arbiter Interval Register Description 42-10
Table 42-5 Software Priority Level Register Description 42-11
Table 42-6 Port 0 Priority Level Register Description 42-12
Table 42-7 Port 1 Priority Level Register Description 42-13
Table 42-8 Port 2 Priority Level Register Description 42-14
Table 42-9 Port 3Priority Level Register Description 42-15
Table 42-10 amarb_master[2:0] Encoding 42-25
Table 43-1 AITC Module Pin List 43-5
Table 43-2 AITC Register Summary 43-8
Table 43-3 INTCNTL Descriptions 43-10
Table 43-4 NIMASK Descriptions 43-13
Table 43-5 INTENNUM Descriptions 43-14
Table 43-6 INTDISNUM Descriptions 43-15
Table 43-7 INTENABLEH / INTENABLEL Descriptions 43-16
Table 43-8 INTTYPEH / INTTYPEL Descriptions 43-17
Table 43-9 NIPRIORITY7 Descriptions 43-18
Table 43-10 NIPRIORITY6 Descriptions 43-19
Table 43-11 NIPRIORITY5 Descriptions 43-20
Table 43-12 NIPRIORITY4 Descriptions 43-21
Table 43-13 NIPRIORITY3 Descriptions 43-22
Table 43-14 NIPRIORITY2 Descriptions 43-23
Table 43-15 NIPRIORITY1 Descriptions 43-24
Table 43-16 NIPRIORITY0 Descriptions 43-25

Table 43-17 NIVECSR Descriptions 43-26
Table 43-18 FIVECSR Descriptions 43-27
Table 43-19 INTSRCH / INTSRCL Description 43-28
Table 43-20 INTFRCH / INTFRCL Descriptions 43-29
Table 43-21 NIPNDH / NIPNDL Descriptions 43-30
Table 43-22 FIPNDH / FIPNDL Descriptions 43-31
Table 43-23 Typical Hardware Accelerated Normal Interrupt Entry Sequence 43-34
Table 43-24 Typical Fast Interrupt Entry Sequence 43-34
Table 44-1 AWPT Module Pin List 44-5
Table 44-2 Watch Point Module Register Summary 44-8
Table 44-3 AWPTD0 through AWPTD15 Description 44-10
Table 44-4 AWPTCNTL Description 44-11
Table 44-5 AWPTENH/AWPTENL Descriptions 44-13
Table 44-6 AWPTA0 through AWPTA3 Description 44-14
Table 44-7 AWPTA4 through AWPTA63 Description 44-15
Table 44-8 AWPTBRPT Description 44-16
Table 44-9 AWPTBASEA Description 44-17
Table 44-10 AWPTSR Description 44-18
Table 44-11 Typical AWPT Opcode Patching Sequence 44-19
Table 45-1 MCUMEM Module Pin List 45-5
Table 45-2 Neptune SRAM Pin Description 45-19
Table 45-3 Neptune ROM Pin Description 45-23
Table 46-1 IPMUX Module Pin List 46-4
Table 46-2 46-10
Table 47-1 IIM Module Pin List 47-2
Table 47-2 UID Register Map Summary (Base Addr = $2485_0000) 47-4
Table 47-3 UID0 Description 47-5
Table 47-4 UID1 Description 47-6
Table 47-5 UID2 Description 47-7
Table 47-6 UID3 Description 47-8
Table 47-7 UID4 Description 47-9
Table 47-8 UID5 Description 47-10
Table 47-9 UID6 Description 47-11
Table 47-10 UID7 Description 47-12
Table 47-11 Hardware Rev Register Description 47-13
Table 48-1 AEIM Module Pin List 48-3
Table 48-2 Interface Requirements for Read and Write Cycles 48-9
Table 48-3 CS0_DPSIZ[2:0] Encoding 48-11
Table 48-4 aeim_siz[1:0] Encoding 48-16

Table 48-5 aeim_qstat[4:0] Encodings - Normal ARM7 Accesses 48-16
Table 48-6 aeim_qstat[4:0] Encodings - Special Cases 48-17
Table 48-7 AEIM Register Summary 48-19
Table 48-8 CS Control Bit Fields 48-21
Table 48-9 AEIM Configuration Register Description 48-24
Table 48-10 CS Primary Configuration Register Descriptions 48-27
Table 48-11 CS Secondary Configuration Register Descriptions 48-29
Table 48-12 CS Wait State Control Register Descriptions 48-34
Table 48-13 CS Burst Clock Control Register Descriptions 48-37
Table 49-1 MQSPI Module Pin List 49-5
Table 49-2 Register Settings Affect on Maximum SCLK Frequency 49-14
Table 49-3 MQSPI Register Summary 49-21
Table 49-4 MQSPI RAM Memory Map 49-26
Table 49-5 SPICDP Description 49-30
Table 49-6 SPICDM Description 49-31
Table 49-7 MCON Description 49-33
Table 49-8 QCFG_L Description 49-35
Table 49-9 QCFG_H Description 49-36
Table 49-10 MTRIG Description 49-37
Table 49-11 STPR_L Description 49-38
Table 49-12 STPR_H Description 49-39
Table 49-13 SDEF Description 49-40
Table 49-14 SFLG Description 49-44
Table 49-15 SPIM Description 49-47
Table 49-16 STFF_L Description 49-49
Table 49-17 STFF_H Description 49-50
Table 49-18 STFE_L Description 49-51
Table 49-19 STFE_H Description 49-52
Table 49-20 QRE_L Description 49-53
Table 49-21 QRE_L Description 49-54
Table 49-22 SDI_MD_L Description 49-55
Table 49-23 SDI_MD_H Description 49-56
Table 49-24 CDPS1 Description 49-57
Table 49-25 CDMS1 Description 49-58
Table 49-26 CDPS2 Description 49-59
Table 49-27 CDMS2 Description 49-60
Table 49-28 CSCFG0A Description 49-61
Table 49-29 Clock Divider Table – 16.8 MHz System Clock 49-63
Table 49-30 CSCFG0B Description 49-65

Table 49-31 Delay After Transfer Table 49-66
Table 49-32 Delay Before Transfer Table 49-68
Table 49-33 QST1A Description 49-74
Table 49-34 QST1B Description. 49-75
Table 49-35 QST2A Description 49-76
Table 49-36 QST2B Description. 49-76
Table 49-37 SST Description 49-78
Table 49-38 SHPFQ1A Description 49-79
Table 49-39 SHPFQ1B Description 49-80
Table 49-40 SHPFQ1C Description 49-81
Table 49-41 SLPFQ1A Description 49-82
Table 49-42 SLPFQ1B Description 49-83
Table 49-43 SLPFQ1C Description 49-84
Table 49-44 SHPFQ2A Description 49-85
Table 49-45 SHPFQ2B Description 49-86
Table 49-46 SHPFQ1C Description 49-87
Table 49-47 SLPFQ2A Description 49-88
Table 49-48 SLPFQ2B Description 49-89
Table 49-49 SLPFQ2C Description 49-90
Table 49-50 BCDP0S1 Description 49-91
Table 49-51 BCDM0S1 Description. 49-92
Table 49-52 BMAS Description 49-96
Table 50-1 L1T Module Pin List. 50-7
Table 50-2 L1 Timer MCU Memory Map 50-11
Table 50-3 L1 Timer MCU Register List 50-11
Table 50-4 Port Definitions: Timing Block 50-15
Table 50-5 Port Definition: Event Generator 50-23
Table 50-6 QBCO Update Summary - MCU 50-29
Table 50-7 Layer 1 Timer: Detectable Errors 50-32
Table 50-8 Layer 1 Timer: Response to Error Conditions 50-34
Table 50-9 MCU Interrupt Generation: Source/Port Map 50-35
Table 50-10 DSP Interrupt Generation: Source/Port Map 50-36
Table 50-11 Layer 1 Timer: Operating Modes 50-38
Table 50-12 Neptune Event Codes 50-41
Table 50-13 FTEC Description 50-50
Table 50-14 FTEQBC Description 50-51
Table 50-15 MEC Description 50-52
Table 50-16 MREQBC Description 50-53
Table 50-17 PTFD Description 50-55

Table 50-18 PTDLV Description 50-56
Table 50-19 Layer 1 Timer Register Summary 50-57
Table 50-20 MSTR Description 50-61
Table 50-21 GTC0 Description 50-62
Table 50-22 GTC1 Description 50-63
Table 50-23 GTS0 Description 50-64
Table 50-24 GTS1 Description 50-66
Table 50-25 IGM Description 50-68
Table 50-26 GPWT Description 50-69
Table 50-27 ES0 Description 50-70
Table 50-28 EM0 Description 50-72
Table 50-29 ES1 Description 50-74
Table 50-30 EM1 Description 50-75
Table 50-31 ES2 Description 50-76
Table 50-32 EM2 Description 50-79
Table 50-33 MCUIS Description 50-80
Table 50-34 MCUIM Description 50-81
Table 50-35 DVIM Description 50-82
Table 50-36 TB_PRE Description 50-83
Table 50-37 TBPREM Description 50-84
Table 50-38 TRQBC Description 50-85
Table 50-39 TRQBC Description 50-86
Table 50-40 TMQBC Description 50-87
Table 50-41 TCFC Description 50-88
Table 50-42 TRFC Description 50-89
Table 50-43 TRAFN_H Description 50-90
Table 50-44 TRAFN_L Description 50-91
Table 50-45 TRAFNM_H Description 50-92
Table 50-46 TRAFNM_L Description 50-93
Table 50-47 TCFN_H Description 50-94
Table 50-48 TCFN_L Description 50-95
Table 50-49 TQBCFT Description 50-96
Table 50-50 TPIT Description 50-97
Table 50-51 TPITM Description 50-98
Table 50-52 QBCO Description 50-99
Table 50-53 QBCO_UDT_F Description 50-100
Table 50-54 QBCO_UDT_Q Description 50-101
Table 50-55 QBCO_UC0 Description 50-102
Table 50-56 QBCO_UC1 Description 50-103

Table 50-57 EG_FT0_CFG Description 50-104
Table 50-58 EG_FT1_CFG Description 50-105
Table 50-59 EG_NFT_CFG Description 50-106
Table 50-60 EG_MA_CFG Description 50-107
Table 50-61 EG_MB_CFG Description 50-108
Table 50-62 EG_MC_CFG Description 50-109
Table 50-63 EG_PT_CFG Description 50-110
Table 50-64 EG_FPTR Description 50-111
Table 50-65 EG_MPTR0 Description 50-112
Table 50-66 EG_MPTR1 Description 50-113
Table 50-67 EG_D0_FD Description 50-114
Table 50-68 EG_D0_QD Description 50-115
Table 50-69 EG_D1_FD Description 50-116
Table 50-70 EG_D1_QD Description 50-117
Table 50-71 EG_D2_FD Description 50-118
Table 50-72 EG_D2_QD Description 50-119
Table 50-73 EG_D3_FD Description 50-120
Table 50-74 EG_D3_QD Description 50-121
Table 50-75 EG_LP0 Description 50-122
Table 50-76 EG_LP1 Description 50-123
Table 50-77 EG_LP2 Description 50-124
Table 50-78 EG_LP3 Description 50-125
Table 50-79 PCFG Description 50-126
Table 50-80 PD Description 50-127
Table 50-81 PS Description 50-128
Table 50-82 PTC Description 50-129
Table 50-83 PDIMASK Description 50-129
Table 50-84 EG_MD Description 50-130
Table 50-85 Layer 1 Timer DSP Register Summary 50-131
Table 50-86 RQBC Description 50-133
Table 50-87 MQBC Description 50-134
Table 50-88 QBCO Description 50-135
Table 50-89 DTX Description 50-136
Table 50-90 DSP_EG_MD Description 50-137
Table 50-91 ISU Downlink Frame Times 50-143
Table 50-92 ISU Uplink Frame Times 50-144
Table 50-93 TCH Frame Table 0 50-145
Table 50-94 TCH Frame Table 1 50-145
Table 50-95 TCH Receive Macro Table A 50-146

Table 50-96 TCH Transmit Macro Table B 50-147
Table 50-97 TCH Power Measurement Macro Table C 50-148
Table 50-98 Delay Registers 50-149
Table 50-99 Loop Registers 50-149
Table 50-100 Example Parameter Table 50-149
Table 51-1 SIM Top Level Interrupt Summary 51-15
Table 51-2 SIM Transmitter Interrupt Summary 51-15
Table 51-3 SIM Receiver Interrupt Summary 51-16
Table 51-4 SIM Port Control Interrupt Summary 51-16
Table 51-5 SIM Port Control Interrupt Summary 51-17
Table 51-6 MCore Interrupt Controller Interface Signals 51-19
Table 51-7 System Signals 51-19
Table 51-8 TEST Interface Signals 51-19
Table 51-9 IP BUS Interface Signals 51-19
Table 51-10 GPIO Interface Signals 51-20
Table 51-11 Secondary SIM Module Interface Signals 51-20
Table 51-12 SIM Module Interrupts 51-45
Table 51-13 Register Summary: $2484X000h - $2484XFFFh 51-46
Table 51-14 PORT1_CNTL Description 51-48
Table 51-15 SETUP Description 51-50
Table 51-16 PORT1_DETECT Description 51-51
Table 51-17 PORT1_XMT_BUF Description 51-52
Table 51-18 PORT1_RCV_BUF Description 51-53
Table 51-19 PORT0_CNTL Description 51-54
Table 51-20 CNTL Description 51-56
Table 51-21 RCV_THRESHOLD Description 51-59
Table 51-22 ENABLE Description 51-60
Table 51-23 XMT_STATUS Description 51-61
Table 51-24 RCV_STATUS Description 51-63
Table 51-25 INT_MASK Description 51-66
Table 51-26 PORT0_XMT_BUF Description 51-68
Table 51-27 PORT0_RCV_BUF Description 51-69
Table 51-28 PORT0_DETECT Description 51-70
Table 51-29 DATA_FORMAT Description 51-71
Table 51-30 XMT_THRESHOLD Description 51-72
Table 51-31 GUARD_CNTL Description 51-73
Table 51-32 OD_CONFIG Description 51-74
Table 51-33 RESET_CNTL Description 51-75
Table 51-34 CHAR_WAIT Description 51-77

Table 51-35 GPCNT Description 51-78
Table 51-36 DIVISOR Description. 51-79
Table 51-37 BLOCK_WAIT_MSB Description 51-79
Table 51-38 BLOCK_WAIT_LSB Description 51-80
Table 52-1 DSM Module Pin List. 52-6
Table 52-2 Port Definitions: Counter Block 52-10
Table 52-3 DSM Register Summary 52-26
Table 52-4 Count32 Description 52-29
Table 52-5 REFCOUNT Description 52-30
Table 52-6 MEASTIME Description 52-31
Table 52-7 SLEEPTIME Description 52-32
Table 52-8 RESTART_TIME Description 52-33
Table 52-9 WAKETIME Description 52-34
Table 52-10 WARMTIME Description 52-35
Table 52-11 LOCKTIME Description 52-36
Table 52-12 CONTROL0 Description 52-37
Table 52-13 CONTROL1 Description 52-39
Table 52-14 CTREN Description 52-41
Table 52-15 STATUS Description 52-42
Table 52-16 STATE Description 52-44
Table 52-17 INT_STATUS Description 52-45
Table 52-18 MASK Description 52-47
Table 52-19 COUNT32_CAP Description 52-49
Table 52-20 WARMPER Description. 52-50
Table 52-21 LOCKPER Description. 52-51
Table 52-22 POSCOUNT Description 52-52
Table 52-23 MGPER Description 52-53
Table 52-24 Measurement Time (msec) vs. DRX mode, Reference Oscillator Accuracy, and Required Timing Slack52-56
Table 53-1 WDOG Module Pin List 53-2
Table 53-2 Watchdog Register Summary 53-9
Table 53-3 WCR Description 53-11
Table 53-4 WSR Description 53-12
Table 53-5 WRSR Description 53-13
Table 54-1 Format of the FCS Field within the RAM 54-4
Table 54-2 Summary of the GEM Function Controls. 54-5
Table 54-3 Processing Cycles Required by GEM to do Various Tasks 54-8
Table 54-4 Potential data access configurations 54-14
Table 54-5 GEM Register Summary. 54-18

Table 54-6 CTL Description 54-20
Table 54-7 Summary of the GEM GPRS Function Controls 54-23
Table 54-8 STAT Description 54-24
Table 54-7 54-25
Table 54-9 Kc Description 54-25
Table 54-10 INPUT Description 54-26
Table 54-11 READ_FRAME_START Description 54-27
Table 54-12 HEADER_LENGTH Description 54-28
Table 54-13 FRAME_LENGTH Description 54-29
Table 54-14 ACCESS_DELAY Description 54-30
Table 54-15 CTL2 Description 54-31
Table 54-16 WRITE_FRAME_START Description 54-33
Table 54-17 54-33
Table 55-1 EGPT Register Summary 55-10
Table 55-2 EGPTCR Description 55-11
Table 55-3 EGPTMR Description 55-14
Table 55-4 EGPTSR Description 55-16
Table 55-5 EGPTIR Description 55-17
Table 55-6 TOCR1 Description 55-19
Table 55-7 TOCR3 Description 55-20
Table 55-8 TOCR4 Description 55-21
Table 55-9 TICR1 Description 55-22
Table 55-10 TICR2 Description 55-23
Table 55-11 PWOR Description 55-24
Table 55-12 TCR Description 55-25
Table 55-13 PWCR Description 55-26
Table 55-14 PWCNR Description 55-27
Table 55-15 ITADRH Description 55-28
Table 55-16 ITADRL Description 55-29
Table 55-17 ITDRH Description 55-30
Table 55-18 ITDRL Description 55-31
Table 55-19 ITCRH Description 55-32
Table 55-20 ITCRL Description 55-33
Table 55-21 EGPT Module Pin List 55-33
Table 56-1 RTR Module Pin List 56-3
Table 56-2 RTR Register Map Summary 56-7
Table 57-1 RTC Pin List 57-3
Table 57-2 RTC Register Summary 57-5
Table 57-3 RTCSCR Description 57-7

Table 57-4 RTCT Description. 57-9
Table 57-5 RTCA Description 57-10
Table 57-6 RTCA_INT/STANDBY output signal 57-12
Table 58-1 KPP Module Pin List 58-3
Table 58-2 Keypad Port Register Summary 58-6
Table 58-3 KPCR Description 58-7
Table 58-4 KPSR Description. 58-8
Table 58-5 KDDR Description 58-9
Table 58-6 KPDR Description 58-10
Table 59-1 DMAC Module Pin List 59-3
Table 59-2 DMAC Register Summary 59-14
Table 59-3 DATA BASE ADDRESS_H Descriptions. 59-16
Table 59-4 DATA BASE ADDRESS_L Descriptions 59-17
Table 59-5 DATA BUFFER SIZE_H Descriptions 59-18
Table 59-6 DATA BUFFER SIZE_L Descriptions 59-19
Table 59-7 COMMAND BASE ADDRESS_H Descriptions. 59-20
Table 59-8 COMMAND BASE ADDRESS_L Descriptions 59-21
Table 59-9 COMMAND BUFFER SIZE_H Descriptions 59-22
Table 59-10 COMMAND BUFFER SIZE_L Descriptions 59-23
Table 59-11 COMMAND STRING SIZE Descriptions 59-24
Table 59-12 FIFO CONFIG Descriptions. 59-25
Table 59-13 LCD CONFIG Descriptions 59-26
Table 59-14 LCD TRANSFER CONFIG Descriptions 59-27
Table 59-15 DMAC Control/Status Descriptions 59-28
Table 59-16 LCD CLOCK CONFIG Descriptions. 59-31
Table 59-17 LCD Write Data Descriptions. 59-32
Table 59-18 DMAC Serial Interface Timing 59-33
Table 59-19 Parallel DMAC Timing Parameters 59-37
Table 59-20 LCD_CLK Frequency Range 59-37
Table 60-1 Single 512 block of example input data (in hexadecimal) 60-3
Table 60-2 HAC Alternate Master Behaviour 60-4
Table 60.2-3 Mapping of burst config bits (no affect on Neptune LTE) 60-6
Table 60-4 HAC Register Summary 60-7
Table 60-5 CTL_STAT Description 60-9
Table 60-6 START_ADDR Description. 60-12
Table 60-7 BLOCK_COUNT Description 60-13
Table 60-8 HSH Description. 60-15
Table 61-1 DRA Module Pin List 61-3
Table 62-1 Revision History 62-1

Table 62-2 MSU Register Summary . . . 62-6
Table 62-3 ID Register Description . . . 62-8
Table 62-4 Set Enable Register Description . . . 62-9
Table 62-5 Clear Enable Register Description . . . 62-10
Table 62-6 Fault Address Register Description . . . 62-11
Table 62-7 Fault Status Register Description . . . 62-12
Table 62-8 Fault Status Bit Encoding . . . 62-13
Table 62-9 Area Enable Register Description . . . 62-14
Table 62-10 Area Control Register Description . . . 62-15
Table 62-11 Area Size Encoding . . . 62-16
Table 62-12 HADDR - Address Segment & Size Alignment . . . 62-17
Table 62-13 Mask[9:0] Encoding . . . 62-17
Table 62-14 Example Area Control Register Programming . . . 62-20
Table 63-1 SCC Pin List . . . 63-2
Table 63-2 Bist Modes . . . 63-6
Table 63-3 Internal Signal Pins . . . 63-11
Table 63-4 SCC Secure RAM Register Summary . . . 63-14
Table 63-5 RED_START Description . . . 63-16
Table 63-6 BLACK_START Description . . . 63-17
Table 63-7 LENGTH Description . . . 63-18
Table 63-8 Control Description . . . 63-19
Table 63-9 Status Description . . . 63-20
Table 63-10 ERROR Description . . . 63-22
Table 63-11 INT_MASK Description . . . 63-24
Table 63-12 Configuration Description . . . 63-25
Table 63-13 INIT_VECTOR Description . . . 63-26
Table 63-14 Code Bit Position . . . 63-31
Table 63-15 SP_ERR Assertion . . . 63-41
Table 63-16 Security Policy . . . 63-41
Table 63-17 Security Policy Outputs . . . 63-42
Table 63-18 Algorithm Sequence Checker LFSR Step Values . . . 63-44
Table 63-19 . . . 63-49
Table 63-20 SCC Security Module Register Summary . . . 63-51
Table 63-21 Address Map . . . 63-52
Table 63-22 Status Description . . . 63-53
Table 63-23 Command Description . . . 63-55
Table 63-24 SEQSTART Description . . . 63-56
Table 63-25 SEQEND Description . . . 63-57
Table 63-26 SEQCHK Description . . . 63-58

Table 63-27 BITCNT Description 63-59
Table 63-28 INCSIZE Description 63-60
Table 63-29 BBDEC Description 63-61
Table 63-30 COMPSIZE Description 63-62
Table 63-31 PTCHK Description 63-63
Table 63-32 CTCHK Description 63-64
Table 63-33 TIMERIV Description 63-65
Table 63-34 TIMERCTL Description 63-66
Table 63-35 TIMER Description 63-67
Table 63-36 DDSTATUS Descriptions 63-68
Table 63-37 63-71
Table 64-1 IIM Module Pin List 64-2
Table 66-1 1-Wire Port Definitions. IP bus interface 66-3
Table 66-2 1-Wire Port Definitions. DS2502 66-3
Table 66-3 1-Wire Port Definitions. System 66-3
Table 66-4 1-Wire Port Definitions. Test 66-3
Table 66-5 1-Wire Register Map Summary: $00211000 - $00211FFF 66-6
Table 66-6 CONTROL Description 66-8
Table 66-7 Time Divider Description 66-9
Table 66-8 System timing requirements 66-9
Table 66-9 Examples of relative time imprecision 66-10
Table 66-10 RPP Sequence delay comparisons 66-11
Table 66-11 WR0 Sequence delay comparisons 66-12
Table 66-12 WR1 Sequence delay comparisons 66-13
Table 67-1 Bus Mux States 67-6
Table 67-2 Test Modes for LTE 67-8
Table 67-3 MCU Memory BIST modes 67-9
Table 67-4 TCM Register Summary 67-12
Table 67-5 TCR Description 67-15
Table 67-6 MTCR Description 67-17
Table 67-7 DSP Test Modes 67-18
Table 67-8 MBIST Description 67-19
Table 67-9 ANA_CTL Description 67-20
Table 67-10 Analog Mux1 67-20
Table 67-11 Analog Mux2 67-21
Table 67-12 Data Register 1 Descriptions 67-23
Table 67-13 67-23
Table 67-14 Data Register 2 Descriptions 67-23
Table 67-15 MCUInBist Description 67-25

Table 67-16 TCM Module Pin List 67-26
Table 67-17 AMT Module Pin List. 67-40
Table 68-1 All the test modes available for Neptune LTE 68-5
Table 68-2 Scan Test Mode Description. 68-9
Table 68-3 Scan Control Signal Generation 68-10
Table 68-4 Scan Chains from Platforms and SOG to be routed in LTE. 68-11
Table 68-5 Top scan router ipt_arm7 scan chain connections. 68-14
Table 68-6 Top scan router ipt_onyx scan chain connections. 68-16
Table 68-7 Top scan router ipt_sog scan chain connections. 68-18
Table 68-8 Top scan router ipt_dig_radio scan chain connections. 68-20
Table 68-9 STC Module Pin List 68-22
Table 69-1 INT Pin List 69-3
Table 69-2 MCU External Interrupts Register Summary 69-5
Table 69-3 EPPAR Description 69-6
Table 69-4 EPDDR Description 69-10
Table 69-5 EPDR Description. 69-11
Table 69-6 EPFR Description 69-12
Table 70-1 BIST Module Pin List. 70-3
Table 70-2 ipt_test_membist_fail Operating Modes. 70-4
Table 70-3 BIST Operating Modes. 70-7
Table 70-4 Safe States for Output Signals When BIST is Disabled. 70-7
Table 70-5 Production Test Mode Stop on Fail Input and Output Interaction 70-8
Table 70-6 Debug Test Mode Continue on Fail Input and Output Interaction. 70-9
Table 70-7 Unlinked Functional Faults. 70-10
Table 70-8 Data Backgrounds Required for 32 I/O Memory 70-11
Table 70-9 Data Backgrounds Required for 24 I/O Memory 70-11
Table 70-10 Data Backgrounds Required for 16 I/O Memory 70-11
Table 70-11 Fault Coverage for Specified March Algorithms 70-12
Table 71-1 SJC Pin List 71-5
Table 71-2 JTAG Instructions. 71-15
Table 71-3 JTCR Description 71-19
Table 71-4 TMS Sequencing for entering ARM-Onyx serial chain via JTAG Control . . 71-22
Table 71-5 TMS Sequencing for Simultaneous Release of ARM ONYX Debug Mode via JTAG Control71-23
Table 72-1 72-3
Table 72-2 72-3
Table 72-3 Analog Mux1 72-5
Table 72-4 Analog Mux2 72-5
Table 72-5 Anatest Control Register Summary 72-7

Table 72-6 Anatest Data1 Register Summary 72-7
Table 72-7 Anatest Data2 Register Summary 72-7
Table 72-8 ANA_CTL Description 72-9
Table 72-9 Data Register Descriptions 72-10
Table 72-10 Data Register Descriptions 72-11
Table 72-4 72-11
Table 72-5 ANATEST Pin List 72-12
Table 73-1 PADIO Module Pin List 73-4
Table 73-2 Recommended Operating Conditions 73-6
Table 73-3 Electrical Characteristics over Industrial Operating Conditions 73-6
Table 73-4 General Operating Specifications 73-7
Table 74-1 Sizes for Neptune 2 ESD Network Components 74-6
Table 74-2 Pad Cells 74-11
Table 74-3 Power Down Modes 74-13
Table 75-1 DC Recommended Operating Conditions 75-2
Table 75-2 DC Absolute Maximum Operating Conditions 75-2
Table 75-3 DC Electrical Characteristics 75-3
Table 75-4 DSP Current Drain 75-4
Table 75-5 MCU Current Drain 75-5
Table 75-6 Deep Sleep Current 75-5
Table 75-7 Power Sequence Delays 75-7
Table 75-8 Electrical Restrictions 75-8
Table 75-9 Valid REFDPLL Settings & 2XPATREF, MCU, & DSP Divider 75-8
Table 75-10 Internal DSP Clocks 75-12
Table 75-11 DSP Clocks 75-13
Table 75-12 MCU Clocks 75-13
Table 75-13 Mode Select and Interrupt Timing Specifications 75-15
Table 75-14 AEIM External Bus AC Timing Specifications Relative to M_CLK 75-16
Table 75-15 AEIM External Bus AC Timing Specifications Relative to CKO 75-19
Table 75-16 Subscriber Interface Module Timings 75-23
Table 75-17 MQSPI Timing 75-26
Table 75-18 USB Clocking 75-27
Table 75-19 USB Input Timings 75-27
Table 75-20 BBP Timing Specification 75-28
Table 75-21 BBP Timing 75-30
Table 75-22 Key To Table 2 SAP Timing 75-33
Table 75-23 SAP Timing 75-34
Table 75-24 JTAG Timing 75-37
Table 76-1 Neptune LTE 280 Production MBGA Ball Pad to Signal Name Net List 76-5

Table 76-2 Neptune LTE 280 Development MBGA Ball Pad to Signal Name Net List . 76-10
Table A-1 Boot Source Configuration A-2
Table A-2 Boot Mode Configuration - Internal Boot ROM A-2
Table A-3 SPS MCU Boot Modes: Clock Configuration A-2
Table A-4 DSP Boot Mode Pin Configuration A-6
Table A-5 SPS Test Mode Pin Configuration A-6
Table A-6 Message Summary A-10
Table A-7 MDI_R0 Description (Short Messages) A-10
Table A-8 MDI_R0 Description (Long Messages) A-12
Table A-9 XYP Fields A-13
Table A-10 Allocation of Neptune Pins to Various Boot Configurations A-20
Table B-1 Tri-State Enable Configuration. B-23
Table B-2 . . . . . B-29
Table B-3 ARM7 Scan Chains B-32
Table B-4 ARM7 Platform Scan Cell Counts B-33
Table B-5 Proposed ARM7 Platform Non-Wrapper Scan Chains B-33
Table B-6 Analog / Mixed-signal and Related Modules B-36
Table B-7 Example RAM Operation with Late Write. B-41

List of Figures

Figure 1-1 Neptune LTE Block Diagram 1-16
Figure 3-1 Neptune Clock Distribution (Page 1 of 2) 3-3
Figure 3-2 Neptune Clock Distribution (Page 2 of 2) 3-4
Figure 3-3 CLKGEN Block Diagram. 3-7
Figure 5-1 CCM Block Diagram 5-3
Figure 5-2 DPLL Port Timing 5-26
Figure 5-3 MFN Load Request Timing 5-27
Figure 5-4 DPLL Reset Logic 5-27
Figure 5-5 Deep Sleep/Clock Cleaner Relationship. 5-28
Figure 5-6 Clock Controls Diagram 5-29
Figure 5-7 Glitchless Clock Mux Timing. 5-30
Figure 5-8 Divide-by-1 to Divide-by-3 5-31
Figure 5-9 Divide-by-5 to Divide-by-2 5-31
Figure 5-10 Entering Low-Power mode. 5-33
Figure 5-11 Restoring Clocks while in low power mode. 5-34
Figure 5-12 Re-enabling clocks and exiting low-power mode. 5-35
Figure 5-13 CKO(H) Selection. 5-36
Figure 5-14 MOD PIN LOGIC 5-37
Figure 5-15 RESET Logic 5-38
Figure 5-16 5-38
Figure 5-17 5-39
Figure 5-18 5-39
Figure 5-19 Synchronization of Reset Deassertion 5-40
Figure 5-20 CKIL Clock Monitor Timing Diagram. 5-41
Figure 6-1 Block Diagram 6-3
Figure 6-2 DSP 56600 P Memory Map - Neptune LTE. 6-10
Figure 6-3 DSP 56600 X and Y Memory Map - Neptune LTE 6-11
Figure 6-4 RQA Interrupt Connection Within Neptune. 6-25
Figure 6-5 DSP DMA Configuration 6-29
Figure 6-6 Neptune LTE MCU Memory Map (MOD - High). 6-30
Figure 6-7 Neptune LTE ROM Boot configuration 6-31
Figure 6-8 Neptune LTE MCU RAM Memory Spaces 6-31
Figure 6-9 Netpune LTE EIM (External) Memory Space 6-32
Figure 6-10 MCU Peripheral Memory Space. 6-33

Figure 6-11 Mixed Signal Modules internal connections 6-65
Figure 6-12 MQSPI/A2DIGL Internal Connections 6-66
Figure 6-13 L1T/A2DIGL Internal Connections 6-67
Figure 6-14 TX/RxCPROC/A2DIGL/BBP internal connections. 6-68
Figure 6-15 Connections for BBP Glue Logic 6-69
Figure 6-16 Typical Waveforms For BBP Transmit - TX Receive 6-70
Figure 6-17 Typical Waveforms For BBP Receive - RxCPROC Transmit 6-70
Figure 7-1 Core Interface Signals. 7-4
Figure 7-2 DSP Address Tracing Timing Diagram 7-20
Figure 8-1 DSPMEM Block Diagram 8-2
Figure 8-2 X/Y Data Memory Block Late Write Architecture 8-3
Figure 8-3 Interleaved RAM Block Partitioning 8-5
Figure 9-1 DCLKG_LOGIC Clock Sources and Clock Outputs 9-2
Figure 9-2 DCLKG_LOGIC General Block Diagram 9-6
Figure 9-3 DCLKG_LOGIC Micro Architecture. 9-7
Figure 10-1 10-2
Figure 10-2 10-5
Figure 11-1 Power on Reset Block Diagram 11-2
Figure 12-1 New Custom Module hierarchy 12-2
Figure 12-2 S-PMB Interface Clock Logic 12-5
Figure 12-3 Write Enable Signal Timing Diagram 12-6
Figure 12-4 Write Enable Signals Connections 12-6
Figure 12-5 Write Enable Connections, DMA Accessible Register 12-7
Figure 12-6 Read Strobe and Enable Signals Connections. 12-7
Figure 12-7 Read Enable Signal. 12-8
Figure 12-8 Core Data Read Buffer 12-9
Figure 12-9 Interrupt Request. 12-9
Figure 12-10 Interrupt Vector Address Bus 12-11
Figure 12-11 The Scan Mode Signal. 12-12
Figure 12-12 The Scan Enable Signal 12-13
Figure 12-13 The Scan Clock Source (scan_clk) 12-13
Figure 12-14 Asynchronous Reset Implementation 12-14
Figure 12-15 Core Write Access Timing Diagram 12-16
Figure 12-16 Core Read Access Timing Diagram 12-16
Figure 12-17 DMA Write Access Timing Diagram. 12-17
Figure 12-18 DMA Read Access Timing Diagram 12-17
Figure 12-19 Shared Memory Write Access 12-18
Figure 12-20 Shared Memory Read Access 12-18
Figure 12-21 Interrupt Timing Diagram. 12-19

Figure 12-22 DMA Request Timing Diagram 12-20
Figure 12-23 Fast DMA Request Timing Diagram. 12-21
Figure 12-24 The NDGCTL timing diagram 12-22
Figure 12-25 Clock in Scan Mode 12-24
Figure 12-26 Clock Behavior in STOP Mode 12-25
Figure 12-27 Write Accessible Registers 12-26
Figure 12-28 Core Readable Registers - Flip Flop Implementation. 12-27
Figure 12-29 Core Readable Registers - Latch Implementation. 12-28
Figure 12-30 DMA Write Accessible Registers. 12-28
Figure 12-31 DMA Read Accessible Registers 12-28
Figure 12-32 Receive and Transmit Register Status Flags. 12-29
Figure 12-33 Sticky Status Bit Implementation 12-29
Figure 12-34 Interrupt Logic 12-30
Figure 12-35 Using Interrupt Acknowledge to De-assert the Request. 12-30
Figure 12-36 12-31
Figure 12-37 Clock Gating. 12-31
Figure 13-1 Top Level LEM Overview 13-2
Figure 13-2 Layer 1 Encryption Module I/O 13-3
Figure 13-3 Block Connectivity of Major Signals 13-5
Figure 13-4 Control Block I/O Diagram 13-6
Figure 13-5 Internal Control Block Diagram 13-8
Figure 13-6 Fire Portion of FSM 13-12
Figure 13-7 Cipher Portion of FSM 13-14
Figure 13-8 Register Block I/O Schematic. 13-15
Figure 13-9 Cipher Block I/O Schematic 13-17
Figure 13-10 Cipher Block Architecture 13-18
Figure 13-11 FED Block LFSR 13-18
Figure 13-12 FED Block I/O 13-19
Figure 13-13 Flow Diagram of Cipher Block Usage 13-23
Figure 13-14 Flow Diagram of FED Usage 13-27
Figure 14-1 BBP Pin Out 14-2
Figure 14-2 BBP Block Diagram 14-5
Figure 14-3 14-6
Figure 14-4 14-38
Figure 15-1 DSP Timer Module Block Diagram 15-2
Figure 15-2 16-bit Timer Module Block Diagram 15-3
Figure 15-3 DTimer Pin Diagram 15-5
Figure 16-1 SAP Block Diagram 16-3
Figure 16-2 SAP Pin Diagram 16-6

Figure 16-3 SAP Connectivity Diagram 16-7
Figure 16-4 SAP Clock Generator Block Diagram 16-19
Figure 17-1 DMA Interfaces 17-2
Figure 17-2 DMA Pin Diagram 17-3
Figure 17-3 DMA Block Diagram 17-7
Figure 17-4 DMA Pack/Unpack Modes Summary 17-16
Figure 18-1 DMA-Peripheral timing diagram 18-2
Figure 18-2 DMA-DSP Peripherals Channels Inside Patriot -Indy Chip. 18-3
Figure 18-3 18-4
Figure 18-4 DMA_ARB Pin Diagram 18-5
Figure 19-1 Using VIAC in MODEM procedure. 19-2
Figure 19-2 Viterbi Accelerator Interface Diagram 19-4
Figure 19-3 Viterbi Accelerator Block Diagram 19-7
Figure 19-4 ACS - Add Compare Select Function. 19-8
Figure 19-5 WED - Window Error Detection Function. 19-8
Figure 19-6 Feeding VIAC in lockstep mode 19-9
Figure 19-7 VIAC's Pipeline in equalization, in lockstep mode 19-10
Figure 19-8 VIAC's Pipeline in Convolutional decoding, constraint length of 5, 1/2 code rate, in lockstep mode19-10
Figure 19-9 VIAC's Pipeline in Convolutional decoding, constraint length of 5, 1/3 or 1/6 code rate, in lockstep mode19-10
Figure 19-10 VIAC's Pipeline in convolutional decoding in constraint length of 7, 1/3 or 1/6 code rate, in lockstep mode19-11
Figure 19-11 VIAC's DMA accesses in independent mode 19-12
Figure 19-12 VIAC's Pipeline in Equalization, in independent mode. 19-12
Figure 19-13 VIAC's Pipeline in Convolutional decoding, constraint length of 5, 1/2 code rate, in independent mode19-13
Figure 19-14 VIAC's Pipeline in Convolutional decoding, constraint length of 5, 1/3 or 1/6 code rate, in independent mode19-13
Figure 19-15 VIAC's Pipeline in convolutional decoding in constraint length of 7, in independent mode19-13
Figure 19-16 VPMAR FIFO. 19-14
Figure 19-17 Viterbi Butterfly Structure 19-15
Figure 19-18 Wrap around handles overflow of path metrics 19-16
Figure 19-19 DMA buffers organization, in equalization 19-17
Figure 19-20 DMA buffers organization, in convolutional decoding, code rate 1/2, constraint length of 5, no packing19-17
Figure 19-21 DMA buffers organization, in convolutional decoding, code rate 1/2, constraint length of 5, packing of 8 bits19-18
Figure 19-22 DMA buffers organization, in convolutional decoding, code rate 1/2, constraint length of 5, packing of 4 bits19-19

Figure 19-23 DMA buffers organization, in convolutional decoding, code rate 1/2, constraint length of 7, no packing19-20
Figure 19-24 DMA buffers organization, in convolutional decoding, code rate 1/2, constraint length of 7, packing of 8 bits19-21
Figure 19-25 DMA buffers organization, in convolutional decoding, code rate 1/2, constraint length of 7, packing of 4 bits19-22
Figure 19-26 DMA buffers organization, in convolutional decoding, code rate 1/3 or 1/6, constraint length of 719-23
Figure 19-27 DMA buffers organization, in convolutional decoding, code rate 1/2, constraint length of 7, packing of 8 bits19-24
Figure 19-28 DMA buffers organization, in convolutional decoding, code rate 1/2, constraint length of 7, packing of 4 bits19-25
Figure 19-29 VIAC states 19-48
Figure 19-30 Example of VIAC normal operation in lockstep mode 19-49
Figure 19-31 Example of VIAC normal operation in independent mode 19-49
Figure 19-32 Preparing the VIAC for trellis main loop, in lockstep mode 19-51
Figure 19-33 Preparing the VIAC for trellis main loop, in equalization, in independent mode . . 19-51
Figure 19-34 Preparing the VIAC for trellis main loop, in convolutional decoding, in lockstep mode19-53
Figure 19-35 Preparing the VIAC for trellis main loop, in convolutional decoding, in independent mode19-54
Figure 20-1 Data Watchpoint (DW) Block Diagram 20-5
Figure 21-1 USB Block Diagram 21-3
Figure 21-2 USB Peripheral 21-5
Figure 21-3 Supported Transaction Types per Endpoint 21-11
Figure 21-4 Supported USB Packet Types 21-12
Figure 21-5 Sync Pattern 21-12
Figure 21-6 5 bit CRC Calculation for Address and Endpoint Fields 21-14
Figure 21-7 16 bit CRC Calculation for Data Field 21-15
Figure 21-8 EOP Width Timing 21-16
Figure 21-9 NRZI Data Encoding 21-16
Figure 21-10 Flow Diagram for NRZI 21-17
Figure 21-11 Bit Stuffing 21-18
Figure 21-12 Flow Diagram for Bit Stuffing 21-19
Figure 21-13 USB Memory Maps (With CPU Memory Map Selection) 21-23
Figure 21-14 Register Summary Legend 21-24
Figure 22-1 UART Module Interface Signals 22-3
Figure 22-2 General Connections for a UART with a Modem 22-10
Figure 22-3 UART Block Diagram and Clock Generation Diagram 22-13
Figure 22-4 Transmitter FIFO Empty Interrupt Suppression Flow Chart 22-15

Figure 22-5 Baud Rate Detection Protocol Diagram 22-20
Figure 22-6 Majority Vote Results 22-55
Figure 22-7 Baud Rate Detection of Divisor = 1 22-56
Figure 23-1 GPIO Overall Block Diagram 23-3
Figure 23-2 23-11
Figure 23-3 23-12
Figure 23-4 23-13
Figure 23-5 23-14
Figure 23-6 23-15
Figure 23-7 23-16
Figure 23-8 23-18
Figure 23-9 23-20
Figure 23-10 23-21
Figure 23-11 Port A Top Level Block Diagram 23-25
Figure 23-12 Port A Output Configuration Block Diagram 23-26
Figure 23-13 External Interrupt Control Block Diagram 23-27
Figure 23-14 Alternate Input Functionality Block Diagram 23-28
Figure 23-15 gpio_porta_funcout[3] Configured for scan_mode 23-29
Figure 23-16 * gpio_porta_funcout[3] Configured for scan_mode 23-30
Figure 23-17 * gpio_porta_altin[2] configured for scan_mode 23-31
Figure 23-18 * gpio_porta_altin[2] configured for scan_mode 23-32
Figure 23-19 * gpio_porta_funcout[3] configured for scan_mode, refer Appendix 23-33
Figure 23-20 * gpio_porta_funcout[3] configured for scan_mode 23-34
Figure 23-21 * gpio_porta_altin[2] configured for scan_mode 23-35
Figure 23-22 * gpio_porta_altin[2] configured for scan_mode 23-36
Figure 23-23 * gpio_porta_altin[3] configured for scan_mode 23-37
Figure 23-24 * gpio_portb_altin[3] configured for scan_mode 23-38
Figure 23-25 23-39
Figure 23-26 * gpio_portb_altin[3] configured for scan_mode 23-40
Figure 23-27 * gpio_portb_altin[3] configured for scan_mode 23-41
Figure 23-28 * gpio_portb_altin[3] configured for scan_mode 23-42
Figure 23-29 23-43
Figure 23-30 * gpio_portb_altin[2] configured for scan_mode 23-44
Figure 23-31 * gpio_portb_altin[2] configured for scan_mode 23-45
Figure 23-32 * gpio_portb_altin[2] configured for scan_mode 23-46
Figure 23-33 * gpio_portb_altin[2] configured for scan_mode 23-47
Figure 23-34 23-48
Figure 23-35 * gpio_portc_altin[2] configured for scan_mode 23-49
Figure 23-36 * gpio_portc_altin[2] configured for scan_mode 23-50

Figure 23-37 * gpio_portc_altin[2] configured for scan_mode 23-51
Figure 23-38 * gpio_portc_altin[2] configured for scan_mode 23-52
Figure 23-39 * gpio_portc_altin[3] configured for scan_mode 23-53
Figure 23-40 * gpio_portc_altin[2] configured for scan_mode 23-54
Figure 23-41 * gpio_portd_funcout[1] configured for scan_mode 23-55
Figure 23-42 * gpio_portd_funcout[1] configured for scan_mode 23-56
Figure 23-43 * gpio_portd_funcout[1] configured for scan_mode 23-57
Figure 23-44 23-58
Figure 23-45 23-59
Figure 23-46 Port B Top Level Block Diagram 23-60
Figure 23-47 Alternate Input Functionality Block Diagram 23-61
Figure 23-48 Port B Output Configuration Block Diagram 23-62
Figure 23-49 External Interrupt Control Block Diagram Port B 23-63
Figure 23-50 Port C Top Level Block Diagram 23-64
Figure 23-51 Port C Output Configuration Block Diagram 23-65
Figure 23-52 External Interrupt Control Block Diagram Port C 23-66
Figure 23-53 Alternate Input Functionality Block Diagram Port C 23-67
Figure 23-54 Port D Top Level Block Diagram 23-68
Figure 23-55 Port D Output Configuration Block Diagram 23-69
Figure 23-56 External Interrupt Control Block Diagram Port D 23-70
Figure 23-57 Alternate Input Functionality Block Diagram Port D 23-71
Figure 23-58 Port E Top Level Block Diagram 23-72
Figure 23-59 Port E Output Configuration Block Diagram 23-73
Figure 23-60 External Interrupt Control Block Diagram Port E 23-74
Figure 23-61 Alternate Input Functionality Block Diagram Port E 23-75
Figure 23-62 Configuring port to be used as scan_output at gpio_portx_funcout[1] 23-136
Figure 23-63 Configuring port to be used as scan input at gpio_portx_altin[2] 23-137
Figure 24-1 Clock Monitor Timing Diagram 24-2
Figure 24-2 CMON Input and Output Diagram 24-3
Figure 24-3 Clock Monitor and Watchdog Connection Diagram 24-4
Figure 25-1 Voice Codec Block Diagram 25-2
Figure 25-2 Voice Codec Module Pin and Interconnection Diagram 25-3
Figure 25-3 Voice Codec ADC Signal Path Block Diagram 25-19
Figure 25-4 Functional Diagram of the A/D Sigma-Delta Converter 25-21
Figure 25-5 Decimation Filter Block Scheme 25-23
Figure 25-6 Frequency Response of the Filter for Even K (a) normalized to 0 dB at DC.25-24
Figure 25-7 Frequency Response of the Filter for Even K (b) 25-24
Figure 25-8 Frequency Response of the Filter for Even K (c) 25-25

Figure 25-9 Frequency Response of the Filter for Even K (d) 25-25
Figure 25-10 LPF filter frequency response 25-28
Figure 25-11 Total Frequency response without HPF (decimation filter + LPF) (a) 25-29
Figure 25-12 Total Frequency response without HPF (b) 25-30
Figure 25-13 Total Frequency response without HPF (c) 25-30
Figure 25-14 Total Group Delay without HPF 25-31
Figure 25-15 HPF Filter Frequency Response (a) 25-32
Figure 25-16 HPF Filter Frequency Response (b) 25-33
Figure 25-17 Total Frequency response with HPF (a) 25-34
Figure 25-18 Total Frequency response with HPF (b) 25-34
Figure 25-19 Total Frequency response with HPF (c) 25-35
Figure 25-20 Total Group Delay with HPF 25-36
Figure 25-21 Voice Codec DAC Signal Path Block Diagram 25-37
Figure 25-22 Total DAC Frequency response without HPF 25-38
Figure 25-23 Total DAC Frequency response with HPF 25-38
Figure 25-24 Interpolation Filter Block Scheme 25-43
Figure 25-25 SD Modulator Scheme 25-44
Figure 25-26 SD Modulator Hardware Implementation Scheme 25-45
Figure 25-27 Spectrum at DAC Sigma-Delta Output (with half full scale 1 kHz sine wave signal)25-45
Figure 25-28 Voice Codec DAC Dynamic Matching Module 25-46
Figure 25-29 Current Sources Mismatch Noise After Second Order Dynamic Matching (Compared to Sigma - Delta Output)25-47
Figure 25-30 Voice Signal Frequency Response Requirements at the ADC Path (VCIHPF=0, LPF Alone Without HPF)25-51
Figure 25-31 Voice Signal Frequency Response Requirements at the ADC Path (VCIHPF=1, HPF and LPF Together)25-51
Figure 25-32 Voice Signal Frequency Response Requirements at the DAC Path (VCOHPF=0, LPF Alone Without HPF)25-54
Figure 25-33 Voice Signal Frequency Response Requirements at the DAC Path (VCOHPF=1, HPF and LPF Together)25-54
Figure 26-1 RxAFE Block Diagram 26-2
Figure 26-2 Bypass Test Mode Connection 26-3
Figure 26-3 RX SDM Output Timing 26-10
Figure 26-4 Normalized Baseband Receive I/Q Sigma Delta Modulator Block Diagram 26-11
Figure 27-1 Block diagram of Rx chain in the Neptune 27-2
Figure 27-2 RxCPROC pinout diagram 27-5
Figure 27-3 Rx coprocessor Block Diagram 27-7
Figure 27-4 Serial Interface startup 27-24
Figure 27-5 First GSM format 27-25

Figure 27-6 Second GSM format 27-26
Figure 27-7 Third GSM format 27-27
Figure 27-8 Receive timing diagram in VLIF mode 27-30
Figure 27-9 Receive timing diagram in DCR mode 27-31
Figure 28-1 RX Lineup (Simplified) 28-3
Figure 28-2 DCADAPT I/O 28-4
Figure 28-3 DCADAPT: Block Diagram 28-7
Figure 28-4 SPI bit positions 28-8
Figure 28-5 Fast DCADAPT Algorithm (One Channel, I or Q) 28-9
Figure 29-1 Block Diagram of the Receiver Sigma-Delta Modulator with Dither Generator . . . 29-2
Figure 29-2 Block Diagram of the Receiver Saturation Detection and SDM Dither Generation 29-3
Figure 29-3 29-4
Figure 29-4 Zoomed views of the Frequency response of the SINC filter 29-9
Figure 29-5 The Residual Quantization Noise at the output of the SINC filter 29-10
Figure 30-1 Black Box Pin List 30-4
Figure 30-2 Transmit Architecture 30-5
Figure 30-3 FN_Mode/DP_Mode 30-11
Figure 30-4 Detailed Block Diagram of Tx Data System 30-13
Figure 30-5 GSM Predistortion Mode 30-14
Figure 30-6 Coefficient Generator 30-15
Figure 30-7 Bit Alignment of the Various Pre-distortion Coefficients 30-16
Figure 30-8 Waveform Generator Block Diagram 30-17
Figure 30-9 Block Diagram of the D/A Chain - Analog Part 30-18
Figure 30-10 Block Pin-out 30-18
Figure 30-11 Dual_port Analog Testing 30-21
Figure 30-12 TX Synthesizer Block Diagram 30-24
Figure 30-13 Power Sequencing 30-32
Figure 30-14 OSC26M Block Diagram 30-35
Figure 30-15 OSC26M Pin Diagram 30-39
Figure 31-1 Power Control Top-Level View 31-2
Figure 31-2 Black Box Pin Diagram 31-3
Figure 31-3 PAC System Diagram 31-5
Figure 31-4 PAC Clocking Example 31-14
Figure 31-5 RAMP_clk And Saturation Timing Diagram 31-15
Figure 31-6 The Activity Detection Section of the AOC Block Diagram 31-17
Figure 31-7 The Proportional and Integral Control Section of the AOC Block Diagram . 31-19
Figure 31-8 Analog Modules 31-21

Figure 31-9 PACII Power Detector Response 31-22
Figure 31-10 PAC INP COND 31-24
Figure 31-11 The D/A Chain Section of the AOC Block Diagram 31-27
Figure 31-12 PA D/A Chain Limits 31-28
Figure 31-13 PAC DAC Chain showing DAZERO and LOW_BATT switches 31-29
Figure 31-14 Analog Test For ADC 31-30
Figure 31-15 GSM Symbol - Digital /Analog Zero Reference Timing 31-33
Figure 31-16 The Timing Events in the GSM Initialization State 31-34
Figure 31-17 The Timing Events in the GSM Initialization and Ramping States 31-35
Figure 31-18 PAC Transmit Burst Timing 31-36
Figure 31-19 PAC Loop Including Element Timing Delays 31-38
Figure 31-20 PAC System Parameters 31-40
Figure 32-1 Generic Foldback Regulator Block Diagram 32-2
Figure 32-2 Block Diagram of Core Regulator Showing Signal Pin 32-7
Figure 32-3 Block Diagram of TX,RX,PAC Regulator Showing Signal Pins 32-9
Figure 32-4 Block Diagram of Codec Regulator Showing Signal Pins 32-11
Figure 32-5 Block Diagram of Synthesizer Regulator Showing Signal Pins 32-12
Figure 33-1 TUNEC Black Box Diagram 33-2
Figure 33-2 Functional Block Diagram 33-7
Figure 33-3 Integrator Timing Diagram 33-8
Figure 33-4 Timing Diagram 33-9
Figure 34-1 DPLL pins 34-5
Figure 34-2 DPLL Functional Block Diagram 34-7
Figure 34-3 Digital Phase-Frequency Detector 34-8
Figure 34-4 Phase Adjusting Block 34-9
Figure 34-5 BRM structure 34-9
Figure 34-6 Self Calibration scheme 34-11
Figure 34-7 The DPLL linear model for FOL mode 34-12
Figure 34-8 The DPLL transfer functions for input phase and VCO noise 34-13
Figure 34-9 The DPLL linear model for FPL mode 34-14
Figure 34-10 The DPLL transfer functions in the FPL mode 34-15
Figure 34-11 The timing diagrams for DPLL stop 34-16
Figure 34-12 The timing diagrams for DPLL start 34-17
Figure 34-13 The timing diagrams for the CMFN 34-19
Figure 34-14 The clock gating timing diagrams 34-20
Figure 34-15 The timing diagrams for the DPPDSYN signal 34-20
Figure 35-1 GPADC Block Diagram 35-2
Figure 35-2 GPADC Normal Mode Operation 35-9
Figure 35-3 GPADC Deep Sleep Mode Operation 35-9

Figure 35-4 GPADC Test Mode Operation 35-10
Figure 36-1 A2DIGL Block Diagram 36-3
Figure 36-2 RC Tuning Element Control Interface and Timing 36-16
Figure 36-3 Receive and Transmit modules control interface 36-24
Figure 36-4 A2DIGL SPI input muxing. 36-25
Figure 36-5 A2DIGL timing control signals muxing. 36-26
Figure 36-6 A2DIGL SSI muxing 36-26
Figure 36-7 Timing Diagram 36-28
Figure 36-8 Timing Diagram 36-28
Figure 36-9 Receive Timing Sequence for VLIF mode, AB_SEL=1, ADAPT_EN=1 . . . 36-29
Figure 36-10 Receive Timing Sequence for VLIF mode, AB_SEL=1, ADAPT_EN=1 without a SPI write at the end of the burst36-31
Figure 36-11 Receive Timing Sequence for VLIF mode, AB_SEL=1, ADAPT_EN=0 . . . 36-32
Figure 36-12 Receive Timing Sequence for VLIF mode, AB_SEL=0, ADAPT_EN=1 . . . 36-33
Figure 36-13 Receive Timing Sequence for VLIF mode, GPRS FCH scan prior to Serving Cell 36-34
Figure 36-14 DC Adapt and SPI Mux Timing Detail for VLIF mode. 36-36
Figure 36-15 Receive Timing Sequence for DCR mode, ADAPT_EN=1, AB_SEL=1 . . . 36-37
Figure 36-16 DC Adapt and SPI Mux Timing Detail for DCR mode 36-39
Figure 36-17 Transmit Timing Sequence, ABR_SEL=0, ABF_SEL=1 36-40
Figure 36-18 Transmit Timing Sequence, ABR_SEL=0, ABF_SEL=0 36-42
Figure 36-19 Transmit Timing Sequence, ABR_SEL=1, ABF_SEL=1 36-43
Figure 36-20 Transmit Timing Sequence, ABR_SEL=1, ABF_SEL=0 36-44
Figure 36-21 General Purpose ADC Interface 36-74
Figure 36-22 Normal Mode Timing using ASC & with APWD = 1, PWD=0/1, DFI=1 . . 36-75
Figure 36-23 More Examples of ASC based conversions with APWD =1, PWD=0/1 36-76
Figure 36-24 Normal Mode Timing using CNVE & with APWD=1, PWD=0/1, DFI=1 . . 36-77
Figure 36-25 More Examples of CNVE based Conversion with APWD = 1, PWD=0/1 . . 36-78
Figure 36-26 Deep Sleep Mode Behavior when APWD = 1, PWD=0/1 36-79
Figure 36-27 Test Mode Behavior, with APWD =1, PWD = 0/1 36-80
Figure 36-28 Pseudo Synchronization from PAT_REF to CKIL clock domain 36-81
Figure 37-1 MDI Schematic Block Diagram. 37-4
Figure 37-2 Pin Diagram 37-5
Figure 37-3 MDI Memory Mapping in the DSP Side. 37-9
Figure 37-4 MCU Memory Configuration 37-10
Figure 37-5 MDI Register Mapping in the MCU Address Space. 37-11
Figure 37-6 MDI Memory Mapping in the MCU Side 37-12
Figure 37-7 MDI Registers. 37-14
Figure 37-16 Messaging Model Using Transmit and Receive Registers. 37-41

Figure 37-17 Messaging Model Using a General Purpose Interrupt. 37-42
Figure 37-18 MDI read to shared memory timing diagram 37-50
Figure 37-19 MDI write to shared memory timing diagram 37-51
Figure 38-1 ARM7 Platform Block Diagram. 38-4
Figure 38-2 ARM7_PLATFORM Bus Structure 38-6
Figure 38-3 HREADY Typical Case 38-9
Figure 38-4 HREADY - Show Cycle Delay Case 38-9
Figure 38-5 . . . . . .Overall Block Diagram38-12
Figure 38-6 Block Diagram of Select Operation in AHBMUX. 38-15
Figure 38-7 RAM BIST Block Diagram (One Configuration) 38-19
Figure 38-8 ROM BIST Block Diagram (One Configuration) 38-20
Figure 38-9 2M ROM Protection Example in Internal Boot Mode (Normal) 38-23
Figure 38-10 4M ROM Protection Example in Internal Boot Mode (Normal) 38-24
Figure 38-11 2M ROM Protection Example in External Boot Mode 38-25
Figure 38-12 4M ROM Protection Example in External Boot Mode 38-26
Figure 38-13 R-AHB Bus Master Mux 38-35
Figure 38-14 DBGRQ and DBGEN Interconnections 38-40
Figure 38-15 R-AHB Slave Reset Timing 38-49
Figure 38-16 R-AHB Bus Timing Parameters 38-50
Figure 38-17 Alternate Bus Master Timing Parameters. 38-53
Figure 40-1 ARM7 Platform Core Arbiter Block Diagram 40-2
Figure 40-2 40-5
Figure 40-3 0 wait state bus handoff 40-10
Figure 40-4 0 wait state, with LOCK asserted bus handoff 40-11
Figure 40-5 1 wait state bus handoff 40-12
Figure 40-6 1 wait state, with LOCK asserted bus handoff 40-13
Figure 40-7 0 wait state, 2 word alternate master access 40-14
Figure 41-1 AIPI interface 41-3
Figure 41-2 Two clock read access (32-bit IPbus) 41-18
Figure 41-3 Three clock read access (32-bit IPbus) 41-19
Figure 41-4 Three clock read access (16-bit IPbus) 41-20
Figure 41-5 Five clock read access (8-bit IPbus) 41-21
Figure 41-6 Three clock write access (32-bit IPbus) 41-22
Figure 41-7 Three clock write access (16-bit/32-bit IPbus) 41-23
Figure 41-8 Four clock write access (16-bit IPbus) 41-24
Figure 41-9 Two clock read with error (16-bit/32-bit IPbus). 41-25
Figure 41-10 Three clock read with error (16-bit IPbus) 41-26
Figure 41-11 Four clock write with error (16-bit/32-bit IPbus) 41-27
Figure 41-12 AIPI access protection/occupation error. 41-28

Figure 41-13 AIPI IPbus v2.0 Bus Timing Parameters 41-29
Figure 42-1 External Arbiter Block Diagram. 42-3
Figure 42-2 AMARB State Diagram 42-18
Figure 42-3 AMARB I/O Block Diagram 42-19
Figure 42-4 Alternate Master Access from Port 0 42-20
Figure 42-5 Simultaneous Requests from Port 0 and Port 1 42-21
Figure 42-6 Wait Mode Entry and Exit 42-22
Figure 42-7 Alternate Master Access During Wait Mode 42-23
Figure 42-8 Bus Handoff Interval Control 42-24
Figure 42-9 Neptune Bus Arbitration (0 wait states, 2 word access) 42-27
Figure 42-10 Neptune Bus Arbitration (1 wait-state, 2 word access) 42-28
Figure 43-1 AITC Block Diagram 43-2
Figure 43-2 Example Memory Map for Opcode Patching With High AITC Vectors 43-4
Figure 43-3 AITC Vector Table Located In High Memory 43-4
Figure 44-1 AWPT Block Diagram 44-2
Figure 44-2 Example Memory Map For Opcode Patching With High AITC Vectors 44-3
Figure 44-3 AWPT Vector Table Details in PC Relative Mode With OFFSET=64 44-4
Figure 44-4 AWPT Vector Table Details in SP Relative Mode. 44-4
Figure 45-1 Neptune Memory Complex Block Diagram. 45-2
Figure 45-2 Memory Controller Block Diagram 45-4
Figure 45-3 Read Write Waveform 45-9
Figure 45-4 Neptune Large RAM Block signals 45-18
Figure 45-5 Read/Write Waveforms 45-20
Figure 45-6 Neptune Large ROM Block signals 45-22
Figure 46-1 Block diagram. 46-3
Figure 47-1 IIM Block Diagram. 47-3
Figure 48-1 AEIM Block Diagram. 48-2
Figure 48-2 Example of AEIM Interface to Memory and Peripherals. 48-5
Figure 48-3 Example for AEIM Interface to Burst Memory 48-6
Figure 48-4 Async mode reads, 1 wait state per external access 48-41
Figure 48-5 Async mode reads, 1 wait state per external access; 0 wait state internal access. . . 48-42
Figure 48-6 Async mode writes, 1 wait state per external access; 0 wait state internal access . . 48-43
Figure 48-7 48-44
Figure 48-8 Async mode read (CS0) followed by write of CS0, 1 wait state per external access; 0 wait state internal access48-45
Figure 48-9 Async mode read of CS0 followed by write to CS1, 1 wait state per external access; 0 wait state internal access48-46

Figure 48-10 Async mode write to CS0 followed by read of CS0, 1 wait state per external access; unknown wait state internal access48-47
Figure 48-11 Async mode write to CS0 followed by read of CS1, 1 wait state per external access; unknown wait state internal access48-48
Figure 48-12 Async mode read of CS0 followed by write to CS1, 0 wait states per external access; 0 wait state internal access48-49
Figure 48-13 Async mode reads, effects of CSIDLE field. 48-50
Figure 48-14 Async mode reads w/ slow OE -> data hi Z, effects of EDEAD field 48-51
Figure 48-15 Async mode reads, effects of OEASS field 48-52
Figure 48-16 Burst Mode Reads (32-bit requests to 16-bit port) 48-54
Figure 48-17 Pulse Mode Burst Reads (32-bit requests, 16-bit port). 48-55
Figure 49-1 Interface Definition. 49-4
Figure 49-2 Top Level MQSPI Architecture 49-7
Figure 49-3 SPI Block Diagram. 49-8
Figure 49-4 MQSPI Memory Map 49-9
Figure 49-5 SPI timing CPOL=1 49-11
Figure 49-6 SPI timing CPOL =0. 49-12
Figure 49-7 Timing Path Limits max MQSPI Receive Frequency 49-14
Figure 49-8 Receive Operation Timing Diagram. 49-16
Figure 49-9 Serial Display Interface (SDI) Function (Hip6w version) 49-20
Figure 49-10 MQSPI Memory Map 49-26
Figure 49-11 SPI timing CPOL=1 49-42
Figure 49-12 SPI timing CPOL =0. 49-43
Figure 50-1 Interface Definition. 50-6
Figure 50-2 Timing Block 50-14
Figure 50-3 Relation Between Reference Frame Timing (top) and Channel Frame Timing (bottom) for trqbcm[15:0] = 4999.50-17
Figure 50-4 Relation between RAFN and CFN when QBCO = 0. 50-18
Figure 50-5 Relation between RAFN and CFN when QBCO = 500. 50-19
Figure 50-6 Relation between RAFN and CFN when QBCO = 4500. 50-19
Figure 50-7 Event Generator 50-22
Figure 50-8 Relationship Between Absolute Event Quarter-Bit Counts (AEQBCs) and Relative Event Quarter-Bit Counts (REQBCs)50-24
Figure 50-9 Frame Table Control Unit. 50-26
Figure 50-10 Pin Control Unit 50-31
Figure 50-11 Example TCH Receive Timing Sequence 50-138
Figure 50-12 Example TCH Transmit Timing Sequence. 50-139
Figure 50-13 Example Power Measurement Timing Sequence 50-140
Figure 50-14 Example Alternate Power Measurement Timing Sequence 50-141
Figure 50-15 Example SV/ISU TDMA Frame Structure Timing 50-142

Figure 50-16 Expanded ISU TDMA Frame Structure Rollover 50-143
Figure 51-1 Top Level SIM Module Diagram 51-14
Figure 51-2 SIM Port Block Diagram 51-18
Figure 51-3 SIM Block Diagram 51-21
Figure 51-4 SIM Block Diagram 51-22
Figure 51-5 SIM Bus Interface Port Diagram 51-23
Figure 51-6 Register Organization Diagram 51-24
Figure 51-7 Register Bit Implementations Diagram 51-25
Figure 51-8 SIM Clock Generator Diagram. 51-26
Figure 51-9 Transmit State Machine Diagram. 51-29
Figure 51-10 Transmit Guard Time Diagram. 51-30
Figure 51-11 Transmit NACK Operation. 51-31
Figure 51-12 SIM Data Conventions 51-31
Figure 51-13 Receive State Machine Diagram. 51-33
Figure 51-14 Start Bit Diagram 51-34
Figure 51-15 Parity Bit Diagram 51-34
Figure 51-16 Framing Error Diagram 51-35
Figure 51-17 Valid Initial Characters. 51-35
Figure 51-18 Inverse Convention vs. Direct Convention. 51-36
Figure 51-19 SIM Card Hookup to Patriot SIM Ports 51-37
Figure 51-20 Automatic Power Down Sequence 51-38
Figure 51-21 Cyclic Redundancy Check Circuit Diagram. 51-40
Figure 51-22 Work Waiting / Character Wait Time. 51-41
Figure 51-23 Work Waiting Time 51-41
Figure 51-24 Block Guard Time 51-41
Figure 51-25 Block Wait Time. 51-42
Figure 51-26 ETU Counter State Diagram. 51-44
Figure 51-27 Suggested “T=1”, EMV, Geldkate Compliant SIM Initialization 51-93
Figure 52-1 Radio platform incorporating Neptune LTE, POWER IC, and RF IC. 52-3
Figure 52-2 Deep Sleep Module Ports 52-5
Figure 52-3 Deep Sleep Module: Top-Level Architecture. 52-9
Figure 52-4 Relationship between Delta-Time parameters, Absolute-Time parameters, and the DSM state52-12
Figure 52-5 DSM state transition diagram 52-13
Figure 52-6 SLEEP Sequence: Typical drift measurement and Deep Sleep preparation sequence. Timing is not to scale.52-16
Figure 52-7 DSM entering Deep Sleep and gearshifting PAT_REF from ccm_dsm_ckih_clk frequency to CKIL frequency52-19
Figure 52-8 Wakeup due to external interrupt for interrupt response modes 1 and 2. 52-21

Figure 52-9 WAKEUP sequence with wakeup due to counter timeout. Timing not to scale . . . 52-22
Figure 52-10 WAKEUP Sequence: Restarting the Layer 1 Timer 52-25
Figure 52-11 Modification to sleep sequence when MAGIC bugfix is enabled 52-54
Figure 52-12 52-58
Figure 53-1 Low Power Control. 53-5
Figure 53-2 53-6
Figure 53-3 Counter State Machine 53-7
Figure 53-4 WDOG and wdog_rst_b Operation 53-8
Figure 53-5 Cmon_noclk 53-9
Figure 54-1 Top Level GEM Diagram 54-3
Figure 54-2 Pictorial View of GPRS Ciphering 54-5
Figure 54-3 GPRS Execution Flow 54-7
Figure 54-4 Method of Addressing a GPRS LLC Frame Stored in RAM 54-11
Figure 54-5 Illustration of Time when Non-Maximum Sized Access will be Done 54-13
Figure 54-6 Buffer descriptor format 54-15
Figure 55-1 EGPT Clock Scheme 55-3
Figure 55-2 Timers Block Diagram 55-4
Figure 55-3 PWM Block Diagram 55-5
Figure 55-4 EPIT Block Diagram 55-5
Figure 55-5 Starting a Count from an Off State 55-8
Figure 55-6 Counter Reloading from the Modulus Latch 55-8
Figure 55-7 Counter in Free-Running Mode 55-9
Figure 56-1 RTR Block Diagram 56-3
Figure 56-2 RTR Block Diagram Detail 56-6
Figure 56-3 RTR Read Cycle, Zero Module-inserted Wait States. 56-6
Figure 57-1 RTC Block Diagram 57-2
Figure 58-1 KPP Peripheral Block Diagram 58-2
Figure 58-2 58-5
Figure 58-3 Keypress Synchronizer Functional Diagram 58-12
Figure 58-4 Decoding Wrong Three Key presses 58-13
Figure 59-1 DMAC System Diagram. 59-2
Figure 59-2 Sample LCD Controller Memory Mapping (Monochrome). 59-7
Figure 59-3 DMAC LCD Controller Memory Mapping (2-bit color/gray scale) 59-8
Figure 59-4 Memory Configuration for Automatic Display Data Transfers 59-9
Figure 59-5 Command Buffer Tag Organization 59-10
Figure 59-6 DMAC Automatic Mode Write Sequence (monochrome display) 59-11
Figure 59-7 DMAC Automatic Mode Data Flow (AUTOMODE[1:0] = 10) 59-12
Figure 59-8 DMAC Serial Transfers to LCD Device. 59-34

Figure 59-9 DMAC Parallel Transfers to LCD Device 59-36
Figure 60-1 Top Level HAC Diagram 60-2
Figure 60-2 Padding value creation in HAC 60-3
Figure 60-3 Parameters used to identify a block of FLASH to be hashed 60-4
Figure 61-1 DRA Block Diagram 61-2
Figure 62-1 MSU Block Diagram 62-3
Figure 62-2 MSU Comparator (one of eight) 62-4
Figure 62-3 Example Memory Configuration 62-19
Figure 63-1 63-2
Figure 63-2 63-10
Figure 63-3 Secure RAM Block Diagram 63-10
Figure 63-4 Laser ID Block Diagram 63-29
Figure 63-5 High Level Block Diagram of Security Monitor with Host Processor 63-34
Figure 63-6 Security Monitor Block Diagram 63-35
Figure 63-7 Security Monitor Detailed Diagram 63-36
Figure 63-8 Secure State Controller Block Diagram 63-38
Figure 63-9 Secure State Controller State Diagram 63-39
Figure 63-10 Security Policy Block Diagram 63-40
Figure 63-11 Algorithm Integrity Checker Block Diagram 63-42
Figure 63-12 ASC Block Diagram 63-43
Figure 63-13 Bit Bank Block Diagram 63-45
Figure 63-14 PT/CT Comparator Block Diagram 63-46
Figure 63-15 Timer Block Diagram 63-48
Figure 63-16 Debug Detector Block Diagram 63-48
Figure 64-1 SLID Block Diagram 64-2
Figure 65-1 MDPI Module Block Diagram (exclude X/Y DPR) 65-2
Figure 65-2 hresp0 and hready Block Diagram 65-6
Figure 65-3 MDPI Multiplexer Block Diagram 65-7
Figure 65-4 MDPI_BISM Timing Diagram (32-bit write) 65-8
Figure 65-5 MDPI_BISM Timing Diagram (32-bit read w/0 wait states) 65-9
Figure 65-6 MDPI_BISM Timing Diagram (16-bit write w/0 wait states) 65-10
Figure 65-7 MDPI_BISM Timing Diagram (16-bit read w/0 wait states) 65-11
Figure 65-8 MDPI_BISM Timing Diagram (out of bounds tea_b generation) 65-12
Figure 65-9 MDPI_BISM Timing Diagram (byte access error - hresp0 generation) 65-13
Figure 66-1 One-Wire Connection 66-2
Figure 66-2 66-2
Figure 66-3 1-Wire Initialization 66-4
Figure 66-4 Write 0 Timing 66-5
Figure 66-5 Write 1 Timing 66-5

Figure 66-6 Read Timing 66-6
Figure 66-7 Reset and Presence Pulses 66-11
Figure 66-8 Write 0 66-12
Figure 66-9 Write 1 66-13
Figure 67-1 Test Control Module (TCM). 67-4
Figure 67-2 TCM Simplified Block Diagram 67-5
Figure 67-3 Alternate Master Test (AMT) Mode Block Diagram 67-32
Figure 67-4 **Bus Arbitration BR-> BG. 67-33**
Figure 67-5 Pipelined, 2 wait state memory access (PIG), 16-bit mode 67-35
Figure 67-6 Pipelined, 2 wait state memory access (PIG), 32-bit mode 67-36
Figure 67-7 Pipelined, 0 wait state memory access, 16-bit mode 67-37
Figure 67-8 Pipelined, 0 wait state memory access, 32-bit mode 67-38
Figure 68-1 LTE Scan Test Controller Block Diagram 68-3
Figure 68-2 high level scan input router. 68-12
Figure 68-3 High level scan output router 68-13
Figure 68-4 Reset signals's muxing for scan modes 68-22
Figure 69-1 Block diagram of INT module 69-2
Figure 69-2 External Interrupt / GPIO Block Diagram (each pin). 69-4
Figure 70-1 BIST Block Diagram 70-2
Figure 70-2 BIST Execution Controlled by ipt_test_membist_invoke in Production Test Mode 70-6
Figure 70-3 Example System Level Integration of Multiple BIST Modules 70-15
Figure 71-1 Neptune JTAG Block Diagram. 71-2
Figure 71-2 ARM7TDMI-S synchronization circuit 71-8
Figure 71-3 Onyx (ONYX configuration) 71-9
Figure 71-4 Onyx (Concatenation mode). 71-10
Figure 71-5 ARM Mode Configuration 71-11
Figure 71-6 ONYXU JTAG Block Diagram 71-12
Figure 71-7 TAP Controller State Machine 71-14
Figure 71-8 Instruction Register Capture Value. 71-15
Figure 71-9 Identification Register Configuration 71-16
Figure 71-10 Bypass Register. 71-18
Figure 71-11 ARM and ONYX Latch and sync. 71-21
Figure 71-12 ARM-ONYX TDI-TDO Waveform 71-26
Figure 72-1 Anatest Block Diagram. 72-2
Figure 72-2 MUX Diagram 72-4
Figure 72-3 MUX2 72-4
Figure 73-1 Schematic Diagram of the I/O Cell. 73-3
Figure 73-2 Output Enable Selector Circuit. 73-8

Figure 73-3 Output Enable Selector Circuit for an Active High Output Enable 73-9
Figure 73-4 Output Enable Selector Circuit for an Active Low Output Enable 73-9
Figure 73-5 Input Type Selector Circuit. 73-10
Figure 73-6 Chip Integration Level Connections to the Input Type Selector Circuit for CMOS Input.73-10
Figure 73-7 Chip Integration Level Connections to the Input Type Selector Circuit for Hysteresis Input.73-11
Figure 74-1 Low Noise ESD NEtwork for HiP7 Neptune Pass 2 74-2
Figure 74-2 Low Noise ESD Network for HiP7 Neptune Pass 2. 74-3
Figure 74-3 HiP7 Neptune Pass 2 ESD Network for Analog Segments 74-4
Figure 74-4 HiP7 ESD Network for the 5V VDD_CP Segment 74-5
Figure 74-5 Large Rail Clamp Circuit 74-6
Figure 74-6 HiP7 Stacked DGO 5V Rail Clamp Circuit 74-7
Figure 74-7 Small Rail Clamp Circuit 74-8
Figure 74-8 Large Rail Clamp Circuit 74-9
Figure 74-9 Diode Networks 74-10
Figure 75-1 Power Up Sequence 75-6
Figure 75-2 Reset Timing. 75-14
Figure 75-3 DSP External Interrupt Timing (Negative Edge-Triggered) 75-14
Figure 75-4 NT0-INT7 External Interrupt Timing. 75-15
Figure 75-5 AEIM External Bus Timing relative to M_CLK 75-17
Figure 75-6 AEIM External Bus Timing Relative to CKO 75-21
Figure 75-7 Subscriber Interface Module Power Down AC Timing 75-22
Figure 75-8 Subscriber Interface Data Format 75-23
Figure 75-9 MQSPI Timing Specifications (CPOL=0, DOPH=0). 75-24
Figure 75-10 MQSPI Timing Specifications (CPOL=0, DOPH=1). 75-24
Figure 75-11 *MQSPI Timing Specifications (CPOL=1, DOPH=0). 75-25
Figure 75-12 MQSPI Timing Specifications (CPOL=1, DOPH=1). 75-25
Figure 75-13 USB Clocking 75-26
Figure 75-14 USB Input Timings. 75-27
Figure 75-15 BBP/SAP Transmitter Timing 75-29
Figure 75-16 BBP/SAP Receiver Timing 75-30
Figure 75-17 Test Clock Input Timing Diagram 75-36
Figure 75-18 HIGHZ (JTAG) Timing Diagram. 75-36
Figure 75-19 TRST Timing Diagram. 75-36
Figure 75-20 Test Access Port Timing Diagram 75-37
Figure 76-1 Mechanical Drawing MAPBGA 280 76-3
Figure 76-2 Mechanical Drawing MAPBGA 280, Notes 76-4
Figure A-1 Diagram of the UART Transmission A-5

Figure A-2 Format of Short Messages A-10
Figure A-3 Format of Long Messages A-12
Figure A-4 Format of memory_write.request Message A-14
Figure A-5 Format of memory_write.response Message A-14
Figure A-6 Format of message_read.request Message. A-15
Figure A-7 Format of message_read.response Message A-16
Figure A-8 Format of memory_check.request Message A-16
Figure A-9 Format of memory_check.response Message A-17
Figure A-10 Format of start_application.request message. A-17
Figure A-11 Format of invalid_opcode.response message A-18
Figure A-12 Format of invalid_opcode.response message A-18
Figure A-13 Format of invalid_opcode.response message A-18
Figure A-14 Format of invalid_opcode.response message A-18
Figure A-15 Mapping of DSP Program Memory Words to MDI Message Words A-19
Figure B-1 Example Neptune Configuration B-3
Figure B-2 Design Steps for Scan Insertion B-4
Figure B-3 Steps for DFT Team to Release to Test B-5
Figure B-4 General Configuration of sdi/sdo Connections. B-6
Figure B-5 Module Scan Ports for Legacy Modules. B-7
Figure B-6 Common Scan Test Bus for Peripherals B-8
Figure B-7 MCU Peripheral Bus Interface B-11
Figure B-8 . . . . . B-12
Figure B-9 Pig Bus Control During Scan Test B-14
Figure B-10 Basic Clock Tree Methodology for Legacy Modules. B-15
Figure B-11 Clock Tree with Clock Divider. B-16
Figure B-12 Clock Tree for Test Only Registers B-16
Figure B-13 Clock Gating Strategy for New Modules B-17
Figure B-14 Test Block/ Test Control Signals Distribution B-18
Figure B-15 External/Divided Clocks Routed to PMBIF B-19
Figure B-16 External/Divided Clocks Routed Internally B-19
Figure B-17 Internal Module Reset Logic B-20
Figure B-18 . . . . . B-20
Figure B-19 NEW SDO Output Convention for Neptune B-21
Figure B-20 . . . . . B-21
Figure B-21 Scan Enable Logic B-22
Figure B-22 Tri-State Logic B-22
Figure B-23 Scan Control Summary. B-24
Figure B-24 Pre-Scan Insertion Example B-25
Figure B-25 S-PMB Interface Clock Logic B-31

Figure B-26 ARM7 Platform B-32
Figure B-27 ARM7 Platform Test Interface Signals B-34
Figure B-28 ARM7 Platform B-35
Figure B-29 Analog Module Test Through Anatest B-37
Figure B-30 Mixed-Signal Module Test in General B-38
Figure 0-1 RAM BIST Interface B-39
Figure B-31 Example Scan Divergence Configuration. B-46
Figure B-32 Gated Clock B-48
Figure B-33 Gated Clock for Scan B-48

Chapter 1
Neptune LTE IC Summary (ICSum)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	04/24/02	Scott King, Shannon Osgood	Added block diagram from requirements document.
0.2	6 May 2002	Scott King	Updated module descriptions
0.3	18 June 2002	Scott King, Shannon Osgood	Added text to end of section 1.1.
0.4	13 Nov 2002	Scott King, Shannon Osgood	Added ULS/LTE information.
0.5	29 April 2003	Dan Pechonis, Shannon Osgood	Updated per DDTS# DSPhl16797 and DSPhl16796.
0.6	05/07/03		Updated for LTE specification release.

1.1 Overview

The Neptune LTE Baseband IC is a digital baseband processor for the 2.75G GSM market. The design is derived from Neptune LTS with changes to memory configuration and several module enhancements. It is a dual-core processor that contains a Synthesizable Onyx DSP core (56600), an ARM7TDMI-S microcontroller, and custom peripherals. Neptune LTE is configured for EDGE applications. Neptune LTE is available in a 280-pin MAPBGA package in either a production or development configuration. Please refer to Chapter 76, “Mechanical Specifications (PKG),” for more information. For Neptune LTE specific pin assignments see Chapter 2, “Neptune LTE IC Pin Description (PINS),” for more information.

1.2 Neptune LTE IC Features

The Neptune LTE IC is derived from the Neptune LTS IC with the following key changes:

- On-chip memory:
  - MCU RAM size is 256 kbytes.
  - MCU ROM size is 1792 kbytes.
  - DSP X, Y, and P RAM and ROM sizes:
    - 127 k x 24 bits PROM
    - 63.75 k x 24 bits PRAM
    - 40 k x 16 bits XROM
    - 20 k x 16 bits XRAM
    - 40 k x 16 bits YROM
    - 20 k x 16 bits YRAM

1.3 Neptune LTE IC Module List

- **DSP Core**
  - DSP56600 S-ONYXU Core
  - 130MHz
  - OnCE Debug
- **DSP Memory (DSPMEM)**
  - 127K x 24 Bits Program ROM
  - 63.75K x 24 Bits Program RAM
  - 40K x 16 Bits XROM
  - 20K x 16 XRAM
  - 40K x 16 Bits YROM
  - 20K x 16 YRAM
- **DSP Peripherals**
  - (LEM) Layer 1 Encryption
  - (BBP) Baseband Interface Port
  - (DTIMER) DSP Timer Module
  - (SAP) Serial Audio Port
  - (DMA) Single Channel DMA module x2
  - (VIAC) Viterbi Accelerator
  - (PDDM) DSP Debug Module
  - (DSIH) DSP Interrupt Handler
  - (VOCOD) Voice Codec
  - (DCLKG) DSP Clock Generator
  - (DMA_ARB) DMA Arbitration Module
  - (DRA) Data RAM Arbiter Module
- **Mixed Signal Modules**
  - (CROSC) Crystal Oscillator
  - (RXCPROC) Receiver Coprocessor
  - (RXAFE) Receiver Analog Front End
  - (TX) Transmitter
  - (DCADAPT) Fast Digital DC Adapt
  - (TRSYNT) Rx/Tx Synthesizer
  - (PAC) Power Amplifier Control
  - (REGUL) Regulators x 4
  - (TUNEC) Tuning Circuit
  - (RxSDG) Receiver Saturation Detection and SDM Dither Generation
  - (REFPLL) Reference PLL
  - (GPADC) General Purpose ADC
  - (ANATEST) Analog Test Interface
  - (POR) Power On Reset
  - (SJC) System JTAG Controller
- **Special Cells**
  - (CMON) Clock Monitors
  - (IIM) IC Identification Module
- **Microcontroller Core**
  - ARM7TDMI-S
  - 52MHz
  - ICE
- **MCU Memory**
  - 448K x 32 Bits ROM
    64K x 32 Bits RAM
- **MCU Peripherals**
  - (CARB) ARM7 Bus Arbiter
  - (AMARB) Alternate Master Arbiter
  - (AEIM) External Interface Module
  - (MDPI) MCU Dual-Port RAM Interface
  - (AIPI) ARM IPBus Interface
  - (MQSPI) Multiple Queue SPI
  - (AITC) ARM7TDMI-S Interrupt Controller
  - (SIM) Subscriber Interface Module
  - (UART) Universal Asynchronous Receiver Transmitter x2
  - (AWPT) MCU Watchpoint
  - (DSM) Deep Sleep Module
  - (WDOG) Watchdog Timer
  - (GEM) GPRS Encryption Module
  - (DMAC) Display Memory Access Controller
  - (EGPT) Enhanced General Purpose Timer
  - (RTR) Real Time Reference
  - (RTC) Real Time Clock
  - (CCM) Clock Control Module
  - (TCM) Test Control Module
  - (KPP) Keypad Port
  - (INT) External Interrupt Module
  - (DPLL) Digital Phase Lock Loop for USB
  - (A2DIGL) Analog to Digital Interface
  - (MCTL) MCU Memory Controller
  - (SCC) Security Controller
  - (MSU) Memory Separation Unit
  - (HAC) Hash Accelerator
  - (OWIRE) One Wire Interface
- **MCU/DSP Shared Peripherals**
  - (USB) USB w/ dedicated DPLL
  - (GPIO) General Purpose I/O
  - (MDI) MCU-DSP Interface
  - (L1T) Layer 1 Timer

1.4 Neptune LTE Functional Summary

The following sections describe the functionality and performance of Neptune LTE IC.

1.4.1 DSP

The DSP56600 S-ONYXU core in Neptune LTE operates at a maximum frequency of 130MHz. This performance will be utilized to support **General Packet Radio Service (GPRS)**, High Speed Circuit Switched Data (HSCSD), extended Voice Annotation (VA), Voice Recognition (VR), Enhanced Data Rate for GSM Evolution (EDGE), and other features.

1.4.2 DSP Memory

The Table below summarizes the DSP memory requirements for the Neptune LTE IC. The DSP memory architecture is a departure from Neptune LT. To achieve improved performance, efficiency and to leverage memory compiler technology the DSP memory architecture is simplified. The X and Y spaces are identical.

The Data RAM no longer has dedicated ports for all the shared memory peripherals. Instead the Data RAM has just 2 ports. One port is for priority access of the DSP core. The second port allows a peripheral to access the Data RAM with reduced priority. The Data RAM is interleaved in a 1 of 8 scheme. This guarantees that peripherals attempting access to Data RAM will complete the access as the DSP Core cycles through the 8 Data RAM blocks.

If more than one peripheral needs access to the Data RAM then these accesses must be arbitrated external to the DSP memory system. In Neptune LTE, this arbitration is achieved with the Data RAM Arbiter (DRA). The DRA arbitrates access to the Y-Data RAM for the VIAC, DMA1 and DMA2 blocks. The MDI has sole access to the secondary X-Data RAM port.

The 1 of 8 interleave scheme works over the entire RAM space. This increases the probability that peripherals will collide with the DSP core on Data RAM transfers. This is of particular concern for MDI transfers. To guarantee a high speed channel for transfer of data between the DSP and MCU core a small true Dual Port RAM is provided in both the X and Y space. This allows single cycle access for both the DSP and MCU core. No hardware support for collisions between the cores is provided. This must be provided by the system software.

Table 1-1. Neptune LTE DSP Memory Requirements

DSP Memory	Size (K=1024)
Program ROM	127K x 24 Bits
Program RAM	63.75K x 24 Bits
X Data ROM	40K x 16 Bits
X Data RAM	19K x 16 Bits
Y Data ROM	40K x 16 Bits
Y Data RAM	19K x 16 Bits
X DP RAM	1K x 16 Bits
Y DP RAM	1K x 16 Bits

1.4.2.1 DSP BIST (DBIST)

DBIST is a module to perform self test on DSP memories. This testing is for both RAM and ROM's. RAM built-in self test (BIST) will test the memory arrays, address and data paths, and control logic around the memory arrays. The fault model is based on IFA-13 which covers all single simple faults: stuck-at, stuck open, transition, coupling, address decode, data paths, and control logic.

For ROM DBIST makes use of the MISR (Multiple input shift register). The memory is read and a signature is generated. This signature can be shifted out and compared with expected signature.

1.4.3 DSP Program Paging

The Synthesizable DSP56600 core can only address 64Kx24 of program space at a given time. Since the Neptune memory requirements exceed this limit, a paging scheme is necessary to expand the program memory. The paging scheme involves defining a common block of 32Kx24 shared across all pages from address 0 to address 0x7FFF. The remaining 32Kx24 of address space is broken into 8Kx24 blocks. Each 8Kx24 block can have up to eight pages. A page configuration register is required to select which of the 8Kx24 blocks is in the DSP56600 memory map at a given time.

1.4.4 DSP Peripherals

The functions of the DSP peripherals required for the Neptune IC are summarized below.

1.4.4.1 Baseband Port Module (BBP)

The BBP module is a full-duplex (56602 SSI) serial port that interfaces to the on-chip TX and RxCPROC modules. Capability has been built into the module to allow for multiple time slot receive and transmit sequences. The Base Band Port (BBP) provides a full-duplex serial port for serial communication with integrated analog baseband modules. The BBP consists of independent transmitter and receiver sections, receiver and transmitter counters and a common BBP clock generator

1.4.4.2 Layer1 Encryption Module (LEM)

The Layer1 Encryption module is GSM-specific module that performs the generation of the A5 receive and transmit ciphering sequences. Additionally, it provides acceleration for GSM FIRE code decoding and encoding.

1.4.4.3 DSP Timer Module (DTIMER)

The DSP DTIMER module is a general purpose timer with three separate timing channels with the associated triggers and interrupt enables.

1.4.4.4 Serial Audio Codec Port (SAP)

The SAP module, like the BBP, is based on a standard SSI (DSP56602 SSI) port, and is included for DAI testing, or, for development, communication with external serial audio codecs.A programmable BRM is provided to divide down the reference clock to the desired frequency when the module itself is required to source the serial clock.

1.4.4.5 Direct Memory Access Controller (DMA)

Two independent DMA channels are provided to support bi-directional transfers for the BBP and SAP modules. Several packing and unpacking modes are supported.

1.4.4.6 DMA Arbitration Module (DMA_ARB)

The DMA_ARB provides arbitration for DMA1 and DMA2 accesses to the SAP and BBP modules

1.4.4.7 Data RAM Arbiter (DRA)

The DRA provides arbitration for DMA1, DMA2, and VIAC accesses to the Y interleaved RAM.

1.4.4.8 Viterbi Accelerator (VIAC)

The VIAC is a Viterbi decoder accelerator that supports the GSM full and half rate channel coding schemes, that is, constraint length codes 5 and 7, and 1/2, 1/3, and special 1/6 rate codes. Additionally, it provides acceleration for adaptive channel equalization, also utilizing Viterbi methods.

1.4.4.9 DSP Special Interrupt Handler (DSIH)

Since the DSP56600 core is limited to eight peripheral interrupt signals, and Neptune has more than eight DSP peripheral interrupt sources, some of the interrupt signals must be OR'ed together. This function is handled by the DSP Special Interrupt Handler module. In cases when more than one of the OR'ed interrupts occurs at the same time, a fixed, hardware priority scheme is used by DSIH.

1.4.4.10 DSP On Chip Emulator Debug Module (OnCE)

The DSP on-chip emulation (OnCE) module provides a means of interacting with the DSP core and its peripherals non-intrusively so that a user may set breakpoints, examine registers, memory, or on-chip peripherals facilitating hardware/software development. The OnCE controller functionality is accessed through the JTAG port.

1.4.4.11 DSP Debug Module (PDDM)

The DSP Debug supplies watchpoint and externally visible trace as defined below.

X/Y Data Watchpoints:

- The data watchpoints provide one set of comparators for each of the X and Y address and data buses. The data watchpoints provide the capability to match any combination of partial data and data address range and R/W. Data watchpoint events are observable through a s/w selectable GPIO pin, a DSP interrupt, or a DSP debug event.

Program Address Watchpoints:

- The program address watchpoints provide one comparator for the DSP Program address and page select bits. The ability to match a program address range is not required. Program Address watchpoint events are observable through a s/w selectable GPIO pin, a DSP interrupt, or a DSP debug event.

Program Address Trace (PAT):

- The DSP program address bus, page select bits, and a valid strobe are available on the development package part through a set of GPIO pins.

1.4.4.12 Voice Codec (VOCOD)

Voice Codec consists of voice coding and voice decoding. The voice coding function takes human speech signal in voltage format then converts it to 8 KHz digital signal. The voice decoding function takes 8 kilo-samples per second digital signal then converts it to an audible voice signal in voltage format.

1.4.4.13 DSP Clock Generator (DCLKG)

The DSP Clock Generator module provides clocks for the DSP Core, the DSP Memories, and the DSP Peripherals.

1.4.5 Mixed Signal Modules

1.4.5.1 Receive Coprocessor (RxCPROC)

The RxCPROC includes the digital signal processing hardware required for the receive transceiver after the initial conversion done by the sigma-delta modulator. It can be configured to be used in the direct conversion mode or in the very low intermediate frequency mode. The RxCPROC decimates and filters the I and Q quadrature input signals and converts them to baseband if required. Processed signals are sent serially to the Base Band Port (BBP) to be further handled by the DSP and VIAC. The RxCPROC supports the GSM and EDGE standards.

1.4.5.2 Receiver Analog Front End (RxAFE)

RxAFE converts the baseband or very low IF received signal from analog to digital. RxAFE consists of two sub-blocks: the RX Anti-Aliasing Filter and 4th order sigma delta modulator Baseband Receive I/Q A/Ds.

1.4.5.3 Fast Digital DC Adapt (DCADAPT)

The Fast Digital DC Adapt circuit cancels out the inherent DC offsets at the input to the baseband amplifier on Algae to a level which guarantees a corresponding offset of within +/-28mV as seen at the inputs to the sigma-delta modulator further down the line on Neptune. This is done before each RX burst and is completed within 38us.

1.4.5.4 Receiver Saturation Detection and SDM Dither Generation (RxSDG)

The receiver overload detection circuit is a part of Automatic Gain Control (AGC) loop of the LCA receiver.The receiver overload detection circuit detects excessive input signal at the Neptune receiver front end. It uses amplitude threshold method to detect the overloading condition of the sigma-delta modulator. When the receiver overload condition is detected, the MCU will get interrupted by the DET_FLAG. The interrupt signals MCU to lower the receiver gain in the Algae IC.

The receiver sigma-delta modulator dither generation circuit provides a dither signal for the receiver front end so that the quantization error will be spectrally flat.

1.4.5.5 Transmitter (TX)

The Neptune transmitter module supports several different methods of transmission: GSM normal mode, GSM Dual Port mode, GSM predistortion mode, and DCS/PCS mode. This module has a digital section which directly modulates the synthesizer and an analog section that supports the dual-port mode.

1.4.5.6 Rx/Tx Synthesizer (TRSYNT)

This synthesizer is to interface with the RX and TX Synthesizer Feedback blocks on the ALGAE IC. The open collector outputs of Algae enter Neptune differentially on VCO1_ N and VCO1_P. The operating frequency range for both Rx/Tx operation is to be between 695MHz and 1GHz.

1.4.5.7 Power Amplifier Control (PAC)

The power control system will be digital AOC which will interface to PAC II power detector. The system can operate with PA gate/base power control as is typically used in linear systems.

1.4.5.8 Regulators (REGUL)

The regulator section of the Neptune Specification contains detailed information on the requirements for the 4 regulators that will be present on the IC. All regulators will be standard linear regulator designs. All regulator supplies (inputs) will be a +/-3% tolerance while that of the regulated supplies (outputs) will be +/-5% tolerance. The core regulator is intended to supply the Neptune digital core with a 1.575V regulated voltage. Two PSRR improvement regulators will power the Rx and Tx paths with a 2.4V regulated supply. Finally, the digital portions of the Tx and Rx synthesizer blocks will be powered by a 1.575V regulated supply.

合成器

1.4.5.9 Tuning Circuit (TUNEC)

The HiP7 digital process does not provide a precision resistor or capacitor. An RC constant tuning circuit (TUNEC) is provided to ensure the required precision of all integrated filters. The tuning scheme requires a precise reference clock in order to provide calibration based on an RC time constant measurement. This scheme is intended to reduce filter cutoff frequency variations to less than +/-10%.

1.4.5.10 Reference PLL (REFPLL)

The reference PLL is an integrated PLL for the Neptune IC. The PLL uses a 26 Mhz signal as an input reference. The PLL produces the clock signals for the MCU, DSP, and an accurate 13 Mhz reference that is corrected to the 13MHz base station system clock.

1.4.5.11 General Purpose ADC (GPADC)

The general purpose ADC (GPADC) will convert an analog signal into a 10-bit digital word. In normal operation, a mux in the GPADC will select a data from three inputs: A2D_DATA, CP_RX_DIV2, and CP_TX_DIV2. A2D_DATA is a signal coming from Seaweed. CP_RX_DIV2 and CP_TX_DIV2 are scaled down and buffered signals of Neptune Tx/Rx synthesizer charge pump outputs. In normal operation, GPADC takes four samples from A2D_DATA, and then it takes one sample each from CP_RX_DIV2 and CP_TX_DIV2. In deep sleep mode, GPADC only samples the data from A2D_DATA six times. 13 MHz clock is used in normal mode operation, and 32 KHz clock is used in deep sleep mode operation.

1.4.5.12 Crystal Oscillator (CROSC)

The reference oscillator provides a stable frequency reference for use by the radio. The oscillator will use a crystal at 26 MHz. Compensation for all types of frequency error in the crystal (temperature, make tolerance, and aging) will be accomplished by offsetting the fractional N division ratio in the synthesizer and the reference PLL.

1.4.5.13 Analog Test Interface (ANATEST)

ANATEST is a new module designed for the Neptune chip. This module provides a way to bring out the signals from the Analog blocks to two dedicated Analog output Pads for observability, debug and testing.

1.4.6 Special Cells

1.4.6.1 IC Identification Module (IIM)

The IIM consists of two functions: the Unique Identifier (UID) and the Hardware Revision Register (REV).

The Unique Identifier cell is a laser programmed register that is programmed to a unique value for each die produced. The Unique Identifier is a 128 bit wide value that is accessible from the PIG bus interface to the MCU.

The Hardware revision register is simply a 16-bit read-only register that is fixed in value. The value is changed in layout whenever a silicon change is performed. This provides the software with a knowledge of what hardware features are available in the current revision. The value in the REV register must be programmable in metal only to allow for metal mask changes to be detected in software. The register is readable only by the MCU.

For Neptune LTE, some of the UID fuses are used to enable or disable the security functions.

1.4.6.2 Clock Monitor (CMON)

The Clock monitor module contains a custom cell that detects when the system clock has not transitioned for a given period of time. The Clock Monitor will signal the Watchdog module to generate a Watchdog event when a clock edge is not detected for 150ns to 620ns at its input. The clock monitor operates on the output of the crystal oscillator module, not the REFPLL. The clock monitor will be disabled automatically in hardware when entering deep sleep (crystal oscillator shut off). The clock monitor will provide software the ability to disable it.

1.4.6.3 Power On Reset (POR)

The POR circuit will provide a reset upon Power Up to be used in certain circuits in the chip that need to be working during RESET. The output of this circuit will also be OR'ed in the chip Reset circuit to activate the chip reset during power up.

This circuit will have three main elements:

- POR - Power On Reset System that recognizes the power up.
- A ring oscillator to generate clocks during the POR period.
- A Counter to count ring oscillator cycles.

1.4.6.4 System JTAG Controller (SJC)

The Neptune IC has an integrated S-ONYXU DSP and ARM7TDMI-S MCU on the same die and includes two JTAG TAP controllers. The ARM MCU ICEBreaker TAP controller, the ONYXU DSP OnCE™ TAP controller, and the Neptune System JTAG TAP controller.

The external JTAG pins are connected to the Neptune System JTAG TAP controller and through JTAG instructions the user may select one of three emulation modules for programming. The user may activate each of the three emulation modules separately or any combination of them simultaneously.

The System JTAG Controller (SJC) will support modes that allow stand-alone ARM and ONYXU tools to work separately or simultaneously, if desired.

NOTE:

Neptune does not have JTAG boundary scan in the pads because the whole chipset is not IEEE1149 compliant.

1.4.7 ARM7 MCU

The ARM7TDMI-S is a member of the ARM family of general-purpose 32-bit microprocessors. The ARM family offers high performance for very low power consumption and gate count.

The ARM7 MCU core in Neptune operates at a frequency up to 52MHz. The increase in processor frequency will increase performance of critical routines located in internal memory. Examples of these routines include: Virtual DMA, V.42bis compression, GPRS support, and the RTOS executive.

1.4.8 MCU Memory

Table 1-2. MCU Memory Configurations

MCU Memory	LTE Configuration (K=1024)
RAM	64K x 32 Bits
ROM	448K x 32 Bits

1.4.9 MCU Peripherals

The functions of the MCU peripherals for the Neptune IC are summarized below. Each peripheral provides the ability to disable the clock at the input to the module. Each peripheral provides a software reset capability. The software reset must reset the peripheral in the same way as a system reset. Neptune includes some clock control logic outside of the peripheral blocks. The Clock Control Module handles this supplemental clock control.

1.4.9.1 ARM7 Platform Core Arbiter (CARB)

The ARM7 core does not support alternate bus masters. The CARB module is added to support alternate masters. In Neptune, it interfaces exclusively to the AMARB, the Alternate Master Arbiter. All modules that require bus access interface to the AMARB.

1.4.9.2 ARM External Interface Module (AEIM)

The External Interface Module (EIM) handles the interface to external devices , including generation of chip selects for external peripherals and memory. It contains the following features:

- Six Chip Selects for external devices, each covering a range of 32Mbytes
- Programmable Wait-State generator for each Chip Select
- Selectable Protection for each Chip Select
- Programmable Data Port Size for each Chip Select
- Control for external/internal Boot ROM device selection
- MCU Bus Monitor counter for all bus cycles
- Programmable general output capability for unused Chip Select outputs
- Show cycles to allow internal bus cycles to be externally monitored
- 20 dedicated General Purpose Outputs
- Synchronous Burst-mode support for most industry standards flash device

1.4.9.3 ARM IPBus Interface (AIPI)

The AIPI module interfaces the R-AHB to the internal IPBus.

1.4.9.4 Alternate Master Arbiter (AMARB)

The alternate master arbiter allows multiple alternate masters, such as the DMAC, GEM, HAC, or TCM, to gain control of the ARM7TDMI-S platform bus. The alternate master arbiter is also used during low power modes to gate the clock to the ARM7TDMI-S core.

1.4.9.5 Multiple Queue Serial Peripheral Interface (MQSPI)

The MQSPI provides two (2) independent QSPI channels which performs the serial programming operations to configure the RF subsystem and selected peripherals. It is designed to minimize the amount of MCU interaction by automating multiple and repetitive serial data transfers. The module has multiple queues to hold these transfers. Module transfers can be triggered by the Layer 1 Timer.

1.4.9.6 ARM7TDMI-S Interrupt Controller (AITC)

The ARM7TDMI-S Interrupt Controller collects interrupt requests from up to 64 multiple sources and provides an interface to the ARM7TDMI-S core for normal and fast interrupts. The AITC includes hardware acceleration of normal interrupt to quickly jump to the highest priority interrupt service routine, effectively providing a vectored interrupt mechanism.

1.4.9.7 SIM Interface Modifications (SIM)

The SIM module is a customized UART with additional features allowing for communication with SmartCards and conforming to the ISO 7816 specification. The module has a transmit buffer of 16 bytes. The receive buffer is 32 bytes.

For Neptune LTE, the T=1 counters have been redesigned to improve performance, overcome deficiencies that exist on the SIM hardware, and provide functionality required by ISO/IEC 7816-3.

1.4.9.8 Universal Asynchronous Receiver Transmitter (UART)

A UART module performs all normal operations associated with “start-stop” asynchronous communications. The UART transmit and receive buffer sizes are 32 bytes each. The UART modules will operate at 115.2Kbps, 460Kbps, and 920Kbps based on a 13MHz input reference clock.

For Neptune LTE, the UART buffers are expanded from 32 to 64 bytes.

1.4.9.9 Deep Sleep Module (DSM)

The Deep Sleep Module allows for optimal power savings in idle modes by allowing the Neptune IC to synchronize to the frame timing automatically even though the timing reference has been removed from the part during the idle time.

1.4.9.10 Watchdog Timer (WDOG)

The Watchdog Timer is used to protect against system failures by providing a means to escape from unexpected events or application errors. Once started, the timer must be serviced by the core on a periodic basis. If servicing does not occur, the module times out and asserts the reset signal.

1.4.9.11 General Packet Radio Service Encryption Module (GEM)

The GPRS Encryption Module (GEM) is a hardware accelerator designed to assist in the encryption/decryption of General Packet Radio Service (GPRS) data packets and generation/verification of Frame Check Sequence (FCS) information contained in GPRS data packets. The GEM is required to arbitrate with the MCU for direct access to memory (DMA). This allows the GEM access to any internal or external memory space, but interrupts all MCU activity until the GEM relinquishes the bus.

For Neptune LTE, the GEM is modified to include enhancements made to other versions of GEM. The modifications includes the ability to eliminate the need to process GPRS frames twice by GEM. Also included is the ability to split GPRS frames into segments to allow GPRS software to reserve less total RAM. This modifications are totally software compatible with the previous version.

1.4.9.12 Keypad Port (KPP)

The Keypad port is a module that is used for keypad matrix scanning.

1.4.9.13 Enhanced General Purpose Timer (EGPT)

The EGPT provides 3 output compare, 2 input capture, and one PWM channel. An Enhanced Periodic Interrupt Timer (EPIT) is included in the EGPT. The EPIT contains a 24-bit down-counter with programmable modulus that can generate an interrupt when the counter reaches zero.

1.4.9.14 Test Control Module (TCM)

The TCM module contains the logic that controls the various test features for the Neptune IC. Any feature that is not useful for debug in functional mode will be conditioned with the Neptune TEST[2:0] input pins. The following functions are provided by the TCM module:

- Test Mode Definitions
- AMT Mode
- Analog Signal observability
- Non Scan Test
- Analog module output selection and configuration
- Memory BIST for ARM7 Memories
- Registered input BIST signals from the ARM7 BIST engine

1.4.9.15 Scan Test Controller (STC)

The STC provides the following test modes and functions for scan test:

- Main scan domain test modes
  - — Arm7_platform scan test mode
  - — Sonyxu scan test mode
  - — Sea of Gates (SOG) scan test mode
  - — Dig_Radio scan mode
  - — Embedded memory test mode (0 and 1)
- Scan divergence mode
- IDDQ test mode
- Scan chain routing

1.4.9.16 MCU Watchpoint (AWPT)

The Watch Point Module is a 32-bit peripheral which can be used to either patch source code routines or fix data tables in the Read-Only Memory (ROM). The AWPT supports up to 64 addresses which can be patched where 4 are any location within memory and 60 are dedicated to ROM patching. Alternatively, 16 of the 64 locations can be used as a 1-word data fix. Alternatively, the 4 generic locations can be used to generate a break point event to the ARM7TDMI-S.

1.4.9.17 USB Digital Phase Locked Loop (REFPLL)

The USB Digital Phase Locked Loop (REFDPLL) is used to generate the 48Mhz frequency required for the USB module. Lock times on the DPLL module will be less than 50us. The DPLL must not require the use of an external capacitor.

1.4.9.18 Real Time Reference (RTR)

The RTR module is a simple 47-bit up counter that runs at the always present 32.768Khz input frequency. The counter is read-only, and is intended to provide software with a time reference to the last system reset event. The module is capable of generating a periodic interrupt whenever one of the lower 16 bits of the counter is set. This is intended to provide the ability to profile the operation of the system software.

1.4.9.19 Real Time Clock (RTC)

The real time clock (RTC) module consists of counters and registers used to maintain the day, time of day and alarm values. The RTC operates even when the phone is powered down and the alarm function can turn it on and alert the processor. If the phone is already on, the processor is interrupted. The RTC uses a frequency reference generated on the Seaweed using a low-cost crystal and inaccuracies in the frequency may be compensated for by trimming the modulus register value. The RTC module also provides power cut logic, and keep-alive memory that activates when a low battery condition is detected.

1.4.9.20 Display Memory Access Controller (DMAC)

The DMAC (display memory access controller) module is designed to transfer data from the display frame buffer in system memory to an external LCD display device. The DMAC can support 4-wire, 3-wire interface (serial), and 8 bit parallel transfers. The DMAC acts as a master on the system bus and transfers the data transparently, with minimal software intervention. Bus utilization of the DMAC is deterministic and controllable via the programmability of the DMAC and alternate master arbiter.

1.4.9.21 Clock Control Module (CCM)

The Clock Control Module handles all intermodule clock routing, the selection of multiple input clock sources for the cores and various peripherals, manage MCU low power modes (DOZE, STOP and WAIT) by disabling peripheral clocks and other clock related features. This module also includes the control logic for asserting soft and hard chip-level system resets.

For Neptune LTE, CCM supports a divide-by-5 from the REFPLL running at 260MHz to achieve a 52MHz 50% duty cycle MCU clock; a CKIL clock monitor has been added; and control of most DCLKG functions have been removed.

1.4.9.22 External Interrupt Module (INT)

This module provides control for five of the eight external interrupt sources. Each pin is individually configurable as a level-sensitive interrupt, an edge-detecting (rising, falling, or both) interrupt, or a general purpose I/O. Each pin has a dedicated interrupt line.

1.4.9.23 Analog to Digital Interface (A2DIGL)

The A2DIGL is a digital module intended for control of mixed signal modules. The A2DIGL is divided into two parts: One part contains SPI Interface control; while the other part contains the MCU-based GPADC digital control.The A2DIGL controls the following modules: REGUL, GPADC, TUNEC, PAC, TX, TRSYNT, CROSC, DCADAPT, RxSDG, RxAFE, and RXCPROC.

1.4.10 Shared Peripherals

Shared peripherals are peripherals that can be accessed by both the DSP and MCU cores. The functionality of these peripherals is detailed below.

1.4.10.1 Universal Serial Bus Module (USB)

The USB module provides the required buffering and protocol to communicate on the Universal Serial Bus. The module is provided with an access port for the DSP and for the MCU. The module will act as a USB device only. Host functionality will not be supported. All four types of USB data transfers are supported: control, isochronous, interrupt, and bulk. Endpoint 0 always functions as a bidirectional control endpoint with an eight (8) byte buffer.

For Neptune LTE, endpoint 1 and 2 buffer size are increased from 16 bytes to 32 bytes.

1.4.10.2 General Purpose I/O (GPIO)

The GPIO module is a stand alone module that serves as the all-in-one communication link between the outside world, the MCU module, and the DSP module. The GPIO module will be implemented as a stand-alone module. The GPIO module will provide the following functionality:

- Standard GPIO functionality
- Multiplexed output functionality
- Alternate input functionality
- Shared Access for DSP and MCU
- MCU, DSP interrupt capability
- Interrupt visibility

1.4.10.3 MCU / DSP Interface (MDI)

The MDI is the communication interface between the DSP and MCU cores. Through this module, each core can access shared memory, messaging registers, and flags. This module also allows each core to interrupt the other, monitor the low power state of the other core, and other useful functions.

1.4.10.4 Layer 1 Timer (L1T)

The Layer 1 Timer module allows for control of all the radio channel timings. Its main function is to unload the MCU from having to schedule events associated with the radio air interface. The timer provides the user a great deal of flexibility in scheduling and executing events using the programmable event tables and macros.

1.5 Block Diagram

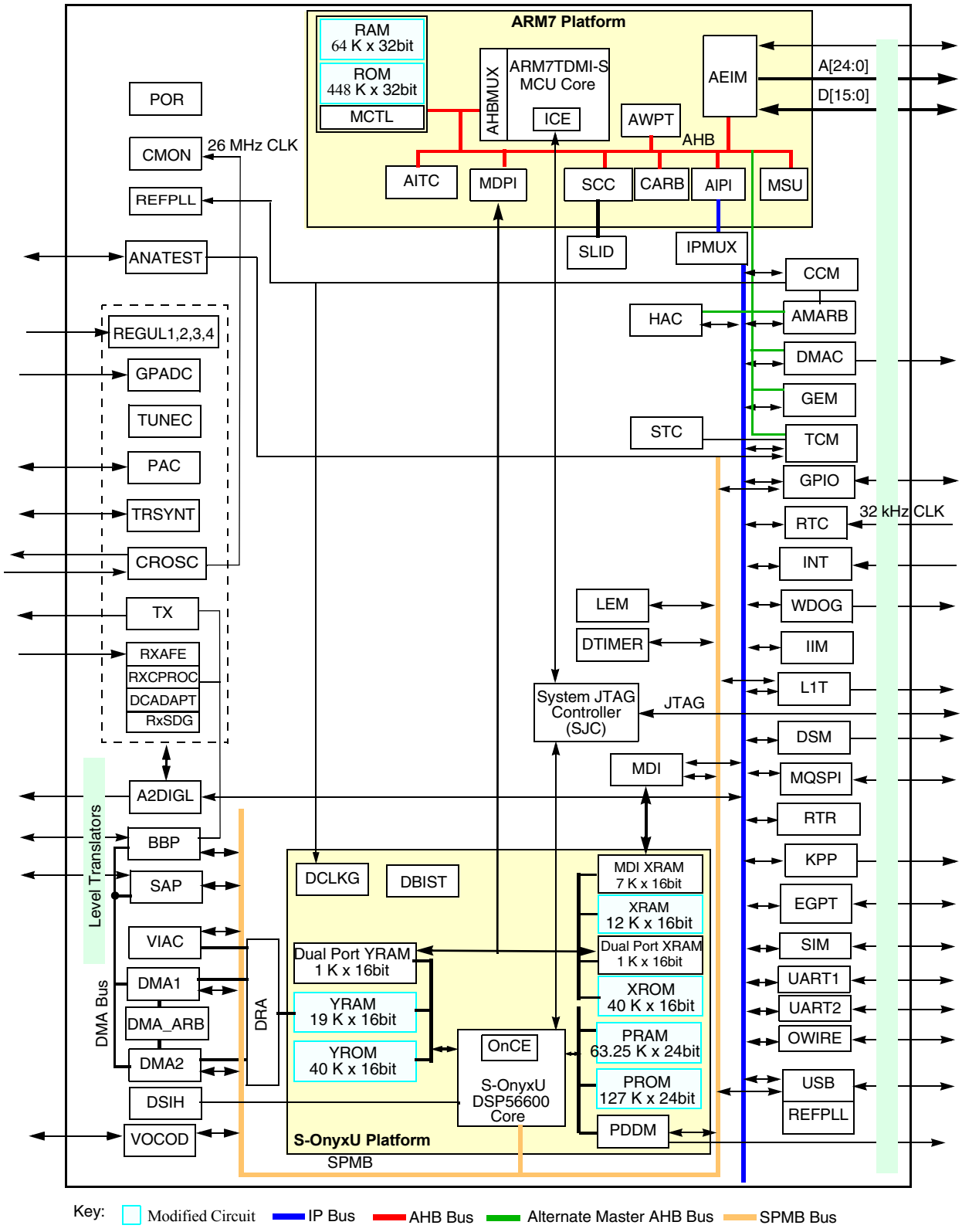


Figure 1-1. Neptune LTE Block Diagram

Chapter 2
Neptune LTE IC Pin Description (PINS)

Revision History Table

Revision	Date	Author	Changes
0.0	4/23/2002	Mark Babcock	Initial Draft for Neptune LTS initiated from Neptune LT version 2.5
0.1	5/3/2002	Mark Babcock	Addition of Signal Description paragraphs.
0.2	17 June 2002	Scott King	Updated 225pin package assignments
0.3	19 July 2002	Mark Babcock	Add 280-pin Production and Development Assignments.
0.31	22 July 2002	Mark Babcock	Modify CKOH drive strength to belong to Group 6.Updated per DDTS# DSPhl14905.
0.4	18 October 2002	Mark Babcock	Reformat Pad Drive description per DSPhl15137. Add No Connect Groupings for 280-pin production package per DSPhl15132. Clarify package versus RTL pin names for VSS_TXRX, CVDD, CVSS, VDD, VSS. Correct default reset column to output for PB10 (SIM_DIO), PC0 (CTS1-Alt), PE13 (CTS2), PE15 (RTS2) per DSPhl15194.
0.5	29 April 2003	Mark Babcock, Shannon Osgood	Updated per DDTS# DSPhl16057 and DSPhl15914.
0.6	05/07/03		Initial release of LTE only specification. Updated DDTS# DSPhl16791/DSPhl16789.

2.1 Pin Assignment Listing

Table 2-1 lists all pins defined for Neptune LTE.

Column “Pad #” indicates the sequential pad ring number for each pin function.

Columns “Dev(elopment) Pin”, “High Prod(uction) Pin”, and “Low Prod(uction) Pin” indicate the package pin assignment for that particular configuration.

Column “GPIO” lists the GPIO Port pin, if any, where the “Pad Name” is the primary (default reset) function.

Column “Std Pad Drive” lists the typical (minimum) drive current required for the pin.

Column “Pad Drive Group” lists the High (8 mA) Drive group programming assignment and the Super High (16 mA) Drive group programming assignments, if applicable. The first value indicates the control bit for the High Drive group; the second for the Super High Drive group.

Column “Pad Default” lists the pad drive strength selected after RESET and execution of initial boot code.

Column “Typical Voltage” lists the expected typical supply voltage of the Supply Power Group.

Column “Pwr Grp” lists the Supply Power Group assignment.

Column “RST State” lists the pin input/output direction at chip RESET.

Column “Comments; Alternate Functions” lists each of the GPIO Port alternate input and output selections available by programming the GPIO control registers. Some selections are test or development mode specific and may not be available for user application. Refer to Chapter 23, “General Purpose Input/Output (GPIO),” and Section 2.5, “GPIO Port Mux Assignments,” on page 2-59 for identification of test or development only selections.

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
AEIM																	
71		H8		D31	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
75		V18		D30	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
78		J8		D29	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
80		K8		D28	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
88		L8		D27	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
90		M8		D26	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
92		P17		D25	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
94		M9		D24	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
122		G16		D23	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
124		H12		D22	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
126		G13		D21	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
128		H11		D20	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
136		B19		D19	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
138		H10		D18	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
140		H9		D17	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
142		B18		D16	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
9	N8	N8		D15	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
7	V4	V4		D14	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
5	T4	T4		D13	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
3	N7	N7		D12	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
273	P4	P4		D11	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
276	T2	T2		D10	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
279	T3	T3		D9	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
282	U3	U3		D8	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
8	W4	W4		D7	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
6	U4	U4		D6	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
4	W3	W3		D5	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
2	V3	V3		D4	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
274	T1	T1		D3	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
277	U1	U1		D2	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
280	U2	U2		D1	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
283	M7	M7		D0	External Data Bus	I/O	26 MHz	69k PU	4 mA			GPO[1]; GPO[11]	8 mA	1.875 V	AVDD	I	
10	T5	T5		A24	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
272	M3	M3		A23	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
275	R3	R3		A22	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
278	N4	N4		A21	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
281	L7	L7		A20	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
121	J17	J17		A19	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
119	K18	K18		A18	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
95	R18	R18		A17	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
125	J18	J18		A16	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
143	E19	E19		A15	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
141	F16	F16		A14	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
139	F17	F17		A13	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
137	F18	F18		A12	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
135	F19	F19		A11	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
133	G19	G19		A10	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
129	H19	H19		A9	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
127	J16	J16		A8	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
93	P19	P19		A7	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
91	R17	R17		A6	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
89	M16	M16		A5	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
87	R19	R19		A4	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
85	T17	T17		A3	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
81	U19	U19		A2	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
79	U18	U18		A1	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
76	U17	U17		A0	External Address Bus	O	26 MHz		4 mA			GPO[0]; GPO[10]	8 mA	1.875 V	AVDD	O	
123	J19	J19		R/W*	Read/Write	O	26 MHz		4 mA			GPO[3]; GPO[13]	8 mA	1.875 V	AVDD	O	

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
69	M13	M13		CS5	Chip Select	O	26 MHz		4 mA			GPO[2]; GPO[12]	8 mA	1.875 V	AVDD	O	
132	G18	G18		CS4*	Chip Select	O	26 MHz		4 mA			GPO[2]; GPO[12]	8 mA	1.875 V	AVDD	O	
118	K17	K17		CS3*	Chip Select	O	26 MHz		4 mA			GPO[2]; GPO[12]	8 mA	1.875 V	AVDD	O	
86	T18	T18		CS2*	Chip Select	O	26 MHz		4 mA			GPO[2]; GPO[12]	8 mA	1.875 V	AVDD	O	
73	V17	V17		CS1*	Chip Select	O	26 MHz		4 mA			GPO[2]; GPO[12]	8 mA	1.875 V	AVDD	O	
74	W18	W18		CS0*	Chip Select	O	26 MHz		4 mA			GPO[2]; GPO[12]	8 mA	1.875 V	AVDD	O	
84	T19	T19		BURSTCLK	Synchronous Burst Clock	I/O	26 MHz		4 mA			GPO[6]; GPO[14]	8 mA	1.875 V	AVDD	O	Burst clock is user output; but pad I/O must be configured to capture output at the pad and route back into AEIM as signal input.
113	L16	L16		LBA*	Load Burst Address	O	26 MHz		4 mA			GPO[6]; GPO[10]	8 mA	1.875 V	AVDD	O	
112	N18	N18		ECB*	End Current Burst	I	26 MHz	69k PU				Vss; Vss		1.875 V	AVDD	I	
120	K16	K16		EB0*	Enable Byte 0	O	26 MHz		4 mA			GPO[3]; GPO[13]	8 mA	1.875 V	AVDD	O	
134	G17	G17		EB1*	Enable Byte 1	O	26 MHz		4 mA			GPO[3]; GPO[13]	8 mA	1.875 V	AVDD	O	
77		V19		EB2*	Enable Byte 2	O	26 MHz		4 mA			GPO[3]; GPO[13]	8 mA	1.875 V	AVDD	O	
111		J12		EB3*	Enable Byte 3	O	26 MHz		4 mA			GPO[3]; GPO[13]	8 mA	1.875 V	AVDD	O	
72	T16	T16		OE*	Output Enable	O	26 MHz		4 mA			GPO[3]; GPO[13]	8 mA	1.875 V	AVDD	O	
55	V14	V14	PE3	EBW*	AEIM Data Bus CS0 Default Bandwidth Select.	I/O		22k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	I	NOTE: For LTS/LTE: EBW* functionality is not supported. AEIM CS0* is configured directly to 16-bit bus width at reset. PE3 still controls the I/O configuration of the pin which is an input at reset.; See Table 2-38, “Port E Mux Assignments,” on page 64

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
MCU Trace and Debug																	
70	P16	P16	PA0	CKO	Reset configurable to internal clock sources	I/O	26 MHz		4 mA			GPO[5]; Boost[1]	8 mA	1.875 V	AVDD	O	Assert pad high drive strength to support 50 pF load or 600-Ohm inpedence.; See Table 2-34, “Port A Mux Assignments,” on page 60
109		K12		DB_STRB	Strobe once per memory access.	O	26 MHz		4 mA			GPO[15]; Boost[1]	8 mA	1.875 V	AVDD	O	
103		L13		DB_SZ1	Encoded Data Size for current MCU access	O	26 MHz		4 mA			GPO[15]; Boost[1]	8 mA	1.875 V	AVDD	O	
102		H18		DB_SZ0	Encoded Data Size for current MCU access	O	26 MHz		4 mA			GPO[15]; Boost[1]	8 mA	1.875 V	AVDD	O	
110		J13		DB_SHOW_RW	Encoded access destinations	O	26 MHz		4 mA			GPO[15]; Boost[1]	8 mA	1.875 V	AVDD	O	
108		K13		QSTAT[4]	Encoded processor state	O	26 MHz		4 mA			GPO[15]; Boost[1]	8 mA	1.875 V	AVDD	O	
107		L12		QSTAT[3]	Encoded processor state	O	26 MHz		4 mA			GPO[15]; Boost[1]	8 mA	1.875 V	AVDD	O	
106		M12		QSTAT[2]	Encoded processor state	O	26 MHz		4 mA			GPO[15]; Boost[1]	8 mA	1.875 V	AVDD	O	
105		M11		QSTAT[1]	Encoded processor state	O	26 MHz		4 mA			GPO[15]; Boost[1]	8 mA	1.875 V	AVDD	O	
104		M10		QSTAT[0]	Encoded processor state	O	26 MHz		4 mA			GPO[15]; Boost[1]	8 mA	1.875 V	AVDD	O	
Layer 1 Timer																	
42	W11	W11	PA2	TOUT5	L1T Output for (spare); Alternate use as OWB I/O	I/O		69k PU	+/-16 mA SH		OD	GPO[7]; GPO[16]	8 mA	2.775 V	DVDD	O	L1T Output for (spare); Reserve alternate use as OWB I/O; Reserve alternate GP-Output use as Thermistor Bias; See Table 2-34, “Port A Mux Assignments,” on page 60

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
19	U6	U6	PA4	TOUT7	L1T Output for TX_EN	I/O		69k PD	+/-16 mA SH			GPO[7]; GPO[16]	8 mA	2.775 V	EVDD	O	TX_EN (PA timing signal); See Table 2-34, “Port A Mux Assignments,” on page 60
20	W7	W7	PA5	TOUT8	L1T Output for (spare)	I/O		69k PD	+/-16 mA SH			GPO[7]; GPO[16]	8 mA	2.775 V	EVDD	O	Spare; Reserve alternate use of GP-Output for RF_Quad_Control; Alternate use for Rx_Acq; See Table 2-34, “Port A Mux Assignments,” on page 60
21	T7	T7	PA6	TOUT9	L1T Output for (spare)	I/O		69k PD	+/-16 mA SH			GPO[7]; GPO[16]	8 mA	2.775 V	EVDD	O	Spare; Reserve alternate use of GP-output for SWB+_EN for CE Bus; Alternate use for Debug/DM_CS.; See Table 2-34, “Port A Mux Assignments,” on page 60
22	N9	N9	PA7	TOUT10	L1T Output for EXC_EN	I/O		69k PD	+/-16 mA SH			GPO[7]; GPO[16]	8 mA	2.775 V	EVDD	O	EXC_EN; See Table 2-34, “Port A Mux Assignments,” on page 60
23	V8	V8	PA8	TOUT11	L1T Output for RX_ANT_EN	I/O		69k PD	+/-16 mA SH			GPO[7]; GPO[16]	8 mA	2.775 V	EVDD	O	See Table 2-34, “Port A Mux Assignments,” on page 60
34	U10	U10	PE1	TOUT12 (Alt)	L1T Output	I/O		69k PD	+/-16 mA SH			GPO[7]; GPO[16]	8 mA	2.775 V	EVDD	O	See Table 2-38, “Port E Mux Assignments,” on page 64
A2DIGL Interface																	
18	V6	V6	PA1	RX_EN_OUT	Receive enable output signal to Algae	I/O		69k PD	+/-16 mA SH			GPO[7]; GPO[16]	8 mA	2.775 V	EVDD	O	Receive enable signal to Algae; See Table 2-34, “Port A Mux Assignments,” on page 60
17	T6	T6	PA3	GSM*_DCS	GSM/DCS band select signal to Athena	I/O			+/-16 mA SH			GPO[7]; GPO[16]	8 mA	2.775 V	EVDD	O	GSM*_DCS; See Table 2-34, “Port A Mux Assignments,” on page 60
External Interrupt Control																	
169	C14	C14	PA9	INT0	External Interrupt	I/O		22k PU	4 mA	Hyst		Boost[0]; Boost[1]	4 mA	2.775 V	BVDD	I	Spare. Note: Alt Out 2 assigned to permit RXSDG detect flag visibility as output and return input to DSP. Add alternate output CTS1 for CE Bus. Reserved alternate use of INT0 for HKSW and GP-output for FM_DATA.; See Table 2-34, “Port A Mux Assignments,” on page 60
170	B14	B14	PA10	INT1	External Interrupt	I/O			4 mA	Hyst		Boost[0]; Boost[1]	4 mA	2.775 V	BVDD	I	Seaweed IC Interrupt (IRQ); See Table 2-34, “Port A Mux Assignments,” on page 60

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
24	W8	W8	PA12	INT3	External Interrupt	I/O		22k PU	4 mA	Hyst		Boost[0]; Boost[1]	4 mA	2.775 V	EVDD	I	INT3 reserved for Algae Interrupt (RFDET_FLAG). INT5 reserved for OPTION2 for CE Bus.; See Table 2-34, “Port A Mux Assignments,” on page 60
171	C13	C13	PA13	INT4	External Interrupt	I/O		69k PD	4 mA	Hyst		Boost[0]; Boost[1]	4 mA	2.775 V	BVDD	I	Reserved use of INT4 for OPTION1 for CE Bus. Alternate I/O function defined for output strobe for logic analyzer capture of DSP Program Address bus.; See Table 2-34, “Port A Mux Assignments,” on page 60
146	D19	D19		INT5	External Interrupt	I		22k PU		Hyst		Vss; Vss		2.775 V	CVDD	I	Reserved use of INT5 for OPTION2 for CE Bus.
36	T10	T10	PC9	INT6	External Interrupt	I/O		22k PU	4 mA	Hyst		Boost[0]; Boost[1]	4 mA	2.775 V	EVDD	I	Spare Interrupt; See Table 2-36, “Port C Mux Assignments,” on page 62
SAP																	
172	D13	D13	PB1	SCKA	Serial Clockñclock input or output used by both the transmitter and receiver in asynchronous modes.	I/O		69k PD	4 mA	Hyst		GPO[9]; Boost[1]	8 mA	2.775 V	BVDD	I	Offboard for DAI testing; See Table 2-35, “Port B Mux Assignments,” on page 61
176	B13	B13	PB2	STDA	Serial Transmit Data-transmits data from the serial transmit shiftregister.I/O	I/O		69k PU	4 mA			GPO[9]; Boost[1]	8 mA	2.775 V	BVDD	I	Offboard for DAI testing; See Table 2-35, “Port B Mux Assignments,” on page 61
175	B12	B12	PB4	SRDA	Serial Receive Data	I/O		69k PD	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	BVDD	I	Offboard for DAI testing; See Table 2-35, “Port B Mux Assignments,” on page 61
177	A12	A12	PB3	SC2A	Serial Control Pin 2-frame sync for both the transmitter and receiver only in asynchronous mode.	I/O		69k PD	4 mA			GPO[9]; Boost[1]	8 mA	2.775 V	BVDD	I	Offboard for DAI testing; See Table 2-35, “Port B Mux Assignments,” on page 61
173	A13	A13	PB0	SC1A	Serial Control	I/O		69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	BVDD	I	Offboard for DAI testing. Reserved use of GP-output for MUTE*. Alternate mux I/O SC2A and output for UART1 DCD1 for CE Bus.; See Table 2-35, “Port B Mux Assignments,” on page 61

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
174	G12	G12	PA15	SC0A	Serial Control	I/O		69k PD	4 mA	Hyst		Boost[0]; Boost[1]	4 mA	2.775 V	BVDD	I	Offboard for DAI testing. Reserved use of GP-Input for AUDIO_IN_STATE/BLUE_WAKE* for CE Bus.; See Table 2-34, “Port A Mux Assignments,” on page 60
General Purpose I/O																	
163	D14	D14	PC12	PC12	General Purpose I/O. Reserved alternate use of GP-Output for CHEMISTRY or MIDRATE1.	I/O		69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	I	Memory test TEST/MA17. Reserve alternate GP-Output for MIDRATE1 for CE Bus.; See Table 2-36, “Port C Mux Assignments,” on page 62
165	B15	B15	PC14	PC14	General Purpose I/O. Reserved alternate use of GP-Output for USB_SOFTCONN or MIDRATE2.	I/O		69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	I	USB Soft Connect (expected GPIO output function using MCU output selection). Memory test TEST/MA19. Reserve GP-Output for MIDRATE2 for CE Bus.; See Table 2-36, “Port C Mux Assignments,” on page 62
SIM																	
253	J4	J4	PB7	SIM_RST	SIM Reset	I/O		69k PD	200 uA req’d; 4 mA actual			Boost[0]; Boost[1]	4 mA	2.850 V	FVDD	O	SIM pins connect directly to SIM card and requires separate 2.875 V; See Table 2-35, “Port B Mux Assignments,” on page 61
269	R1	R1	PB8	SIM_PD	SIM Position Detect	I/O		69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	I	See Table 2-35, “Port B Mux Assignments,” on page 61
254	L1	L1	PB9	SIM_CLK	SIM Clock	I/O	4 MHz	69k PD	200 uA req’d; 4 mA actual			Boost[0]; Boost[1]	4 mA	2.850 V	FVDD	O	SIM pins connect directly to SIM card and requires separate 2.875 V; See Table 2-35, “Port B Mux Assignments,” on page 61
255	K3	K3	PB10	SIM_DIO	SIM Data In/Out	I/O		22k PU	+/-1 mA’/ 30 pF/ 1 us req’d; 4 mA actual		OD	Boost[0]; Boost[1]	4 mA	2.850 V	FVDD	O	SIM pins connect directly to SIM card and requires separate 2.875 V. SIM_DIO configured as continuous output when SIM enabled.; See Table 2-35, “Port B Mux Assignments,” on page 61
258	M1	M1	PB11	SIM_VCCEN	SIM VCCEN controls SIM VCC Regulator in Seaweed	I/O		69k PD	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	O	Used for CE/SPS Boot mode select at Reset.; See Table 2-35, “Port B Mux Assignments,” on page 61

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
Display Memory Control																	
266	P1	P1	PB12	SDATA	Serial/Parallel Data Output to LCD Driver and LCD_DATA[7] for parallel data output.	I/O			4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	O	See Table 2-35, “Port B Mux Assignments,” on page 61
265	M4	M4	PB13	LCD_CLK	Serial Clock/Parallel Data Output to LCD Driver and LCD_DATA[6] for parallel output.	I/O			4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	O	See Table 2-35, “Port B Mux Assignments,” on page 61
264	N2	N2	PE9	LCD_DATA[5]	Parallel Data Output to LCD Driver	I/O			4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	O	See Table 2-38, “Port E Mux Assignments,” on page 64
263	L4	L4	PE8	LCD_DATA[4]	Parallel Data Output to LCD Driver	I/O			4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	O	See Table 2-38, “Port E Mux Assignments,” on page 64
262	N1	N1	PE7	LCD_DATA[3]	Parallel Data Output to LCD Driver	I/O			4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	O	See Table 2-38, “Port E Mux Assignments,” on page 64
261	M2	M2	PE6	LCD_DATA[2]	Parallel Data Output to LCD Driver	I/O			4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	O	See Table 2-38, “Port E Mux Assignments,” on page 64
260	L2	L2	PE5	LCD_DATA[1]	Parallel Data Output to LCD Driver	I/O			4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	O	See Table 2-38, “Port E Mux Assignments,” on page 64
259	L3	L3	PE4	LCD_DATA[0]	Parallel Data Output to LCD Driver	I/O			4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	O	See Table 2-38, “Port E Mux Assignments,” on page 64
267	N3	N3	PB14	LCD_CS	Chip Select for LCD Driver	I/O			4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	O	See Table 2-35, “Port B Mux Assignments,” on page 61
268	P2	P2	PB15	LCD_RS	Register Select for LCD Driver	I/O			4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	O	See Table 2-35, “Port B Mux Assignments,” on page 61
Keypad Port																	
235	G3	G3	PC3	COLUMN0		I/O		22k PU	4 mA	Hyst	OD	Boost[0]; Boost[1]	4 mA	2.775 V	JVDD	I	Tarpon uses Column 0; 1 and Rows 0; 1; 4-7; See Table 2-36, “Port C Mux Assignments,” on page 62
236	F2	F2	PC4	COLUMN1		I/O		22k PU	4 mA	Hyst	OD	Boost[0]; Boost[1]	4 mA	2.775 V	JVDD	I	Tarpon uses Column 0; 1 and Rows 0; 1; 4-7; See Table 2-36, “Port C Mux Assignments,” on page 62
237	F3	F3	PC5	ROW0	Reserve alternate use of GP-Input For ADJ_READY.	I/O		22k PU	4 mA	Hyst		Boost[0]; Boost[1]	4 mA	2.775 V	JVDD	I	Tarpon uses Column 0; 1 and Rows 0; 1; 4-7; See Table 2-36, “Port C Mux Assignments,” on page 62

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
238	F1	F1	PC6	ROW1	Reserve alternate use of GP-Output for ADJ_WAKE.	I/O		22k PU	4 mA	Hyst		Boost[0]; Boost[1]	4 mA	2.775 V	JVDD	i	Tarpon uses Column 0; 1 and Rows 0; 1; 4-7; See Table 2-36, “Port C Mux Assignments,” on page 62
239	G4	G4		ROW2		I		22k PU		Hyst		Vss; Vss		2.775 V	JVDD	I	
240	H7	H7		ROW3		I		22k PU		Hyst		Vss; Vss		2.775 V	JVDD	I	
241	J7	J7	PC7	ROW4		I/O		22k PU	4 mA	Hyst		Boost[0]; Boost[1]	4 mA	2.775 V	JVDD	I	Tarpon uses Column 0; 1 and Rows 0; 1; 4-7; See Table 2-36, “Port C Mux Assignments,” on page 62
242	G1	G1	PC8	ROW5		I/O		22k PU	4 mA	Hyst		Boost[0]; Boost[1]	4 mA	2.775 V	JVDD	I	Tarpon uses Column 0; 1 and Rows 0; 1; 4-7; See Table 2-36, “Port C Mux Assignments,” on page 62
243	H2	H2	PC10	ROW6		I/O		22k PU	4 mA	Hyst		Boost[0]; Boost[1]	4 mA	2.775 V	JVDD	I	Tarpon uses Column 0; 1 and Rows 0; 1; 4-7; See Table 2-36, “Port C Mux Assignments,” on page 62
244	G2	G2	PC11	ROW7		I/O		22k PU	4 mA	Hyst		Boost[0]; Boost[1]	4 mA	2.775 V	JVDD	I	Tarpon uses Column 0; 1 and Rows 0; 1; 4-7; See Table 2-36, “Port C Mux Assignments,” on page 62
Real Time Clock																	
231	E3	E3		CKIL	(RTC_CLK1 - 32.768 kHz)	I	32 kHz					Vss; Vss		1.575 V	LVDD	I	From Seaweed
230	G8	G8		STANDBY	Deep sleep output triggered off Layer 1 Timer and RTC TODA_INT output.	O			4 mA			Vss; Vss	4 mA	1.575 V	LVDD	O	To Seaweed. Must support rise/fall time < 1us.
Clock Control																	
208	A4	A4		XTAL	REF_OSC_CLK - 26 MHz oscillator differential.	A	26 MHz Sine Wave							2.775 V	VDDA		
209	B4	B4		EXTAL	REF_OSC_CLK - 26 MHz oscillator differential.	A	26 MHz Sine Wave							2.775 V	VDDA		
210	B5	B5		TRKG_OSC_ OUT	Tracking oscillator output.	O	26 MHz	69k PD	4 mA			Vss; Boost[1]	4 mA	2.775 V	VDDA	O	26 MHz

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
RF Interface																	
25	V7	V7	PD0	RF_CLK	Pass-through SPI clock	I/O	13 MHz		4 mA			GPO[8]; Boost[1]	8 mA	2.775 V	EVDD	O	Algae SPI; See Table 2-37, “Port D Mux Assignments,” on page 63
27	U8	U8	PD1	RF_DATA	Pass-through SPI data	I/O	13 MHz		4 mA			GPO[8]; Boost[1]	8 mA	2.775 V	EVDD	O	Algae SPI; See Table 2-37, “Port D Mux Assignments,” on page 63
28	W9	W9	PD2	RF_CS	Pass-through CS	I/O	13 MHz	69k PD	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	EVDD	O	Algae Chip Select; sourced by MQSPI/CS5 through A2DIGL; See Table 2-37, “Port D Mux Assignments,” on page 63
SPI																	
43	T11	T11	PD3	QSCKA	(SPI_CLK)	I/O	13 MHz		4 mA			GPO[4]; Boost[1]	8 mA	2.775 V	DVDD	O	Seaweed SPI; See Table 2-37, “Port D Mux Assignments,” on page 63
44	V11	V11	PD4	MISOA	MQSPI Master data In A	I/O	13 MHz	69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	DVDD	I	Seaweed SPI; See Table 2-37, “Port D Mux Assignments,” on page 63
45	V12	V12	PD5	MOSIA	MQSPI Master data out A	I/O	13 MHz		4 mA			GPO[4]; Boost[1]	8 mA	2.775 V	DVDD	O	Seaweed SPI; See Table 2-37, “Port D Mux Assignments,” on page 63
26	T8	T8	PE14	MISOB (Alt)	MQSPI Master data In B	I/O	13 MHz	69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	EVDD	I	Reserve alternate use of GP-Output for GPS_WAKE.; See Table 2-38, “Port E Mux Assignments,” on page 64
46	T12	T12	PD6	SPI_CS0	MQSPI Chip Select.	I/O	13 MHz	69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	DVDD	O	Reserve use of CS0 for CLI display. Reserve alternate use of GP-Output for BLUE_GPS_SELECT.; See Table 2-37, “Port D Mux Assignments,” on page 63
47	U12	U12	PD7	SPI_CS1	MQSPI Chip Select.	I/O	13 MHz	69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	DVDD	O	Reserve use of CS1 for CLI display. Reserve alternate use of GP-Output for BLUE_WAKE*.; See Table 2-37, “Port D Mux Assignments,” on page 63
48	T13	T13	PD8	SPI_CS2	MQSPI Chip Select.	I/O	13 MHz		4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	DVDD	O	Reserve use for camera select.; See Table 2-37, “Port D Mux Assignments,” on page 63
49	W12	W12	PD9	SPI_CS3	MQSPI chip select for Seaweed.	I/O	13 MHz	69k PD	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	DVDD	O	Seaweed Chip Select; See Table 2-37, “Port D Mux Assignments,” on page 63
147	D18	D18	PC15	SPI_CS6	MQSPI Chip Select.	I/O	13 MHz	69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	O	Reserve use for NAV Key SPI CS.; See Table 2-36, “Port C Mux Assignments,” on page 62

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
35	N10	N10	PE2	SPI_CS8 (Alt)	MQSPI Chip Select. Reserved for RF_CS2	I/O	13 MHz	69k PD	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	EVDD	O	See Table 2-38, “Port E Mux Assignments,” on page 64
JTAG																	
56	W14	W14		TDI	Test Data Input	I		47k PU				Vss; Vss		2.775 V	CVDD		
57	U14	U14		TDO	Test Data Output	O		47k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD		
58	N12	N12		TRST	Test Reset	I		47k PU		Hyst		Vss; Vss		2.775 V	CVDD		
60	V15	V15		TMS	Test Mode Select	I		47k PU				Vss; Vss		2.775 V	CVDD		
66	W16	W16		TCK	Test Clock	I		47k PU				Vss; Vss		2.775 V	CVDD		
64	T15	T15	PC1	RTCK	ARM debugger ref clock	I/O			4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	O	See Table 2-36, “Port C Mux Assignments,” on page 62
61	U15	U15		MCU_DE*	MCU Debug Event	I/O		47k PU	4 mA		OD	Boost[0]; Boost[1]	4 mA	2.775 V	CVDD		
62	W15	W15		DSP_DE*	DSP Debug Event	I/O		47k PU	4 mA		OD	Boost[0]; Boost[1]	4 mA	2.775 V	CVDD		
164	A15	A15	PC13	SJC_MOD	Mode select for System JTAG	I/O		69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	I	See Table 2-36, “Port C Mux Assignments,” on page 62
Miscellaneous Control																	
50	V13	V13		RESET_IN*	System External Reset Input	I				Hyst		Vss; Vss		2.775 V	DVDD	I	
11	W5	W5		RESET_OUT*	Internal Reset Output	O			4 mA			Boost[0]; Boost[1]	4 mA	1.875 V	AVDD	O	
51	U13	U13	PA14	WDOG*	(WDO) Watch Dog Reset Output	I/O			4 mA			GPO[5]; Boost[1]	8 mA	2.775 V	DVDD	O	NOTE: GPIO configures as output; but WDOG module disables output driver at RESET* (High Z).; See Table 2-34, “Port A Mux Assignments,” on page 60
245	K7	K7		MOD	Boot Mode Select ñ low=external flash execution; high=internal ROM execution	I		69k PU				Vss; Vss		2.775 V	JVDD	I	

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
250	J3	J3		TEST0	Enables test modes	I		69k PD				Vss; Vss		2.850 V	FVDD	I	Active High asserts test modes.
251	H3	H3		TEST1	Enables test modes	I		69k PD				Vss; Vss		2.850 V	FVDD	I	Test mode selection pin.
252	K1	K1		TEST2	Enables test modes	I		69k PD				Vss; Vss		2.850 V	FVDD	I	Test mode selection pin.
USB																	
157	D15	D15	PD10	USB_TXENB	Output Enable. Active LOW. enables the external USB transceiver to transmit data on the bus	I/O		69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	O	See Table 2-37, “Port D Mux Assignments,” on page 63
158	A17	A17	PD11	USB_VPIN	Gated version of D+ from the USB transceiver. Outputs are logic ì0î and logic ì1î. Used to detect single ended zero (SE0) error conditions and interconnect speed	I/O		69k PD	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	I	See Table 2-37, “Port D Mux Assignments,” on page 63
159	C16	C16	PD12	USB_VMIN	Gated version of D- from the USB transceiver. Outputs are logic ì0î and logic ì1î. Used to detect single ended zero (SE0) error conditions and interconnect speed	I/O		69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	I	See Table 2-37, “Port D Mux Assignments,” on page 63
160	B16	B16	PD13	USB_XRXD	CMOS level receive data input from external USB transceiver	I/O		69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	I	See Table 2-37, “Port D Mux Assignments,” on page 63
161	C15	C15	PD14	USB_VPOUT	Differential transmit data output to external USB transceiver	I/O		69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	O	See Table 2-37, “Port D Mux Assignments,” on page 63
162	A16	A16	PD15	USB_VMOUT	Differential transmit data output to external USB transceiver	I/O		69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	O	See Table 2-37, “Port D Mux Assignments,” on page 63

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
156	B17	B17	PA11	USB_SUSPEND	Assert power control to transceiver	I/O		69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	O	Reserved use with Seaweed.; See Table 2-34, “Port A Mux Assignments,” on page 60
RX and TX Synthesizers																	
216	C4	C4		VDD_CP	Supply for charge pump	P								4.750 V	VDD_CP		
212	B2	B2		VSS_CP	Ground for charge pump	P								4.750 V	VDD_CP		
204	A5	A5		REG_BYP_PRE	Output of prescaler regulator for bypassing. Supply for prescaler circuits. Input of this regulator is VDD_TXRX	P								1.575 V	REG_BYP_PRE		(External Capacitor) 0.01uf ~ 0.1uf
202	C7	C7		VSS_PRE (2)	Ground for prescaler circuit for bypassing. Package name VSS_PRE.	P	DC							1.575 V	REG_BYP_PRE		Double-Bonded
206				VSS_PRE (1)	Double bond (pad only) to complement VCO1.	P	DC							1.575 V	REG_BYP_PRE		Double-Bonded
215	D4	D4		TX_CP	TX Charge Pump	A	<26 MHz		+/-1 mA					4.750 V	VDD_CP		
213	D5	D5		RX_CP	RX Charge Pump	A	<26 MHz		+/-1 mA					4.750 V	VDD_CP		
214	B3	B3		CP_BIAS	(Requires 1% External Resistor)	A								4.750 V	VDD_CP		
203	A6	A6		VCO1_N	Rx/Tx VCO differential for Neptune.	RF	1 GHz		RF					1.575 V	REG_BYP_PRE		Reduced ESD of 500V is required due to high frequency; Voltage=200 mV PP min Freq=sine wave
205	B6	B6		VCO1_P	Rx/Tx VCO differential for Neptune.	RF	1 GHz		RF					1.575 V	REG_BYP_PRE		Reduced ESD of 500V is required due to high frequency; Voltage=200 mV PP min Freq=sine wave
199	D8	D8		TX_MOD	TX Dual Port Modulation Output	A	<10 MHz							2.775 V	VDD_TXRX		

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
UART1																	
178	G11	G11	PE10	PE10	Reset value to GP input. Preferred selection for UART1 TXD1 Transmit Data. Reserved use of GP-Input for DSEL0 for CE Bus.	I/O		69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	BVDD	I	Reserved use of GP-Input for DSEL0 for CE Bus.; See Table 2-38, “Port E Mux Assignments,” on page 64
179	G10	G10	PE11	PE11	Reset value to GP input. Preferred selection for UART1 RXD1 Receive Data	I/O		69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	BVDD	I	Reserved use of GP-Input for DSEL1 for CE Bus.; See Table 2-38, “Port E Mux Assignments,” on page 64
168	A14	A14	PE12	PE12	Reset value to GP input. Preferred selection for UART1 RTS1 Ready to Send.	I/O		69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	BVDD	I	Reserved use of GP-Input for DSEL2 for CE Bus.; See Table 2-38, “Port E Mux Assignments,” on page 64
41	U11	U11	PC0	CTS1 (Alt)	UART1 Clear to Send.	I/O		69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	DVDD	O	Reserve alternate use of GP-Output for BL_HOST_WAKE for CE Bus.; See Table 2-36, “Port C Mux Assignments,” on page 62
59	N13	N13	PB5	TXD2	UART2 Transmit Data	I/O		69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	O	See Table 2-35, “Port B Mux Assignments,” on page 61
63	N17	N17	PB6	RXD2	UART2 Receive Data	I/O		69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	I	See Table 2-35, “Port B Mux Assignments,” on page 61
154	D16	D16	PE13	CTS2	UART2 Clear to Send	I/O		69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	O	See Table 2-38, “Port E Mux Assignments,” on page 64
65	V16	V16	PE15	RTS2	UART2 Ready to Send	I/O		69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	I	See Table 2-38, “Port E Mux Assignments,” on page 64
148	E17	E17		UTXD1	UART1 dedicated Transmit Data	O			4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	O	
149	C19	C19		URXD1	UART1 dedicated Receive Data	I		69k PU				Vss; Boost[1]		2.775 V	CVDD	I	
150	D17	D17		UCTS1	UART1 dedicated Clear to Send	O			4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	O	
151	C18	C18	PE0	URTS1	UART1 dedicated Ready to Send	I/O		69k PU	4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	I	See Table 2-38, “Port E Mux Assignments,” on page 64
BaseBand Port																	

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
33	W10	W10		SC2B	BBP Frame Sync	I/O		69k PD	4 mA			GPO[9]; Boost[1]	8 mA	2.775 V	EVDD	I	
32	T9	T9		SCKB	BBP Serial Bit Clock	I/O		69k PD	4 mA			GPO[9]; Boost[1]	8 mA	2.775 V	EVDD	I	
31	U9	U9		STDB (Alt)	BBP Serial Transmit Data	O		69k PU	4 mA			GPO[9]; Boost[1]	8 mA	2.775 V	EVDD	O	
PA Control																	
183	A10	A10		PA_OUT	Power Amp Control	A	<10 MHz		+10/-5 mA into 0.01 uF					2.775 V	VDD_ TXRX		PA module
185	D12	D12		PA_REF	Power Amp Feedback	A	<10 MHz							2.775 V	VDD_ TXRX		PA module
184	B10	B10		PA_DET	Power Amp Feedback	A	<10 MHz							2.775 V	VDD_ TXRX		PA module
229	F4	F4		LOW_BATT*	Pulls down AOC on instantaneous low battery (current surge)	I	DC	47k PU				Vss; Vss		1.575 V	LVDD	I	From Seaweed
186	C12	C12		ATST_P	Analog Test Positive	A								2.775 V	VDD_ TXRX		For all analog not just CODEC
187	D11	D11		ATST_N	Analog Test Negative	A								2.775 V	VDD_ TXRX		For all analog not just CODEC
RX Data Converters																	
196	D9	D9		VDD_TXRX	Input to TXRX regulator	P								2.775 V	VDD_ TXRX		
188	C10	C10		VSS_TXRX (3)	Ground for analog part of TX and RX circuits. Package name VSS_TXRX.	P								2.775 V	VDD_ TXRX		Double-Bonded
200				VSS_TXRX (2)	Double bond (pad only) to package pin VSS_TXRX (3).	P								2.775 V	VDD_ TXRX		Double-Bonded
201	C8	C8		VSS_TXRX (1)	Additional ground for analog part of TX and RX circuits. Package name VSS_TXRX (3).	P								2.775 V	VDD_ TXRX		Third Bond and second pin.

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
197	B7	B7		REG_BYP_TXRX (_O)	RX Output of TXRX regulator output for bypassing. Supply for analog part of TX and RX circuits. Package name REG_BYP_TXRX.	P								2.775 V	VDD_TXRX		Double-Bonded. Bond pad is REG_BYP_TXRX_O. Bond pads REG_BYP_TXRX_O and REG_BYP_TXRX_I are both connect to REG_BYP_TXRX pin
198				REG_BYP_TXRX (_I)	Double bond (pad only) to package pin REG_BYP_TXRX. Power input for TX/RX analog circuits from TXRX regulator.	P								2.775 V	VDD_TXRX		Double-Bonded. Bond pad is REG_BYP_TXRX_I. Bond pads REG_BYP_TXRX_O and REG_BYP_TXRX_I are both connect to REG_BYP_TXRX pin
189	C9	C9		VAG_RX	Pad for bypass cap of internally generated reference.	P								2.775 V	VDD_TXRX		Internally generated. Pad for bypass cap.
190	A8	A8		RX_I	RX ADC I Channel Input	A	270 kHz							2.775 V	VDD_TXRX		
191	B8	B8		RX_IX	RX ADC I Channel Input	A	270 kHz							2.775 V	VDD_TXRX		
193	A9	A9		RX_Q	RX ADC Q Channel Input	A	270 kHz							2.775 V	VDD_TXRX		
192	B9	B9		RX_QX	RX ADC Q Channel Input	A	270 kHz							2.775 V	VDD_TXRX		
194	A7	A7		REF_HI	ADC Reference	A								2.775 V	VDD_TXRX		
195	D10	D10		REF_LO	ADC reference	A								2.775 V	VDD_TXRX		
Audio Codec																	
221	D7	D7		VDD_CODEC	Input to CODEC regulator	P								2.775 V	VDD_CODEC		from Seaweed
222	C2	C2		VSS_CODEC	Supply for analog part of CODEC circuit	P								2.775 V	VDD_CODEC		
223	H13	H13		REG_BYP_CODEC	Output of CODEC regulator for bypassing. Supply for analog part of CODEC circuit	P								2.775 V	VDD_CODEC		

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
227	D1	D1		VAG_CODEC	Analog Reference for analog part of CODEC circuit	P								2.775 V	VDD_CODEC		from Seaweed
226	C1	C1		VAGOUT_CODEC	Buffered version of VAG _CODEC	P								2.775 V	VDD_CODEC		
217	C6	C6		REF_CODEC_P	Codec reference differential bypass.	A								2.775 V	VDD_CODEC		
218	D6	D6		REF_CODEC_N	Codec reference differential bypass.	A								2.775 V	VDD_CODEC		
224	G9	G9		ADC_N	Codec ADC Differential Input	A	8 kHz							2.775 V	VDD_CODEC		
225	D3	D3		ADC_P	Codec ADC Differential Input	A	8 kHz							2.775 V	VDD_CODEC		
220	C3	C3		DAC_N	Codec DAC Differential Output	A	8 kHz							2.775 V	VDD_CODEC		
219	E4	E4		DAC_P	Codec DAC Differential Output	A	8 kHz							2.775 V	VDD_CODEC		
General Purpose ADC																	
234	E1	E1		ADC_DATA	ADC analog input	A								2.775 V	JVDD		Seaweed analog mux output
52	W13	W13	PC2	ADC_SYNC	ADC converter control to analog mux on Seaweed	I/O			4 mA			GPO[5]; Boost[1]	8 mA	2.775 V	DVDD	O	Seaweed analog mux control. Alternate input for INT7.; See Table 2-36, “Port C Mux Assignments,” on page 62
Power Supplies																	
207	C5	C5		VDDA	Reference Clock Power. Note: Expect this pin to be defined as dedicated power for 26 MHz oscillator.	P								2.775 V	VDDA		

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
211	A3	A3		VSSA	Reference Clock GND. Note: expect this pin to be defined as dedicated ground for 26 MHz oscillator.	P								2.775 V	VDDA		
13	V5	V5		AVDD (1_PAD2)	Package: AVDD(1)	P								1.875 V	AVDD		
82	R16	R16		AVDD (3_PAD2)	Package: AVDD(2)	P								1.875 V	AVDD		
96	P18	P18		AVDD (3_PAD1)	Package: AVDD(3)	P								1.875 V	AVDD		
114	L19	L19		AVDD (2_PAD3)	Package: AVDD(4)	P								1.875 V	AVDD		
130	H17	H17		AVDD (2_PAD2)	Package: AVDD(5)	P								1.875 V	AVDD		
144	E18	E18		AVDD (2_PAD1)	Package: AVDD(6)	P								1.875 V	AVDD		
284	V2	V2		AVDD (1_PAD1)	Package: AVDD(7)	P								1.875 V	AVDD		
1	R4	R4		AVSS (1_PAD2)	Package: AVSS(1)	P								1.875 V	AVDD		
68	W17	W17		AVSS (3_PAD2)	Package: AVSS(2)	P								1.875 V	AVDD		
83	N16	N16		AVSS (3_PAD1)	Package: AVSS(3)	P								1.875 V	AVDD		
101	M19	M19		AVSS (2_PAD3)	Package: AVSS(4)	P								1.875 V	AVDD		
117	K19	K19		AVSS (2_PAD2)	Package: AVSS(5)	P								1.875 V	AVDD		
131	H16	H16		AVSS (2_PAD1)	Package: AVSS(6)	P								1.875 V	AVDD		
271	R2	R2		AVSS (1_PAD1)	Package: AVSS(7)	P								1.875 V	AVDD		
180	A11	A11		BVDD		P								2.775 V	BVDD		
167	E16	E16		BVSS		P								2.775 V	BVDD		

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
67	U16	U16		CVDD (3_PAD)	Package: CVDD(1)	P								2.775 V	CVDD		
152				CVDD (2_PAD2)	Double bond only to package pin CVDD(2).	P								2.775 V	CVDD		
166	C17	C17		CVDD (2_PAD1)	Package: CVDD(2)	P								2.775 V	CVDD		
270	P3	P3		CVDD (1_PAD)	Package: CVDD(3)	P								2.775 V	CVDD		
54	T14	T14		CVSS (3_PAD)	Package: CVSS(1)	P								2.775 V	CVDD		
145	A18	A18		CVSS (2_PAD2)	Package: CVSS(2)	P								2.775 V	CVDD		
153				CVSS (2_PAD1)	Double bond only for 264- and 280-pin packages.	P								2.775 V	CVDD		
257	K4	K4		CVSS (1_PAD)	Package: CVSS(3)	P								2.775 V	CVDD		
53	N11	N11		DVDD		P								2.775 V	DVDD		
40	V10	V10		DVSS		P								2.775 V	DVDD		
29	V9	V9		EVDD (_PAD2)	Package: EVDD	P								2.775 V	EVDD		
30				EVDD (_PAD1)	Double bond only to package pin EVDD.	P								2.775 V	EVDD		
16	U7	U7		EVSS (_PAD2)	Package: EVSS	P								2.775 V	EVDD		
39				EVSS (_PAD1)	Double bond only to package pin EVSS.	P								2.775 V	EVDD		
256	K2	K2		FVDD	Power supply for level shift.	P								2.850 V	FVDD		
249	J1	J1		FVSS		P								2.850 V	FVDD		
246	H1	H1		JVDD		P								2.775 V	JVDD		
233	G7	G7		JVSS		P								2.775 V	JVDD		
232	E2	E2		LVDD	Real Time Clock Power	P								1.575 V	LVDD		Supply RTC logic and IO Recommended Voltage Range: 1.575 V typ; 1.5 Vmin; 1.65 Vmax

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
228	D2	D2		LVSS	Real Time Clock GND	P								1.575 V	LVDD		Supply RTC logic and IO
99				QVDD (_PAD1)	Double bond (pad only) to package pin QVDD.	P								1.875 V	QVDD		Double Bonded.
98	M17	M17		QVDD (_PAD2)	Input of core regulator. Package name QVDD.	P								1.875 V	QVDD		Double Bonded.
97	N19	N19		QVSS	Input of core regulator and ground for bulk bypass capacitor.	P								1.875 V	QVDD		
100	M18	M18		REG_BYP_CORE	Output of core regulator for bypassing. Bulk capacitor.	P								1.875 V	QVDD		
15	W6	W6		VDD (1_PAD)	Output of core regulator. For high frequency bypassing. Package: VDD(1). RTL: VDD4_PAD.	P								2.775 V	EVDD		NOTE: The Supply Group listed is only for ESD protection structure connection. It does not represent VDD level for this pad.
116	L17	L17		VDD (2_PAD)	Output of core regulator. For high frequency bypassing. Package: VDD(2). RTL: VDD3_PAD.	P								1.875 V	AVDD		NOTE: The Supply Group listed is only for ESD protection structure connection. It does not represent VDD level for this pad.
182	C11	C11		VDD (3_PAD)	Output of core regulator. For high frequency bypassing. Package: VDD(3). RTL: VDD2_PAD.	P								2.775 V	VDD_TXRX		NOTE: The Supply Group listed is only for ESD protection structure connection. It does not represent VDD level for this pad.
248	J2	J2		VDD (4_PAD)	Output of core regulator. For high frequency bypassing. Package: VDD(4). RTL: VDD1_PAD.	P								2.775 V	JVDD		NOTE: The Supply Group listed is only for ESD protection structure connection. It does not represent VDD level for this pad.
14	U5	U5		VSS (1_PAD)	Output of core regulator. For high frequency bypassing. Package: VSS(1). RTL: VSS4_PAD.	P								2.775 V	EVDD		NOTE: The Supply Group listed is only for ESD protection structure connection. It does not represent VDD associated with this pad.

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
115	L18	L18		VSS (2_PAD)	Output of core regulator. For high frequency bypassing. Package: VSS(2). VSS3_PAD.	P								1.875 V	AVDD		NOTE: The Supply Group listed is only for ESD protection structure connection. It does not represent VDD associated with this pad.
181	B11	B11		VSS (3_PAD)	Output of core regulator. For high frequency bypassing. Package: VSS(3). VSS2_PAD.	P								2.775 V	BVDD		NOTE: The Supply Group listed is only for ESD protection structure connection. It does not represent VDD associated with this pad.
247	H4	H4		VSS (4_PAD)	Output of core regulator. For high frequency bypassing. Package: VSS(4). VSS1_PAD.	P								2.775 V	JVDD		NOTE: The Supply Group listed is only for ESD protection structure connection. It does not represent VDD associated with this pad.
Test specific and Extra Grounds																	
155				ECC_BITFIXED	Probe only test pad	O	52 MHz		4 mA			Boost[0]; Boost[1]	4 mA	2.775 V	CVDD	T	Probe only test pad
12				ECC_NOERR	Probe only test pad	O	52 MHz		4 mA			Boost[0]; Boost[1]	4 mA	1.875 V	AVDD	T	Probe only test pad
37				FUSE_RSTB (LTE)	(LTE) Probe only test pad	I		22K PU				vss; vss		2.775 V	EVDD	I	Probe only test pad
38				FUSE_MODE (LTE)	(LTE) Probe only test pad	I		22K PU				vss; vss		2.775 V	EVDD	I	Probe only test pad
	V1	V1		AVDD (1_PAD1 Dup1)	Duplicate pin for corner redundancy. Package: AVDD(7)												This package pin is a duplicate copy.
	W1	W1		AVDD (1_PAD1 Dup2)	Duplicate pin for corner redundancy. Package: AVDD(7)												This package pin is a duplicate copy.
	W2	W2		AVDD (1_PAD1 Dup3)	Duplicate pin for corner redundancy. Package: AVDD(7)												This package pin is a duplicate copy.
	W19	W19		CS0* (Dup1)	Duplicate pin for corner redundancy												This package pin is a duplicate copy.
	V18			CS0* (Dup2)	Duplicate pin for corner redundancy												This package pin is a duplicate copy.
	V19			CS0* (Dup3)	Duplicate pin for corner redundancy												This package pin is a duplicate copy.

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
	A19	A19		CVSS (2_PAD2 Dup1)	Duplicate pin for corner redundancy. Package: CVSS(2)												This package pin is a duplicate copy.
	B18			CVSS (2_PAD2 Dup2)	Duplicate pin for corner redundancy. Package: CVSS(2)												This package pin is a duplicate copy.
	B19			CVSS (2_PAD2 Dup3)	Duplicate pin for corner redundancy. Package: CVSS(2)												This package pin is a duplicate copy.
	A1	A1		VSS_CP (Dup1)	Duplicate pin for corner redundancy												This package pin is a duplicate copy.
	A2	A2		VSS_CP (Dup2)	Duplicate pin for corner redundancy												This package pin is a duplicate copy.
	B1	B1		VSS_CP (Dup3)	Duplicate pin for corner redundancy												This package pin is a duplicate copy.
	H8			No Connect	This pin is a no connect for this package. This pin is part of no connect group NC_H8.												This package pin is a no connect.
	J8			No Connect	This pin is a no connect for this package. This pin is part of no connect group NC_H8.												This package pin is a no connect.
	K8			No Connect	This pin is a no connect for this package. This pin is part of no connect group NC_H8.												This package pin is a no connect.
	L8			No Connect	This pin is a no connect for this package. This pin is part of no connect group NC_H8.												This package pin is a no connect.
	M8			No Connect	This pin is a no connect for this package. This pin is part of no connect group NC_H8.												This package pin is a no connect.

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
	P17			No Connect	This pin is a no connect for this package. This pin is NC_P17.												This package pin is a no connect.
	M9			No Connect	This pin is a no connect for this package. This pin is part of no connect group NC_M9.												This package pin is a no connect.
	G16			No Connect	This pin is a no connect for this package. This pin is part of no connect group NC_J12.												This package pin is a no connect.
	H12			No Connect	This pin is a no connect for this package. This pin is part of no connect group NC_H9.												This package pin is a no connect.
	G13			No Connect	This pin is a no connect for this package. This pin is part of no connect group NC_H9.												This package pin is a no connect.
	H11			No Connect	This pin is a no connect for this package. This pin is part of no connect group NC_H9.												This package pin is a no connect.
	H10			No Connect	This pin is a no connect for this package. This pin is part of no connect group NC_H9.												This package pin is a no connect.
	H9			No Connect	This pin is a no connect for this package. This pin is part of no connect group NC_H9.												This package pin is a no connect.

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
	J12			No Connect	This pin is a no connect for this package. This pin is part of no connect group NC_J12.												This package pin is a no connect.
	K12			No Connect	This pin is a no connect for this package. This pin is part of no connect group NC_J12.												This package pin is a no connect.
	L13			No Connect	This pin is a no connect for this package. This pin is part of no connect group NC_J12.												This package pin is a no connect.
	H18			No Connect	This pin is a no connect for this package. This pin is NC_H18.												This package pin is a no connect.
	J13			No Connect	This pin is a no connect for this package. This pin is part of no connect group NC_J12.												This package pin is a no connect.
	K13			No Connect	This pin is a no connect for this package. This pin is part of no connect group NC_J12.												This package pin is a no connect.
	L12			No Connect	This pin is a no connect for this package. This pin is part of no connect group NC_J12.												This package pin is a no connect.
	M12			No Connect	This pin is a no connect for this package. This pin is part of no connect group NC_M9.												This package pin is a no connect.

Table 2-1. Neptune LTE Pin Assignments

Pad #	Prod Pin	Dev Pins	GPIO	Pin Name	Description	Pin Type	Max Freq	Pull U/D	Std Drive (RST)	Hyst -eris is	Open Drain	Hi; Max Drive Control	Pad Default	Typical Voltage	Pwr Grp	RST State	Comments; Alternate Functions
	M11			No Connect	This pin is a no connect for this package. This pin is part of no connect group NC_M9.												This package pin is a no connect.
	M10			No Connect	This pin is a no connect for this package. This pin is part of no connect group NC_M9.												This package pin is a no connect.

2.2 Detail Pin Description

In the signal description tables, “Reset Default” means that this signal is selected at reset. “Mux Alternate” means that the signal is available through selection of one of the GPIO alternate inputs or outputs. In the signal description tables below, signals names suffixed with “*” are low active signals. The “*” is the same as applying an overbar to the signal name.

2.2.1 AEIM Signals

NOTE:

D16-D31 are available in the development package for external visibility to the internal data bus and for external ROM emulation for software development.

NOTE:

(Neptune LTE does not support $\overline{\text{EBW}}$ functionality. AEIM $\overline{\text{CS0}}$ defaults to select 16-bit bus width at reset. Port E3 still supports general purpose I/O functionality on the $\overline{\text{EBW}}$ pin.)

Table 2-2. AEIM Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
D16 -D31			(Data Bus)	Active-high, bi-directional input/outputs. D16 is the least significant bit and D31 is the most significant. These pins provide the bi-directional data bus for external memory accesses. D31ñD16 are held in the previous logic state when there is no external bus activity. This is done with weak ìkeepersî inside the I/O buffers. They are also kept in their previous state during hardware reset
D0-D15			(Data Bus)	Active-high, bi-directional input/outputs. D0 is the least significant bit and D15 is the most significant. These pins provide the bi-directional data bus for external memory accesses. D0ñD15 are held in the previous logic state when there is no external bus activity. This is done with weak ìkeepersî inside the I/O buffers. They are also kept in their previous state during hardware reset.
A0-A24			(Address Bus)	Active-high outputs, specifies the address for external memory accesses. ADDR0 is the least significant bit and ADDR23 is the most significant. To minimize power dissipation, ADDR0ñADDR23 do not change state when external memory is not being accessed.
R/W*			(Read/Write)	The output signal R/W indicates the bus access type. When accessing memories it can be used as WE. In the case of accessing a peripheral chip, this pin acts as a read/write. R/W is set to high (logic one) during hardware reset.

Table 2-2. AEIM Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
CS5			(Chip Select 5)	This active-high output signal is asserted based on the decode of the internal address bus bits A31-A24 of the access address.
CS4*			(Chip Select 4)	This active-low output signal is asserted based on the decode of the internal address bus bits A31-A24 of the access address.
CS3*			(Chip Select 3)	This active-low output signal is asserted based on the decode of the internal address bus bits A31-A24 of the access address.
CS2*			(Chip Select 2)	This active-low output signal is asserted based on the decode of the internal address bus bits A31-A24 of the access address.
CS1*			(Chip Select 1)	This active-low output signal is asserted based on the decode of the internal address bus bits A31-A24 of the access address.
CS0*			(Chip Select 0)	This active-low output signal is asserted based on the decode of the internal address bus bits A31-A24 of the access address. It is used as the external Flash memory chip select. This pin defaults to a chip select with 62 wait states after reset.
BURSTCLK			(Burst Clock)	This output signal is used to clock external burst capable devices to synchronize the loading and incrementing of addresses, and delivery of burst read data to the EIM.
LBA*			(Load Burst Address)	This active-low output signal is asserted during burst mode accesses to cause the external burst capable device to load a new starting burst address. Assertion of LBA indicates that a valid address is present on the address bus.
ECB*			(End Current Burst)	This active-low input signal is asserted by external burst capable devices to indicate the end of the current (continuous) burst sequence. Following assertion, the EIM will terminate the current burst sequence and initiate a new one.
EB0*			(Enable Byte 0)	This active-low output signal indicates access to the data byte 0 DATA[7:0] during read or write cycle. This pin may be used as Write Byte Enable if programmed so.The EB0 output is used when accessing 16 bits wide SRAM with byte select capability.
EB1*			(Enable Byte 1)	This active-low output signal indicates access to the data byte 1 DATA[15:8] during read or write cycle. This pin may be used as Write Byte Enable if programmed so. The EB1 output is used when accessing 16 bits wide SRAM with byte select capability.

Table 2-2. AEIM Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
EB2*			(Enable Byte 2)	This active-low output signal indicates access to the data byte 2 DATA[23:16] during read or write cycle. This pin may be used as Write Byte Enable if programmed so.
EB3*			(Enable Byte 3)	This active-low output signal indicates access to the data byte 3 DATA[31:24] during read or write cycle. This pin may be used as Write Byte Enable if programmed so.
OE*			(Output Enable)	This active-low output signal is used to indicate that the bus access is a read and enables slave devices to drive the data bus with read data. OE is negated during hardware reset.
EBW*	PE3	Reset Default	(EIM Data Bus CS0 Default Bus Width Select/PE3)	This active-low input signal controls the selected data bus bandwidth of CS0 at reset. In production, an internal pulldown selects only D[15:0]. NOTE: EBW* function is not supported in LTE. AEIM CS0* defaults at reset to 16-bit bus width. PE3 still controls general purpose I/O for this pin.

2.2.2 Real Time Trace Signals

NOTE:

$\overline{\text{DB\_STRB}}$, DB_SIZ[1:0], DB_SHOW_RW, QSTAT[4:0] are available in the development package only.

Table 2-3. Real Time Trace Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
CKO	PA0	Reset Default	(Clock output)	External visibility of internal clocks as selected by Clock Control Module (CCM).
DB_STRB			(Debug Strobe)	output strobe per memory cycle access.
DB_SZ0-DBSZ1			(Debug Access Size)	Encoded data size for current MCU accesses.
DB_SHOW_RW			(Debug Show Read/Write)	External visibility to internal bus operations.
QSTAT0-QSTAT4			(Debug State)	Encoded value describing processor state to support external tracing of MCU execution.

2.2.3 Layer 1 Timer Signals

Seven (TOUT[12:7,5]) of the L1T pins are dedicated as primary pin functions. The other nine (TOUT[15:13], TOUT[6,4:0]) are multiplexed on other pins or are used internally within Neptune.

Table 2-4. Layer 1 Timer Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
TOUT5	PA2	Reset Default	(Timer output 5)	Layer 1 timer output
TOUT7	PA4	Reset Default	(Timer output 7)	Layer 1 timer output
TOUT8	PA5	Reset Default	(Timer output 8)	Layer 1 timer output
TOUT9	PA6	Reset Default	(Timer output 9)	Layer 1 timer output
TOUT10	PA7	Reset Default	(Timer output 10)	Layer 1 timer output
TOUT11	PA8	Reset Default	(Timer output 11)	Layer 1 timer output
TOUT12	PE1	Reset Default	(Timer output 12)	Layer 1 timer output
TOUT0	PD8	Mux Alternate	(Timer output 0)	Layer 1 timer output
TOUT0	PA5	Mux Alternate	(Timer output 0)	Layer 1 timer output
TOUT1	PD9	Mux Alternate	(Timer output 1)	Layer 1 timer output
TOUT1	PA6	Mux Alternate	(Timer output 1)	Layer 1 timer output
TOUT12	PD3	Mux Alternate	(Timer output 12)	Layer 1 timer output
TOUT13	PD4	Mux Alternate	(Timer output 13)	Layer 1 timer output
TOUT14	PD5	Mux Alternate	(Timer output 14)	Layer 1 timer output
TOUT15	PD6	Mux Alternate	(Timer output 15)	Layer 1 timer output

Table 2-4. Layer 1 Timer Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
TOUT15	PE13	Mux Alternate	(Timer output 15)	Layer 1 timer output
TOUT2	PD1	Mux Alternate	(Timer output 2)	Layer 1 timer output
TOUT3	PD2	Mux Alternate	(Timer output 3)	Layer 1 timer output
TOUT4	PD10	Mux Alternate	(Timer output 4)	Layer 1 timer output
TOUT6	PA12	Mux Alternate	(Timer output 6)	Layer 1 timer output

2.2.4 A2DIGL Mixed Signal Interface

Table 2-5. A2DIGL Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
RX_EN_OUT	PA1	Reset Default	(RX_EN_output)	Receive enable signal to Algae
GSM*_DCS	PA3	Reset Default	(GSM*_DCS)	GSM/DCS band select signal (to Athena)

2.2.5 External Interrupt Signals

Table 2-6. External Interrupt Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
INT0	PA9	Reset Default	(External Interrupt 0)	Bi-directional (Schmitt trigger input) pins used as an external interrupt input. The interrupt can be programmed to be level sensitive or edge triggered (positive-edge or negative-edge). When edge triggered, triggering occurs at a voltage level and is not directly related to the rise or fall time of the interrupt signal. However, as the rise or fall time of the interrupt signal increases, the probability to generate multiple interrupts due to this ìnoiseî also increases.
INT1	PA10	Reset Default	(External Interrupt 1)	Bi-directional (Schmitt trigger input) pins used as an external interrupt input. The interrupt can be programmed to be level sensitive or edge triggered (positive-edge or negative-edge). When edge triggered, triggering occurs at a voltage level and is not directly related to the rise or fall time of the interrupt signal. However, as the rise or fall time of the interrupt signal increases, the probability to generate multiple interrupts due to this ìnoiseî also increases.
INT3	PA12	Reset Default	(External Interrupt 3)	Bi-directional (Schmitt trigger input) pins used as an external interrupt input. The interrupt can be programmed to be level sensitive or edge triggered (positive-edge or negative-edge). When edge triggered, triggering occurs at a voltage level and is not directly related to the rise or fall time of the interrupt signal. However, as the rise or fall time of the interrupt signal increases, the probability to generate multiple interrupts due to this ìnoiseî also increases.
INT4	PA13	Reset Default	(External Interrupt 4)	Bi-directional (Schmitt trigger input) pins used as an external interrupt input. The interrupt can be programmed to be level sensitive or edge triggered (positive-edge or negative-edge). When edge triggered, triggering occurs at a voltage level and is not directly related to the rise or fall time of the interrupt signal. However, as the rise or fall time of the interrupt signal increases, the probability to generate multiple interrupts due to this ìnoiseî also increases.

Table 2-6. External Interrupt Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
INT5			(External Interrupt 5)	Bi-directional (Schmitt trigger input) pins used as an external interrupt input. The interrupt can be programmed to be level sensitive or edge triggered (positive-edge or negative-edge). When edge triggered, triggering occurs at a voltage level and is not directly related to the rise or fall time of the interrupt signal. However, as the rise or fall time of the interrupt signal increases, the probability to generate multiple interrupts due to this ìnoiseî also increases.
INT6	PC9	Reset Default	(External Interrupt 6)	Bi-directional (Schmitt trigger input) pins used as an external interrupt input. The interrupt can be programmed to be level sensitive or edge triggered (positive-edge or negative-edge). When edge triggered, triggering occurs at a voltage level and is not directly related to the rise or fall time of the interrupt signal. However, as the rise or fall time of the interrupt signal increases, the probability to generate multiple interrupts due to this ìnoiseî also increases.
INT2	PA11	Mux Alternate	(External Interrupt 2)	Bi-directional (Schmitt trigger input) pins used as an external interrupt input. The interrupt can be programmed to be level sensitive or edge triggered (positive-edge or negative-edge). When edge triggered, triggering occurs at a voltage level and is not directly related to the rise or fall time of the interrupt signal. However, as the rise or fall time of the interrupt signal increases, the probability to generate multiple interrupts due to this ìnoiseî also increases.
INT4	PE3	Mux Alternate	(External Interrupt 4)	Bi-directional (Schmitt trigger input) pins used as an external interrupt input. The interrupt can be programmed to be level sensitive or edge triggered (positive-edge or negative-edge). When edge triggered, triggering occurs at a voltage level and is not directly related to the rise or fall time of the interrupt signal. However, as the rise or fall time of the interrupt signal increases, the probability to generate multiple interrupts due to this ìnoiseî also increases.

Table 2-6. External Interrupt Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
INT5	PA12	Mux Alternate	(External Interrupt 5)	Bi-directional (Schmitt trigger input) pins used as an external interrupt input. The interrupt can be programmed to be level sensitive or edge triggered (positive-edge or negative-edge). When edge triggered, triggering occurs at a voltage level and is not directly related to the rise or fall time of the interrupt signal. However, as the rise or fall time of the interrupt signal increases, the probability to generate multiple interrupts due to this ìnoiseî also increases.
INT7	PC2	Mux Alternate	(External Interrupt 7)	Bi-directional (Schmitt trigger input) pins used as an external interrupt input. The interrupt can be programmed to be level sensitive or edge triggered (positive-edge or negative-edge). When edge triggered, triggering occurs at a voltage level and is not directly related to the rise or fall time of the interrupt signal. However, as the rise or fall time of the interrupt signal increases, the probability to generate multiple interrupts due to this ìnoiseî also increases.

2.2.6 SAP Signals

Table 2-7. SAP Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
SCKA	PB1	Reset Default	(SCKA, Serial Clock)	Bi-directional (Schmitt trigger input) pin providing the serial bit rate clock for the interface. The SCKA is a clock input or output of the SAP used by both the transmitter and receiver in synchronous modes or by the transmitter in asynchronous modes.
STDA	PB2	Reset Default	(SAP Serial Transmit Data Pin)	is used for transmitting data from the SAP serial transmit shift register. STDA is an output when data is being transmitted.
SRDA	PB4	Reset Default	(SAP Serial Receive Data Pin)	receives serial data and transfers the data to the SAP receive shift register. SRDA is input when data is being received.
SC2A	PB3	Reset Default	(SAP Serial Control Pin 2)	This pin is used for frame sync I/O. SC2A is the SAP frame sync for both the transmitter and receiver in synchronous mode and for the transmitter only in asynchronous mode.

Table 2-7. SAP Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
SC1A	PB0	Reset Default	(SAP Serial Control Pin 1)	The function of this pin is determined by the selection of either synchronous or asynchronous mode. In asynchronous mode, this pin is the receiver frame sync I/O. For synchronous mode, this pin is used for serial I/O flag 1. SC1A may be programmed as a general purpose I/O pin when the SC1A function is not being used.
SC0A	PA15	Reset Default	(SAP Serial Control Pin 0)	Input pin with Schmitt trigger buffer whose function is determined by the selection of either synchronous or asynchronous mode. For asynchronous mode, this pin will be used for the receive clock I/O. For synchronous mode, this pin is used for serial input flag 0. SC0A may be programmed as a general purpose I/O pin when the SC0A function is not being used.
SC2A	PB0	Mux Alternate	(SAP Serial Control Pin 2)	This pin is used for frame sync I/O. SC2A is the SAP frame sync for both the transmitter and receiver in synchronous mode and for the transmitter only in asynchronous mode.
SRDA	PE12	Mux Alternate	(SAP Serial Receive Data Pin)	receives serial data and transfers the data to the SAP receive shift register. SRDA is input when data is being received.
STDA	PE11	Mux Alternate	(SAP Serial Transmit Data Pin)	is used for transmitting data from the SAP serial transmit shift register. STDA is an output when data is being transmitted.

2.2.7 GPIO Signals

Table 2-8. GPIO Signals

Signal	Port	Reset Default?	Desc	Definition
PA0-PA15			(Port A)	General Purpose Input/Output accessible for either DSP or MCU I/O.
PB0-PB15			(Port B)	General Purpose Input/Output accessible for either DSP or MCU I/O.
PC0-PC15			(Port C)	General Purpose Input/Output accessible for either DSP or MCU I/O.
PC12	PC12	Reset Default	(Port C 12)	General Purpose Input/Output accessible for either DSP or MCU I/O.
PC14	PC14	Reset Default	(Port C 14)	General Purpose Input/Output accessible for either DSP or MCU I/O.
PC14	PC14	Reset Default	(Port C 14)	General Purpose Input/Output accessible for either DSP or MCU I/O.
PD0-PD15			(Port D)	General Purpose Input/Output accessible for either DSP or MCU I/O.

Table 2-8. GPIO Signals

Signal	Port	Reset Default?	Desc	Definition
PE0-PE15			(Port E)	General Purpose Input/Output accessible for either DSP or MCU I/O.
PE10	PE10	Reset Default	(Port E 10)	General Purpose Input/Output accessible for either DSP or MCU I/O.
PE11	PE11	Reset Default	(Port E 11)	General Purpose Input/Output accessible for either DSP or MCU I/O.
PE12	PE12	Reset Default	(Port E 12)	General Purpose Input/Output accessible for either DSP or MCU I/O.

2.2.8 SIM Signals

Table 2-9. SIM Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
SIM_RST	PB7	Reset Default	(SIM0 Reset)	The SIM Reset pin is an active-low reset output. The Smart Card Port can activate a reset to the SmartCard by driving SIM Reset pin to low. This pin may be programmed as a general purpose I/O pin when the SIM function is not being used.
SIM_PD	PB8	Reset Default	(SIM Presence Detect)	SIM Presence Detect pin is an input Schmitt trigger which signals the insertion or removal of the SmartCard. This pin may be programmed as a general purpose I/O pin when the SIM function is not being used.
SIM_CLK	PB9	Reset Default	(SIM Clock)	The SIM clock pin is an output clock from the Smart Card Port to the Smart Card. This pin may be programmed as a general purpose I/O pin when the SIM function is not being used.
SIM_DIO	PB10	Reset Default	(SIM Transmit and Receive Data)	Transmit and Receive Data line for SIM Port0. The default state at reset is active-low output.
SIM_VCCEN	PB11	Reset Default	(SIM VCC Enable)	SIM VCC Enable is an active-high enable to an external device which supplies VCC to the card. If the SIM interface drives this pin high the external device will supply Vcc to the card. If the SIM Interface drives this pin low then the external device will power off the Vcc supply to the card. This pin may be programmed as a general purpose I/O pin when the SIM function is not being used.

2.2.9 Display Memory Access Controller Signals

Table 2-10. DMAC Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
SDATA	PB12	Reset Default	(LCD Serial Data output)	Serial/Parallel Data output to LCD Driver
LCD_CLK	PB13	Reset Default	(LCD Serial Data Input)	Serial Clock/Parallel Data output to LCD Driver
LCD_DATA[5]	PE9	Reset Default	(LCD Parallel Data Output)	The LCD Data bus is used to write information to external LCD controller
LCD_DATA[4]	PE8	Reset Default	(Parallel Display Data Output)	The LCD Data bus is used to write information to external LCD controller
LCD_DATA[3]	PE7	Reset Default	(LCD Parallel Data Output)	The LCD Data bus is used to write information to external LCD controller
LCD_DATA[2]	PE6	Reset Default	(LCD Parallel Data Output)	The LCD Data bus is used to write information to external LCD controller
LCD_DATA[1]	PE5	Reset Default	(LCD Parallel Data Output)	The LCD Data bus is used to write information to external LCD controller
LCD_DATA[0]	PE4	Reset Default	(Parallel Data output to LCD)	The LCD Data bus is used to write information to external LCD controller
LCD_CS	PB14	Reset Default	(Chip Select for LCD Driver)	Active-low chip select for the external display controller.
LCD_RS	PB15	Reset Default	(Register Select for LCD Driver)	LCD Register Select signal that indicates to the LCD device whether the data being written is display data or control data.
LCD_DATA[0]	PD6	Mux Alternate	(LCD Parallel Data Output)	The LCD Data bus is used to write information to external LCD controller
LCD_DATA[1]	PD7	Mux Alternate	(Parallel Data output to LCD)	The LCD Data bus is used to write information to external LCD controller
LCD_DATA[2]	PA5	Mux Alternate	(LCD Parallel Data Output)	The LCD Data bus is used to write information to external LCD controller
LCD_DATA[2]	PC12	Mux Alternate	(LCD Parallel Data Output)	The LCD Data bus is used to write information to external LCD controller
LCD_DATA[3]	PA15	Mux Alternate	(LCD Parallel Data Output)	The LCD Data bus is used to write information to external LCD controller
LCD_DATA[3]	PE14	Mux Alternate	(LCD Parallel Data Output)	The LCD Data bus is used to write information to external LCD controller

Table 2-10. DMAC Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
LCD_DATA[4]	PB0	Mux Alternate	(LCD Parallel Data Output)	The LCD Data bus is used to write information to external LCD controller
LCD_DATA[4]	PC13	Mux Alternate	(LCD Parallel Data Output)	The LCD Data bus is used to write information to external LCD controller
LCD_DATA[4]	PE3	Mux Alternate	(LCD Parallel Data Output)	The LCD Data bus is used to write information to external LCD controller
LCD_DATA[5]	PC14	Mux Alternate	(LCD Parallel Data Output)	The LCD Data bus is used to write information to external LCD controller
LCD_DATA[5]	PA6	Mux Alternate	(LCD Parallel Data Output)	The LCD Data bus is used to write information to external LCD controller

2.2.10 Keypad Port Signals

NOTE:

All keypad port signals have Schmitt trigger inputs.

Table 2-11. KPP Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
COLUMN0	PC3	Reset Default	(Keypad Column Strobe 0)	This pin is used as a Keypad Column Strobe. This pin can be designated as open drain output by writing a one into the appropriate bit in the Keypad Port Control Register.
COLUMN1	PC4	Reset Default	(Keypad Column Strobe 1)	This pin is used as a Keypad Column Strobe. The 8 most significant bits (15-8) can be designated as open drain outputs by writing a one into the appropriate bits in the Keypad Port Control Register.
ROW0	PC5	Reset Default	(Row Sense 0)	This pin is used as a Keypad Row Sense.
ROW1	PC6	Reset Default	(Row Sense 1)	This pin is used as a Keypad Row Sense.
ROW2			(Row Sense 2)	This pin is used as a Keypad Row Sense.
ROW3			(Row Sense 3)	This pin is used as a Keypad Row Sense.
ROW4	PC7	Reset Default	(Row Sense 4)	This pin is used as a Keypad Row Sense.

Table 2-11. KPP Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
ROW5	PC8	Reset Default	(Row Sense 5)	This pin is used as a Keypad Row Sense.
ROW6	PC10	Reset Default	(Row Sense 6)	This pin is used as a Keypad Row Sense.
ROW7	PC11	Reset Default	(Row Sense 7)	This pin is used as a Keypad Row Sense.

2.2.11 Real Time Clock Signals

Table 2-12. RTC Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
CKIL			(Low Frequency Clock Input)	Also called RTC_CLK.It is a 32,768Hz square wave clock input.
STANDBY			(Standby and Time-Of-Day Interrupt)	When the IC is fully powered, this pin functions as the Deep Sleep Module standby signal output. When only the RTC is powered, this pin functions as the RTC's TODA_INT output.

2.2.12 Clock Control Signals

Table 2-13. CCM Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
XTAL			(Differential Crystal Input)	26 MHz reference clock differential. This will be the default clock input at reset.
EXTAL			(Differential Crystal Output)	26 MHz reference clock differential.
TRKG_OSC_OUT			(Tracking Oscillator Output)	Gated 26 MHz output to Algae for RC tuning.
CCM_USB_CLK	PC8	Mux Alternate	(USB Clock)	External input clock for USB (development selection)
CKOH	PA14	Mux Alternate	(Clock output)	Output visibility of various system clocks selected by the CCM.

Table 2-13. CCM Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
CKOH	PC2	Mux Alternate	(Clock output)	Output visibility of various system clocks selected by the CCM.
REF_CLK	PA3	Mux Alternate	(MCU PLL Bypass)	PLL Reference Bypass clock for MCU

2.2.13 RF Interface Signals

Table 2-14. RF Interface Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
RF_CLK	PD0	Reset Default	(RF Serial Transfer Clock)	Pass-through SPI clock
RF_DATA	PD1	Reset Default	(RF Serial Transfer Data)	Pass-through SPI data
RF_CS	PD2	Reset Default	(RF Serial Transfer Chip Select)	Pass-through CS
RXSDG_DET_FLAG	PA9	Mux Alternate	(RXSDG_DET_FLAG)	output visibility to RXSDG Detect Flag. Also, input redirection for this signal to be readable to the DSP via the GPIO registers.

2.2.14 MQSPI Signals

Table 2-15. MQSPI Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
QSCKA	PD3	Reset Default	(MQSPI Serial Clock)	This output pin provides the serial clock from the MQSPI. There is a programmable number of clock cycles delay between the assertion of the chip select signal and the first transition of the serial clock. The polarity and phase of the serial clock are programmable.
MISOA	PD4	Reset Default	(MQSPI Master Data In / Slave Data output)	This input signal provides serial data input to the MQSPI. Data input can be programmed to be sampled by the rising or falling edge of QSCK, and each byte is written to RAM as least significant bit first.

Table 2-15. MQSPI Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
MOSIA	PD5	Reset Default	(MQSPI Master Data output/ Slave Data In)	This output signal provides serial data output from the MQSPI. Data output can be programmed to be changed on the falling or rising edge of QSCK, and each byte is transmitted least significant bit first.
MISOB	PE14	Reset Default	(MQSPI Master Data In / Slave Data output)	This input signal provides serial data input to the MQSPI SPI2. Data input can be programmed to be sampled by the rising or falling edge of QSCK, and each byte is written to RAM as least significant bit first.
SPI_CS0	PD6	Reset Default	(MQSPI Chip Select 0)	This output pin provides an MQSPI peripheral chip select. The chip select can be programmed to be active-high or active-low.
SPI_CS1	PD7	Reset Default	(MQSPI Chip Select 1)	This output pin provides an MQSPI peripheral chip select. The chip select can be programmed to be active-high or active-low.
SPI_CS2	PD8	Reset Default	(MQSPI Chip Select 2)	This output pin provides an MQSPI peripheral chip select. The chip select can be programmed to be active-high or active-low.
SPI_CS3	PD9	Reset Default	(MQSPI Chip Select 3)	This output pin provides an MQSPI peripheral chip select. The chip select can be programmed to be active-high or active-low.
SPI_CS6	PC15	Reset Default	(MQSPI Chip Select 6)	This output pin provides MQSPI peripheral chip select 6. The chip select can be programmed to be active-high or active-low.
SPI_CS8	PE2	Reset Default	(MQSPI Chip Select 8)	This output pin provides an MQSPI peripheral chip select. The chip select can be programmed to be active-high or active-low.
MISOB	PA12	Mux Alternate	(MQSPI Master Data In / Slave Data output)	This input signal provides serial data input to the MQSPI SPI2. Data input can be programmed to be sampled by the rising or falling edge of QSCK, and each byte is written to RAM as least significant bit first.
SPI_CS4	PA15	Mux Alternate	(MQSPI Chip Select 4)	This output pin provides an MQSPI peripheral chip select. The chip select can be programmed to be active-high or active-low.
SPI_CS5	PD14	Mux Alternate	(MQSPI Chip Select 5)	This output pin provides an MQSPI peripheral chip select. The chip select can be programmed to be active-high or active-low.
SPI_CS5	PD5	Mux Alternate	(MQSPI Chip Select 5)	This output pin provides an MQSPI peripheral chip select. The chip select can be programmed to be active-high or active-low.
SPI_CS6	PD2	Mux Alternate	(MQSPI Chip Select 6)	This output pin provides MQSPI peripheral chip select 6. The chip select can be programmed to be active-high or active-low.

Table 2-15. MQSPI Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
SPI_CS7	PD11	Mux Alternate	(MQSPI Chip Select 7)	This output pin provides an MQSPI peripheral chip select. The chip select can be programmed to be active-high or active-low.
SPI_CS7	PC7	Mux Alternate	(MQSPI Chip Select 7)	This output pin provides an MQSPI peripheral chip select. The chip select can be programmed to be active-high or active-low.
SPI_CS8	PD12	Mux Alternate	(MQSPI Chip Select 8)	This output pin provides an MQSPI peripheral chip select. The chip select can be programmed to be active-high or active-low.
SPI_CS8	PC8	Mux Alternate	(MQSPI Chip Select 8)	This output pin provides an MQSPI peripheral chip select. The chip select can be programmed to be active-high or active-low.
SPI_CS9	PD13	Mux Alternate	(MQSPI Chip Select 9)	This output pin provides an MQSPI peripheral chip select. The chip select can be programmed to be active-high or active-low.
SPI_CS9	PC10	Mux Alternate	(MQSPI Chip Select 9)	This output pin provides an MQSPI peripheral chip select. The chip select can be programmed to be active-high or active-low.

2.2.15 JTAG Signals

Table 2-16. JTAG Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
TDI			(Test Data Input)	The test data input TDI pin is the serial input for test instructions and data. TDI is sampled on the rising edge of TCK.
TDO			(Test Data output)	The test data output TDO pin is the serial output for test instructions and data. TDO is tri-stateable and is actively driven in the shift IR and shift DR controller states. TDO changes on the falling edge of TCK.
TRST			(Test Reset)	This active-low Schmitt trigger input pin TRST is used to asynchronously initialize the test controller.
TMS			(Test Mode Select)	The test mode select input (TMS) pin is used to sequence the test controller is state machine. The TMS is sampled on the rising edge of TCK.
TCK			(Test Clock)	The test clock input TCK pin is the test clock used to synchronize the JTAG test logic.
RTCK	PC1	Reset Default	(Return Test Clock)	Return synchronization test clock to ARM development tools.

Table 2-16. JTAG Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
MCU_DE*			(MCU Debug Event)	As an input, this open-drain, bi-directional, active-low pin provides, a means to enter the debug mode of operation from an external command controller. As an output, it is a means to acknowledge that the MCU has entered the debug mode. This pin when asserted as an input causes the MCU Core to finish the current instruction being executed, save the instruction pipeline information, enter the debug mode and wait for commands to be entered from the serial debug input line. This pin is asserted as an output for several clock cycles when the MCU enters debug mode as a result of a debug request or as a result of meeting a breakpoint condition. If MCU_DE was used to enter the Debug Mode then MCU_DE must be negated after the debug logic responds with an acknowledge and before sending the first command. When an asserted transition occurs on the MCU_DE pin while the MCU TAP controller is in the Test-Logic-Reset state, and the TRST input remains negated then the debug configuration is selected on this port. The debug configuration will remain in effect until the next assertion of TRST. This pin can be connected directly to DSP_DE external to Neptune to allow the DSP debug event to break the MCU.
DSP_DE*			(DSP Debug Event)	This open drain bi-directional active-low pin provides, as an input, a means of entering the debug mode of operation from an external command controller, and as an output, a means of acknowledging that the DSP has entered the debug mode. This pin when asserted as an input causes the DSP56600 core to finish the current instruction being executed, save the instruction pipeline information, enter the debug mode and wait for commands to be entered from the serial debug input line. This pin is asserted by the DSP56600 core as an output for three clock cycles when the DSP enters debug mode as a result of a debug request or as a result of meeting a breakpoint condition. If DSP_DE was used to enter the Debug Mode then DSP_DE must be negated after the OnCEô responds with an acknowledge and before sending the first OnCEô command. This pin can be connected directly to MCU_DE external to Neptune to allow the MCU debug event to break the DSP.
SJC_MOD	PC13	Reset Default	(SJC_MOD)	Select for ARM or joint DSP/ARM JTAG mode.

2.2.16 Miscellaneous Control Signals

Table 2-17. Miscellaneous Control Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
RESET_IN*			(Reset In)	An active-low Schmitt trigger input that provides reset to the internal circuitry. The RESET input will be qualified as valid if it will be asserted for at least three CKIL clock cycles.
RESET_OUT*			(Reset Out)	This is an active-low output that reflects the state of internal system reset.
WDOG*	PA14	Reset Default	(Watchdog output)	An active-low output that can be tri-stated, asserted under one of the following conditions: 1) watchdog count expires; 2) clock monitor event detected; or 3) a watchdog software event is detected.
MOD			(Boot Mode Select)	MOD selects MCU boot mode during hardware reset. The state of the MOD pin is latched when RESET is negated. If this pin is driven at a logic-high four (4) CKIL clock cycles before RESET negation, then the internal MCU ROM will be enabled and program execution will begin from internal memory. If this pin is driven at a logic-low at least four (4) CKIL clock cycles before RESET negation, then program execution will begin from external memory. For external memory execution on RESET, AEIM CS0 is mapped to system memory beginning at address 0x00000000.
TEST0			(Factory Test Mode 0)	The Factory Test Mode input is used to select the Factory Test Mode. The detailed description of the functionality of this pin is detailed in the Neptune Test Control Module (TCM) chapter.
TEST1			(Factory Test Mode 1)	Addition Factory Test Mode configuration selection.
TEST2			(Factory Test Mode 2)	Addition Factory Test Mode configuration selection.

2.2.17 USB Signals

Table 2-18. USB Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
USB_TXENB	PD10	Reset Default	(USB Transmit Enable)	Active Low Transmit Enable. This signal is an input to the USB transceiver module to signal when to transmit data on the USB bus.
USB_VPIN	PD11	Reset Default	(USB V+ Data In)	USB data in from USB Transceiver V+ Single-ended receiver

Table 2-18. USB Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
USB_VMIN	PD12	Reset Default	(USB V-Data In)	USB data in from USB Transceiver V-Single-ended receiver.
USB_XRXD	PD13	Reset Default	(USB_XRXD)	CMOS logic value of value received from USB wires.
USB_VPOUT	PD14	Reset Default	(USB V+ Data output)	USB data output from Neptune USB module to USB transceiver for V+
USB_VMOUT	PD15	Reset Default	(USB V-Data output)	USB data output from Neptune USB module to USB transceiver for V
USB_SUSPEND	PA11	Reset Default	(USB Suspend)	This signal is active-high and signifies the USB device is suspended. This signal is intended to be used to deactivate system circuitry (including the USB transceiver) in order to enter a low power state.

2.2.18 RX and TX Synthesizers

Table 2-19. RX and TX Synthesizer Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
VDD_CP			(Charge Pump VDD)	Power for Transmit/receive synthesizer.
VSS_CP			(Charge Pump VSS)	Local ground for transmit/receive synthesizer.
REG_BYP_PRE			()	Regulated bypass capacitor for high frequency prescaler supply.
VSS_PRE			(Prescaler VSS)	Local ground for high frequency prescaler.
TX_CP			(Transmit synthOutput)	Filtered output of the transmit FracN Synthesizer.
RX_CP			(Receiver Synth Output)	Filtered output of the receiver FracN Synthesizer.
CP_BIAS			(Charge Pump Bias)	Bias developed across external precision resistor for internal transmitter/receiver charge pump.
VCO1_N			(TX_VCO)	Transmit/receiver VCO input frequency differential level from Algae into synthesizer prescaler divider.
VCO1_P			(TX_VCO)	Transmit/receiver VCO input frequency differential level from Algae into synthesizer prescaler divider.
TX_MOD			(Transmit Modulation)	High frequency data output of dual port modulation. Along with TX_CP, this controls the transmit VCO to create a modulated carrier.

2.2.19 UART Signals

Table 2-20. UART Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
CTS1	PC0	Reset Default	(UART1 Clear To Send)	UART1 Clear to Send output
TXD2	PB5	Reset Default	(UART2 Transmit)	UART2 Transmit Data
RXD2	PB6	Reset Default	(UART2 Receive)	UART2 Receive Data
CTS2	PE13	Reset Default	(UART2 Clear To Send)	UART2 Clear to Send output
RTS2	PE15	Reset Default	(UART2 Request To Send)	UART2 Request to Send
URTS1	PE0	Reset Default	(UART1 Request To Send)	UART1 Request to Send (Dedicated copy, not routed through GPIO)
CTS1	PD15	Mux Alternate	(UART1 Clear To Send)	UART1 Clear to Send output
CTS1	PA9	Mux Alternate	(UART1 Clear To Send)	UART1 Clear to Send output
DCD1	PA11	Mux Alternate	(UART1 Data Carrier Detect)	UART1 Data-Carrier-Detect
DSR1	PD10	Mux Alternate	(UART1 Data Set Ready)	UART1 Data-Set-Ready
DSR1	PE11	Mux Alternate	(UART1 Data Set Ready)	UART1 Data-Set-Ready
DTR1	PD11	Mux Alternate	(UART1 Data Terminal Ready)	UART1 Data-Terminal Ready
DTR1	PE12	Mux Alternate	(UART1 Data Terminal Ready)	UART1 Data-Terminal Ready
RI1	PC14	Mux Alternate	(UART1 Ring Indicator)	UART1 Ring Indicator
RI1	PE10	Mux Alternate	(UART1 Ring Indicator)	UART1 Ring Indicator

Table 2-20. UART Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
RTS1	PD13	Mux Alternate	(UART1 Request To Send)	UART1 Request to Send
RTS1	PE12	Mux Alternate	(UART1 Request To Send)	UART1 Request to Send
RXD1	PD12	Mux Alternate	(UART1 Receive)	UART1 Receive Data
TXD1	PD14	Mux Alternate	(UART1 Transmit)	UART1 Transmit Data

2.2.20 PA Control

Table 2-21. PA Control Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
PA_OUT			(PA Output)	Control signal to Athena Power Amplifier. Formerly known as Automatic output level Control Driver. Controls the final amplifier to the antenna.
PA_REF			(PA Reference)	Reference feedback from Athena Power Amplifier.
PA_DET			(PA Detect	Antenna power detect feedback from Athena Power Amplifier.
LOW_BATT*			(Low Battery Detected)	Low Battery signal from Seaweed. This active-low input signals the detection of an under voltage, which can cause the PA_output to drive low to disable the Athena Power Amplifier, and enable the RTC's power cut timer.
ATST_P			(Analog Test I/O)	Positive differential input or output for test. Function determined by ANATEST module.
ATST_N			(Analog Test I/O)	Negative differential input or output for test. Function determined by ANATEST module.

2.2.21 RX Data Converters

Table 2-22. RX Data Converter Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
VDD_TXRX			(TXRX Regulator VDD)	Power supply input to TXRX regulator. Input should be 2.775 V from Seaweed Digital I/O regulator. This should be routed through a star configuration to isolate it from Neptune's digital I/O.
VSS_TXRX			(TXRX Regulator VSS)	Local ground input for TXRX regulator. This should be routed through a star configuration to isolate it from Neptune's digital I/O.
REG_BYP_TXRX			(TXRX Supply Bypass Output)	Supply output of TXRX regulator for externally bypassing. Place external capacitor between this pin and VSS_TXRX.
VAG_RX			(Receiver Analog Decoupling)	External decoupling for the receiver analog ground. Place external capacitor between this pin and VSS_RXTX.
RX_I			(RX Input)	Positive in-phase channel of received signal from algae. This may be very low IF or baseband signal.
RX_IX			(RX Input)	Negative in-phase channel of received signal from algae. This may be very low IF or baseband signal.
RX_Q			(RX Quad Input)	Positive quadrature-phase channel of received signal from algae. This may be very low IF or baseband signal.
RX_QX			(RX Quad Input)	Negative quadrature-phase channel of received signal from algae. This may be very low IF or baseband signal.
REF_HI			(RX ADC Decoupling)	External decoupling for analog reference high for receiver ADC. Place external capacitor between this pin and VSS_RXTX.
REF_LO			(RX ADC Decoupling)	External decoupling for analog reference low for receiver ADC. Place external capacitor between this pin and VSS_RXTX.

2.2.22 Audio Codec

Table 2-23. Audio Codec Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
VDD_CODEC			(Codec Regulator VDD)	Power supply input to CODEC regulator. Input should be 2.775 V from Seaweed Audio regulator.
VSS_CODEC			(Codec Regulator VSS)	Local ground input for CODEC regulator.

Table 2-23. Audio Codec Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
REG_BYP_CODEC			(Codec Regulator Bypass)	Power supply output of CODEC regulator for external bypassing capacitor. Place external capacitor between this pin and VSS_CODEC.
VAG_CODEC			(Codec Analog ground decoupling)	External decoupling for the CODEC analog ground. Place external capacitor between this pin and VSS_CODEC.
VAGOUT_CODEC			(Buffered Codec Analog Ground)	Buffered version of CODEC analog ground to Seaweed. Place external capacitor between this pin and VSS_CODEC.
REF_CODEC_P			(Codac Reference Bypass)	Codec reference differential bypass.
REF_CODEC_N			(Codac Reference Bypass)	Codec reference differential bypass.
ADC_N			(ADC Input)	Negative differential input to ADC from Seaweed. This is voice band signal.
ADC_P			(ADC Input)	Positive differential input to ADC from Seaweed. This is voice band signal.
DAC_N			(DAC Output)	Negative differential output of DAC to Seaweed. This is voice band signal.
DAC_P			(DAC Output)	Positive differential output of DAC to Seaweed. This is voice band signal.

2.2.23 General Purpose A/D

Table 2-24. GPADC Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
ADC_DATA			(ADC Input)	ADC input signal from Seaweed. Seaweed can mux several low frequency analog signals for measurement.
ADC_SYNC	PC2	Reset Default	(ADC Sync)	Sync signal to Seaweed to control the mux.

2.2.24 Power and Ground

Table 2-25. Power and Ground Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
VDDA			(Oscillator VDD)	Supply for CROSC Circuit. Input should be 2.775 V from Seaweed I/O regulator.
VSSA			(Oscillator VSS)	Local ground input for CROSC Circuit
AVDD			(Group A Supply - 7 pins)	Supply for External Memory Interface Pins. Input should be 1.875 V from Seaweed DIO regulator.
AVSS			(Group A Ground - 7 pins)	Local ground input for External Memory Interface Pins.
BVDD			(Group B Supply)	Supply for Seaweed Data Pins. Input should be 2.775 V from Seaweed I/O regulator.
BVSS			(Group B Ground)	Local ground input for Seaweed Data Pins.
CVDD			(Group C Supply - 4 pins)	Supply for LCD, Serial I/O, and Test Pins. Input should be 2.775 V from Seaweed I/O regulator.
CVSS			(Group C Ground - 4 pins)	Local ground input for LCD and Test Pins.
DVDD			(Group D Supply)	Supply for Seaweed Control Pins. Input should be 2.775 V from Seaweed I/O regulator.
DVSS			(Group D Ground)	Local ground input for Seaweed Control Pins.
EVDD			(Group E Supply -2 pins)	Supply for Algae Control, Serial I/O, and Data Pins. Input should be 2.775 V from Seaweed I/O regulator.
EVSS			(Group E Ground - 2 pins)	Local ground input for Algae Control and Data Pins.
FVDD			(Group F Supply)	Supply for SIM Pins. Special power down 2.8V supply. Input should be 2.85 V from Seaweed SIM regulator.
FVSS			(Group F Ground)	Local ground input for SIM Pins.
JVDD			(Group J Supply)	Supply for Keypad and Low Frequency Control Pins. Also supply for Analog portion of General Purpose ADC, ANATEST, and parts of TXRX Regulator and Core Regulator. Input should be 2.775 V from Seaweed I/O regulator.
JVSS			(Group J Ground)	Local ground input for Keypad and Low Frequency Control Pins. Also supply for Analog portion of General Purpose ADC.

Table 2-25. Power and Ground Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
LVDD			(Group L Supply)	Supply for RTC Pins and Circuit. Special standby 1.575V supply. Also analog reference for all analog circuits. Input should be 1.575 V from Seaweed REF regulator.
LVSS			(Group L Ground)	Local ground input for RTC Pins and Circuit. Special standby 1.575V supply. Also analog reference for all analog circuits.
QVDD			(Group Q Supply)	Input Supply for Core Regulator. Input should be 1.875 V from Seaweed DIG regulator.
QVSS			(Group Q Ground)	Local ground input for Core Regulator.
REG_BYP_CORE			(Core Regulator bypass connection)	External decoupling for output of Core Regulator. Place external bulk bypass capacitor between this pin and QVSS. Supply for all digital circuits except RTC.
VDD			(VDD Decoupling - 4 pins)	External decoupling for output of Core Regulator. Brought to four pairs around die. Place high frequency bypass capacitors to VSS at each pair. Supply for all digital circuits except RTC.
VSS			(VSS Decoupling - 4 pins)	External decoupling for output of Core Regulator. Brought to four pairs around die. Place high frequency bypass capacitors to VSS at each pair. Supply for all digital circuits except RTC.

2.2.25 General Purpose Timer

Table 2-26. EGPT Signals

Signal	GPIO Port (If any)	Reset Default?	Description	Definition
IC1	PA9	Mux Alternate	(Input Capture 1)	An Input Capture Timer channel is connected to this pin.
IC2	PA13	Mux Alternate	(Input Capture 2)	An Input Capture Timer channel is connected to this pin.
IC2	PA13	Mux Alternate	(Input Capture 2)	MCU GPTimer Input Capture
OC1	PC10	Mux Alternate	(output Compare 1)	MCU GPTimer Output Compare Pin
OC2	PC4	Mux Alternate	(Output Compare 2)	MCU GPTimer Output Compare Pin
OC3	PC12	Mux Alternate	(Output Compare 2)	MCU GPTimer Output Compare Pin

Table 2-26. EGPT Signals

Signal	GPIO Port (If any)	Reset Default?	Description	Definition
PWM	PD12	Mux Alternate	(Pulse Width Modulation Output)	This active-high signal is the PWM (Pulse Width Modulator) output pin.
PWM_N	PD11	Mux Alternate	(Pulse Width Modulation Output)	This active-low signal is the PWM (Pulse Width Modulator) output pin.

2.2.26 DSP Debug and Trace

Table 2-27. DSP Debug and Trace Signals

Signal	GPIO Port (If any)	Reset Default?	Description	Definition
DSP_DBG_PAGE0	PD8	Mux Alternate	(DSP Page Bit 0)	DSP Debug. See Table.
DSP_DBG_PAGE1	PD7	Mux Alternate	(DSP Page Bit 1)	DSP Debug. See Table.
DSP_DBG_PAGE2	PC0	Mux Alternate	(DSP Page Bit 2)	DSP Debug. See Table.
DSP_DBG_PAW	PA15	Mux Alternate	(DSP_DBG_PAW)	DSP Debug Program Address Watchpoint
DSP_DBG_XDW	PA6	Mux Alternate	(DSP_DBG_XDW)	DSP Debug X Data Watchpoint
DSP_DBG_YDW	PA5	Mux Alternate	(DSP_DBG_YDW)	DSP Debug Y Data Watchpoint
PGAD[15:0]	Table 23-23 on page 23-119	Mux Alternate	(DSP Program Address 15:0)	DSP Program Address for debug trace.

Table 2-28. DSP Debug Page Visibility

PAGE1	PAGE0	Description
0	0	Page 0 Active
0	1	Page 1 Active
1	0	Page 2 Active
1	1	Page 3 Active

2.2.27 DSP Triple Timer

Table 2-29. DSP Triple Timer Signals

Signal	GPIO Port (If any)	Reset Default ?	Description	Definition
TIN0	PB15	Mux Alternate	(DSP Triple Timer Input 0)	DSP Triple Timer Input
TIN1	PB14	Mux Alternate	(DSP Triple Timer Input 1)	DSP Triple Timer Input
TIN2	PB13	Mux Alternate	(DSP Triple Timer Input 2)	DSP Triple Timer Input
TRIP_TOUT0	PB15	Mux Alternate	(DSP Triple Timer Output 0)	DSP Triple Timer Output
TRIP_TOUT1	PB14	Mux Alternate	(DSP Triple Timer Output 1)	DSP Triple Timer Output
TRIP_TOUT2	PB13	Mux Alternate	(DSP Triple Timer Output 2)	DSP Triple Timer Output

2.2.28 One Wire Interface

Table 2-30. One Wire Interface Signals

Signal	Port	Reset Default?	Desc	Definition
OWB	PA2	Mux Alternate	(One Wire I/O)	One wire data I/O

2.3 Programmable I/O Drive Strength

Table 2-31 describes the I/O pins that are available for programmable output drive strength. See Section 6.8, “IO Pad Drive Programming,” on page 6-72 for information on control register programming of I/O drive strength.

Table 2-31. I/O Drive Groups

Group	I/O pins in 8 mA Drive Group	I/O pins in 16 mA Drive Group	Control Bits	Default at Reset	Notes
1	Address[23:0]	Address[23:0], $\overline{\text{LBA}}$	GPO0:GPO10	4 mA	

Table 2-31. I/O Drive Groups

Group	I/O pins in 8 mA Drive Group	I/O pins in 16 mA Drive Group	Control Bits	Default at Reset	Notes
2	Data[31:0]	Data[31:0]	GPO1:GPO11	4 mA	
3	$\overline{\text{CS}}$[4:0] CS5	$\overline{\text{CS}}$[4:0] CS5	GPO2:GPO12	4 mA	
4	$\overline{\text{EB}}$[3:0] R/$\overline{\text{W}}$ $\overline{\text{OE}}$	$\overline{\text{EB}}$[3:0] R/$\overline{\text{W}}$ $\overline{\text{OE}}$	GPO3:GPO13	4 mA	
5	QSCKA MOSIA		GPO4	4 mA	Power Control IC SPI
6	CKOH CKO		GPO5	4 mA	
7	$\overline{\text{LBA}}$ BURSTCLK	BURSTCLK	GPO6:GPO14	4 mA	
8	TOUT[12:7,5] RX_EN_OUT GSM*_DCS	TOUT[12:7,5] RX_EN_OUT GSM*_DCS	GPO7:GPO16	4 mA	
9	RF_CLK RF_DATA		GPO8	4 mA	IF IC SPI
10	SAP: SCKA, STDA, SC2A BBP: SCKB, STDB, SC2B		GPO9	4 mA	
11	DB_STRB* DB_SZ[1:0] DB_SHOW_RW QSTAT[4:0]		GPO15	4 mA	Real Time Trace

NOTE:

Defaults are 4 mA Drive, but the Neptune LTE internal MCU Boot code reconfigures these pins as 8 mA Drive by writing to the General Purpose Outputs of the EIM module.

NOTE:

The $\overline{\text{LBA}}$ pin is controlled by Group 7 for 4 mA or 8 mA drive strength. However, it is controlled by Group 1 for 16 mA drive strength. Therefore it's Drive Strength can be a bit confusing. Table 2-32 attempts to clarify $\overline{\text{LBA}}$ drive strength settings.

Table 2-32. $\overline{\text{LBA}}$ Drive Strength Settings

Group 1	Group 7	$\overline{\text{LBA}}$ Drive Strength
4 mA or 8 mA	4 mA or 8 mA	4 mA or 8 mA
4 mA or 8 mA	16 mA	4 mA or 8 mA
16 mA	4 mA or 8 mA	16 mA
16 mA	16 mA	16 mA

2.4 Power Supply Description

Neptune LTE power supply configuration is a met by a combination of externally applied voltages and other regulated voltages generated internally. Table 2-33 describe the source and purpose of each power source.

Table 2-33. Power Supply Description

Pin Name	Description
VDD_CP VSS_CP	Supply for charge pump.
REG_BYP_PRE VSS_PRE	Output of prescaler regulator for external bypass capacitor. Supply for prescaler circuits. Input of this regulator is VDD_TXRX.
VDD_TXRX VSS_TXRX	Input to TXRX regulator.
REG_BYP_TXRX	Output of TXRX regulator for external bypass capacitor to VSS_TXRX. Supply for analog part of TX and RX circuits.
VAG_RX	External decoupling for Analog Reference for analog part of RX circuit. Place external resistor between this pin and VSS_TXRX.
VDD_CODEC VSS_CODEC	Input to CODEC regulator.
REG_BYP_CODEC	Output of CODEC regulator for external bypass capacitor to VSS_CODEC. Supply for analog part of CODEC circuit.
VAG_CODEC	Analog Reference for analog part of CODEC circuit. Place an external bypass capacitor to VSS_CODEC on this pin.
VAGOUT_CODEC	Buffered output version of VAG _CODEC. Place an external bypass capacitor to VSS_CODEC on this pin.
VDDA VSSA	Supply for CROSC circuit.
AVDD AVSS	Supply for External Memory Interface Pins.
BVDD BVSS	Supply for Seaweed Data Pins.
CVDD CVSS	Supply for LCD/Test Pin.
DVDD DVSS	Supply for Seaweed Control Pins.
EVDD EVSS	Supply for Algae Control and Data Pins.
FVDD FVSS	Supply for SIM Pins - special 2.85 V supply.
JVDD JVSS	Supply for Keypad and Low Frequency Control Pins. Also powers analog portion of General Purpose ADC, ANATEST, and parts of TXRX Regulator and Core Regulator.
LVDD LVSS	Supply for RTC Pins and Circuit - special 1.575 V supply. Also the reference for many analog circuits.
QVDD QVSS	Input to core VDD regulator. All current dissipated by the core VDD circuits source through this supply.
REG_BYP_CORE	Output of core VDD regulator for external bulk bypass capacitor to QVSS. The core VDD regulator is the supply for all digital portions of Neptune LTE except RTC.
VDD VSS	Output of core VDD regulator brought to four pairs around die for external high frequency bypass capacitors. The core VDD regulator is the supply for all digital portions of Neptune LTE except RTC.

2.5 GPIO Port Mux Assignments

Each GPIO port pin has four possible input selections and eight possible output selections.

2.5.1 Port A Mux Assignments

Table 2-34. Port A Mux Assignments

Port A	DDR	Alt In	Alt Out	IN 00	IN 01	IN 10	IN 11	OUT 000	OUT 001	OUT 010	OUT 011	OUT 100	OUT 101	OUT 110	OUT 111
Port A_0	1	00	010		TEST/DI0			PA_CTL_MCU[0]	PA_CTL_DSP[0]	CCM/CKO	DSP/BDIO[0]	SIM/RCV_CLK	TEST/SCAN_OUT[17]	GPADC/A2D_OUT[0] (BITMAP_0)	
Port A_1	1	00	010		TEST/DI1			PA_CTL_MCU[1]	PA_CTL_DSP[1]	A2DIGL/RX_EN_OUT	DSP/BDIO[1]	SIM/XMT_CLK	TEST/SCAN_OUT[18]	GPADC/A2D_OUT[1] (BITMAP_1)	USB/TOKEN[3]
Port A_2	1	00	010		TEST/DI2	OWIRE/OWB_IN		PA_CTL_MCU[2]	PA_CTL_DSP[2]	L1T/TOUT5	DSP/BDIO[2]		TEST/SCAN_OUT[19]	GPADC/A2D_OUT[2] (BITMAP_2)	OWIRE/OWB_OUT
Port A_3	1	00	010	CCM/REF_CLK	TEST/DI3			PA_CTL_MCU[3]	PA_CTL_DSP[3]	A2DIGL/GSM*_DCS	DSP/BDIO[3]	USB/RCV_DATA	TEST/SCAN_OUT[20]	GPADC/A2D_OUT[3] (BITMAP_3)	USB/TOKEN[2]
Port A_4	1	00	010		TEST/DI4			PA_CTL_MCU[4]	PA_CTL_DSP[4]	L1T/TOUT7	DSP/BDIO[4]	USB/EOPR	TEST/SCAN_OUT[27]	GPADC/A2D_OUT[4] (BITMAP_4)	
Port A_5	1	00	010		TEST/DI5	TEST/SCAN_IN[1]		PA_CTL_MCU[5]	PA_CTL_DSP[5]	L1T/TOUT8	DSP/BDIO[5]	DMAC/LCD_DATA[2] (Alt)	DSP/DBG_YDW	GPADC/A2D_OUT[5] (BITMAP_5)	L1T/TOUT0 (Alt)
Port A_6	1	00	010		TEST/DI6	TEST/SCAN_IN[2]		PA_CTL_MCU[6]	PA_CTL_DSP[6]	L1T/TOUT9	DSP/BDIO[6]	DMAC/LCD_DATA[5] (Alt)	DSP/DBG_XDW	GPADC/A2D_OUT[6] (BITMAP_6)	L1T/TOUT1 (Alt)
Port A_7	1	00	010		TEST/DI7			PA_CTL_MCU[7]	PA_CTL_DSP[7]	L1T/TOUT10	DSP/BDIO[7]	USB/TOKEN[1]	TEST/SCAN_OUT[22]	GPADC/A2D_OUT[7] (BITMAP_7)	
Port A_8	1	00	010		TEST/DI8			PA_CTL_MCU[8]	PA_CTL_DSP[8]	L1T/TOUT11	DSP/BDIO[8]	USB/TOKEN[0]	TEST/SCAN_OUT[23]	GPADC/A2D_OUT[8] (BITMAP_8)	
Port A_9	0	00	000	INT/INT0	TEST/DI9	EGPT/IC1		PA_CTL_MCU[9]	PA_CTL_DSP[9]	RFI/RXSDG_DET_FLAG	DSP/BDIO[9]	UART1/CTS1 (Alt2)	TEST/SCAN_OUT[24]	GPADC/A2D_OUT[9] (BITMAP_9)	
Port A_10	0	00	000	INT/INT1	TEST/DI10		TEST/CODEC0_IN	PA_CTL_MCU[10]	PA_CTL_DSP[10]		DSP/BDIO[10]	USB/EO_SYNC	TEST/SCAN_OUT[25]	TEST/CODEC0_OUT (BITMAP_10)	
Port A_11	1	00	100	INT/INT2	TEST/DI11	TEST/SCAN_IN[3]	TEST/CODEC1_IN	PA_CTL_MCU[11]	PA_CTL_DSP[11]	UART1/DCD1	DSP/BDIO[11]	USB/USB_SUSPEND		TEST/CODEC1_OUT (BITMAP_11)	
Port A_12	0	00	000	INT/INT3	TEST/DI12	MQSPI/MISOB	INT/INT5 (Alt)	PA_CTL_MCU[12]	PA_CTL_DSP[12]	L1T/TOUT6	DSP/BDIO[12]	USB/RCV	TEST/SCAN_OUT[26]		
Port A_13	0	00	000	INT/INT4	TEST/DI13	EGPT/IC2	TEST/CODEC3_IN	PA_CTL_MCU[13]	PA_CTL_DSP[13]	MQSPI/HPF1_WRITE	DSP/BDIO[13]	DSP/PGAD4	TEST/SCAN_OUT[21]	TEST/CODEC3_OUT (BITMAP_13)	DSM/CLK-SEL
Port A_14	1	00	010		TEST/DI14	TEST/SCAN_IN[4]	TEST/CODEC4_IN	PA_CTL_MCU[14]	PA_CTL_DSP[14]	WDOG/WDOG*	DSP/BDIO[14]	CCM/CKOH		TEST/CODEC4_OUT (BITMAP_14)	L1T/FLOAD
Port A_15	0	00	000	SAP/SC0A	TEST/DII5	A2DIGL/SCLK_OUT1	TEST/SCAN_IN[5]	PA_CTL_MCU[15]	PA_CTL_DSP[15]		DSP/BDIO[15]	MQSPI/SPI_CS4	DMAC/LCD_DATA[3] (Alt)	RXCR-POC/SCLK_OUT	DSP/DBG_PAW

2.5.2 Port B Mux Assignments

Table 2-35. Port B Mux Assignments

Port B	DDR	Alt In	Alt Out	IN 00	IN 01	IN 10	IN 11	OUT 000	OUT 001	OUT 010	OUT 011	OUT 100	OUT 101	OUT 110	OUT 111
Port B_0	0	00	000	SAP/SC1A	SAP/SC2A (Alt)	A2DIGL/SDFS1	CCM/GPIO_MCU_CLK	PB_CTL_MCU[0]	PB_CTL_DSP[0]	UART1/DCD1	SAP/SC2A (Alt)	GPIO/PINTOUT[4]	L1T/MQBC[14]		DMAC/LCD_DATA[4] (Alt)
Port B_1	0	00	000	SAP/SCKA	TEST/MAS-TER	A2DIGL/TX_CLK1		PB_CTL_MCU[1]	PB_CTL_DSP[1]	SAP/SCKA	MQSPI/ARAM[8]	GPIO/PINTOUT[3]	L1T/MQBC[15]		
Port B_2	0	00	000	SAP/STDA			TEST/TEST_CLK1	PB_CTL_MCU[2]	PB_CTL_DSP[2]	SAP/STDA	MQSPI/ARAM[7]	GPIO/PINTOUT[2]	LEM/ENC_CLK	DSP/PGAD5	BBP/STDB
Port B_3	0	00	000	SAP/SC2A	TEST/RW		TEST/TEST_CLK3	PB_CTL_MCU[3]	PB_CTL_DSP[3]	SAP/SC2A	MQSPI/ARAM[6]	GPIO/PINTOUT[1]	LEM/RSTSRC		L1T/MQBC[12]
Port B_4	0	00	000	SAP/SRDA	TEST/MA0		TEST/SCAN_IN[6]	PB_CTL_MCU[4]	PB_CTL_DSP[4]	CCM/EXT_BOOT_B	MQSPI/ARAM[5]	GPIO/PINTOUT[0]	LEM/ENC_RESET_N	RxCPROC/SRDX	DSP/PGAD3
Port B_5	1	00	010			TEST/SCAN_IN[28]	TEST/A2DI_SCKB_IN	PB_CTL_MCU[5]	PB_CTL_DSP[5]	UART2/TXD2		DSP/PGAD1			
Port B_6	0	00	000	UART2/RXD2		TEST/SCAN_IN[29]		PB_CTL_MCU[6]	PB_CTL_DSP[6]			DSP/PGAD2		(BIST/BITMAP_15)	
Port B_7	1	00	010		TEST/MA1	TEST/SCAN_IN[7]	TEST/CODEC5_IN	PB_CTL_MCU[7]	PB_CTL_DSP[7]	SIM/SIM_RST	MQSPI/ARAM[4]	DSM/STATE[3]	LEM/INTREQ	TEST/CODEC5_OUT (BITMAP_16)	L1T/MQBC[11]
Port B_8	0	00	000	SIM/SIM_PD	TEST/MA2	TEST/SCAN_IN[8]	TEST/CODEC6_IN	PB_CTL_MCU[8]	PB_CTL_DSP[8]		MQSPI/ARAM[3]	USB/MSM_STATE[4]	LEM/STATE[0]	TEST/CODEC6_OUT (BITMAP_17)	L1T/MQBC[10]
Port B_9	1	00	010		TEST/MA3	TEST/SCAN_IN[9]	TEST/CODEC7_IN	PB_CTL_MCU[9]	PB_CTL_DSP[9]	SIM/SIM_CLK	MQSPI/ARAM[2]	USB/MSM_STATE[3]	LEM/STATE[1]	TEST/CODEC7_OUT (BITMAP_18)	L1T/MQBC[9]
Port B_10	1	00	010	SIM/SIM_DIO	TEST/MA4	TEST/SCAN_IN[10]	TEST/CODEC8_IN	PB_CTL_MCU[10]	PB_CTL_DSP[10]	SIM/SIM_DIO	MQSPI/ARAM[1]	USB/MSM_STATE[2]	LEM/STATE[2]	TEST/CODEC8_OUT (BITMAP_19)	L1T/MQBC[8]
Port B_11	1	00	010		TEST/MA5	TEST/SCAN_IN[11]	TEST/CODEC9_IN	PB_CTL_MCU[11]	PB_CTL_DSP[11]	SIM/SIM_VCCEN	MQSPI/ARAM[0]	USB/MSM_STATE[1]	LEM/STATE[3]	TEST/CODEC9_OUT (BITMAP_20)	L1T/MQBC[7]
Port B_12	1	00	010		TEST/MA6	TEST/SCAN_IN[12]	A2DIGL/STDB1	PB_CTL_MCU[12]	PB_CTL_DSP[12]	DMAC/SDATA	DSP/BDIO[16]	USB/MSM_STATE[0]	L1T/MQBC[6]		DSM/STATE[2]
Port B_13	1	00	010	TRIPLE_TIM/TIN2	TEST/MA7	TEST/SCAN_IN[13]	TEST/CODEC10_IN	PB_CTL_MCU[13]	PB_CTL_DSP[13]	DMAC/LCD_CLK	MQSPI/LD_SR1	USB/DCLK	TRIPLE_TIM/TRIP_TOUT2	TEST/CODEC10_OUT (BITMAP_21)	DSP_DBG/DSP_AT
Port B_14	1	00	010	TRIPLE_TIM/TIN1	TEST/MA8	TEST/SCAN_IN[14]	TEST/CODEC11_IN	PB_CTL_MCU[14]	PB_CTL_DSP[14]	DMAC/LCD_CS	MQSPI/LD_SR2	USB/RXDATA_POS	TRIPLE_TIM/TRIP_TOUT1	TEST/CODEC11_OUT (BITMAP_22)	DSM/STATE[1]
Port B_15	1	00	010	TRIPLE_TIM/TIN0	TEST/MA9	TEST/SCAN_IN[15]	TEST/CODEC12_IN	PB_CTL_MCU[15]	PB_CTL_DSP[15]	DMAC/LCD_RS	DSP/PGAD13	DSM/STATE[0]	TRIPLE_TIM/TRIP_TOUT0	TEST/CODEC12_OUT (BITMAP_23)	L1T/MQBC[13]

2.5.3 Port C Mux Assignments

Table 2-36. Port C Mux Assignments

Port C	DDR	Alt In	Alt Out	IN 00	IN 01	IN 10	IN 11	OUT 000	OUT 001	OUT 010	OUT 011	OUT 100	OUT 101	OUT 110	OUT 111
Port C_0	1	00	010			TEST/TEST_CLK0	TEST/CODEC2_IN	PC_CTL_MCU[0]	PC_CTL_DSP[0]	UART1/CTS1 (Alt)				TEST/CODEC2_OUT (BITMAP_12)	
Port C_1	1	00	010		TEST/MA10	TEST/SCAN_IN[16]		PC_CTL_MCU[1]	PC_CTL_DSP[1]	JTAG/RTCK	DSP/BDIO[17]		MQSPI/HPF2_WRITE	GPADC/END_ADC	L1T/ALOAD
Port C_2	1	00	010	INT/INT7	TEST/MA11	TEST/SCAN_IN[17]	TEST/CODEC13_IN	PC_CTL_MCU[2]	PC_CTL_DSP[2]	ADC/ADC_SYNC	DSP/BDIO[18]	CCM/CKOH (Alt)	MQSPI/LPF1_WRITE	TEST/CODEC13_OUT (MISR_0)	L1T/BLOAD
Port C_3	0	00	000	KPP/COLUMN0	TEST/MA12	TEST/SCAN_IN[18]	TEST/CODEC14_IN	PC_CTL_MCU[3]	PC_CTL_DSP[3]	KPP/COLUMN0	DSP/BDIO[19]	DSP/PGAD8	MQSPI/LPF2_WRITE	TEST/CODEC14_OUT (MISR_1)	
Port C_4	0	00	000	KPP/COLUMN1	TEST/MA13	TEST/SCAN_IN[19]	TEST/CODEC15_IN	PC_CTL_MCU[4]	PC_CTL_DSP[4]	KPP/COLUMN1	DSP/BDIO[20]	EGPT/OC2	MQSPI/TRIG_QUEUE[4]	TEST/CODEC15_OUT (MISR_2)	DSP/PGAD9
Port C_5	0	00	000	KPP/ROW0	TEST/MA14	TEST/SCAN_IN[20]		PC_CTL_MCU[5]	PC_CTL_DSP[5]	DSP/PGAD14	DSP/BDIO[21]	TEST/pac_dig_gpio_adc[4]	MQSPI/TRIG_QUEUE[3]	BIST/ipt_membist_fail	TX/TX_CLK
Port C_6	0	00	000	KPP/ROW1	TEST/MA15	TEST/SCAN_IN[21]		PC_CTL_MCU[6]	PC_CTL_DSP[6]	A2DIGL/CONVERT	DSP/BDIO[22]	TEST/pac_dig_gpio_adc[3]	MQSPI/TRIG_QUEUE[2]	BIST/ipt_test_membist_data_out[0]	DSP_DBG/DSP_AT
Port C_7	0	00	000	KPP/ROW4	TEST/EXEC	TEST/SCAN_ENABLE		PC_CTL_MCU[7]	DSP/PGAD6	A2DIGL/PWD	DSP/BDIO[23]	TEST/pac_dig_gpio_adc[2]	MQSPI/TRIG_QUEUE[1]	BIST/ipt_test_membist_data_out[1]	MQSPI/SPI_CS7 (Alt)
Port C_8	0	00	000	KPP/ROW5	TEST/MA21	CCM/CCM_USB_CLK	TEST/SCAN_IN[22]	PC_CTL_MCU[8]	PC_CTL_DSP[8]	A2DIGL/EN_GPADC	DSP/DSP_RD_B	TEST/pac_dig_gpio_adc[1]	DSP/PGAD13	BIST/ipt_test_membist_data_out[2]	MQSPI/SPI_CS8 (Alt2)
Port C_9	0	00	000	INT/INT6				PC_CTL_MCU[9]	PC_CTL_DSP[9]						
Port C_10	0	00	000	KPP/ROW6	TEST/MA20	L1T/FRAME_TICK	TEST/SCAN_IN[23]	PC_CTL_MCU[10]	PC_CTL_DSP[10]	EGPT/OC1	DSP/DSP_WR_B	TEST/pac_dig_gpio_adc[0]	DSP/PGAD7	BIST/ipt_test_membist_data_out[3]	MQSPI/SPI_CS9 (Alt)
Port C_11	0	00	000	KPP/ROW7	TEST/MA16	TEST/SCAN_IN[24]		PC_CTL_MCU[11]	PC_CTL_DSP[11]		DSP/PGAD15	TEST/pac_dig_gpio_sync		BIST/ipt_test_membist_data_out[4]	
Port C_12	0	00	000	GPIO/PC12	TEST/MA17	TEST/SCAN_IN[25]		PC_CTL_MCU[12]	PC_CTL_DSP[12]	EGPT/OC3	DSP/PGAD12	DMAC/LCD_DATA[2] (Alt2)	GEM/DEBUG[10]	BIST/ipt_test_membist_data_out[5]	L1T/FATAL
Port C_13	0	00	000	JTAG/SJC_MOD	TEST/MA18	TEST/SCAN_IN[26]		PC_CTL_MCU[13]	PC_CTL_DSP[13]			DMAC/LCD_DATA[4] (Alt2)	GEM/DEBUG[11]	BIST/ipt_test_membist_data_out[6]	L1T/OUT_OF_SYNC
Port C_14	0	00	000	GPIO/PC14	TEST/MA19	TEST/SCAN_IN[27]		PC_CTL_MCU[14]	PC_CTL_DSP[14]	UART1/RI1	DSP/PGAD14	DMAC/LCD_DATA[5] (Alt2)	GEM/DEBUG[12]	BIST/ipt_test_membist_data_out[7]	DSP/DSP_DBG_PAGE2
Port C_15	1	00	010					PC_CTL_MCU[15]	PC_CTL_DSP[15]	MQSPI/SPI_CS6					

2.5.4 Port D Mux Assignments

Table 2-37. Port D Mux Assignments

Port D	DDR	Alt In	Alt Out	IN 00	IN 01	IN 10	IN 11	OUT 000	OUT 001	OUT 010	OUT 011	OUT 100	OUT 101	OUT 110	OUT 111
Port D_0	1	00	010		TEST/DI16		TEST/FVMAIN_IN	PD_CTL_MCU[0]	PD_CTL_DSP[0]	A2DIGL/RF_CLK	TEST/DO[0]		GEM/DEBUG[9]	UART1/TX_FIFO_EMPTY	MQSPI/QSCKB
Port D_1	1	00	010		TEST/DI17		BIST/IPT_TEST_MEMBIST_INVOKE	PD_CTL_MCU[1]	PD_CTL_DSP[1]	A2DIGL/RF_DATA	TEST/DO[1]	L1T/TOUT2	GEM/DEBUG[8]	UART1/TX_FIFO_FULL	MQSPI/MOSIB
Port D_2	1	00	010		TEST/DI18		BIST/IPT_TEST_MEMBIST_RESET	PD_CTL_MCU[2]	PD_CTL_DSP[2]	A2DIGL/RF_CS	TEST/DO[2]	L1T/TOUT3	GEM/DEBUG[7]	UART1/RX_FIFO_FULL	MQSPI/SPI_CS6 (Alt)
Port D_3	1	00	010		TEST/DI19		A2DIGL/SPI_CLK1	PD_CTL_MCU[3]	PD_CTL_DSP[3]	MQSPI/QSCKA	TEST/DO[3]	L1T/TOUT12	GEM/DEBUG[6]	UART1/RX_FIFO_EMPTY (MISR_3)	
Port D_4	0	00	000	MQSPI/MISOA	TEST/DI20		A2DIGL/SPI_DATA1	PD_CTL_MCU[4]	PD_CTL_DSP[4]		TEST/DO[4]	L1T/TOUT13	GEM/DEBUG[5]	UART1/NRZ_CONV_RX (MISR_4)	
Port D_5	1	00	010	A2DIGL/CE_A1	TEST/DI21		A2DIGL/t_convert	PD_CTL_MCU[5]	PD_CTL_DSP[5]	MQSPI/MOSIA	TEST/DO[5]	L1T/TOUT14	GEM/DEBUG[4]	MQSPI/SPI_CS5 (Alt)	UART1/NRZ_DEC_TX
Port D_6	1	00	010		TEST/DI22			PD_CTL_MCU[6]	PD_CTL_DSP[6]	MQSPI/SPI_CS0	TEST/DO[6]	L1T/TOUT15	GEM/DEBUG[1]	TEST/CODEC_STROBE (MISR_5)	DMAC/LCD_DATA[0] (Alt)
Port D_7	1	00	010		TEST/DI23		A2DIGL/CE1	PD_CTL_MCU[7]	PD_CTL_DSP[7]	MQSPI/SPI_CS1	TEST/DO[7]		DSP/DSP_DBG_PAGE1	GEM/DEBUG[0] (MISR_7)	DMAC/LCD_DATA[1] (Alt)
Port D_8	1	00	010		TEST/DI24		A2DIGL/RXACQ1	PD_CTL_MCU[8]	PD_CTL_DSP[8]	MQSPI/SPI_CS2	TEST/DO[8]	L1T/TOUT0	DSP/DSP_DBG_PAGE0	GEM/DEBUG[2] (MISR_8)	UART1/BRM_CLOCK
Port D_9	1	00	010		TEST/DI25		A2DIGL/DMCS1	PD_CTL_MCU[9]	PD_CTL_DSP[9]	MQSPI/SPI_CS3	TEST/DO[9]	L1T/TOUT1	GEM/DEBUG[3]	(MISR_9)	UART1/SOFTWARE_RESET
Port D_10	1	00	010		TEST/DI26		TEST/GPIO_GCLKW	PD_CTL_MCU[10]	PD_CTL_DSP[10]	USB/USB_TXENB	TEST/DO[10]	L1T/TOUT4	UART1/DSR1		
Port D_11	0	00	000	USB/USB_VPIN	TEST/DI27	UART1/DTR1		PD_CTL_MCU[11]	PD_CTL_DSP[11]	EGPT/PWM_N	TEST/DO[11]	L1T/RCLK	GEM/DEBUG[13]		MQSPI/SPI_CS7
Port D_12	0	00	000	USB/USB_VMIN	TEST/DI28	UART1/RXD1		PD_CTL_MCU[12]	PD_CTL_DSP[12]	EGPT/PWM	TEST/DO[12]	L1T/SYNC_UPDATE		BIST/ipt_test_membist_done	MQSPI/SPI_CS8
Port D_13	0	00	000	USB/USB_XRXD	TEST/DI29	UART1/RTS1		PD_CTL_MCU[13]	PD_CTL_DSP[13]		TEST/DO[13]	L1T/CCLK		RXCPROC/RXCPROC_SDFS	MQSPI/SPI_CS9
Port D_14	1	00	010		TEST/DI30		TEST/CODEC_DIR_IN	PD_CTL_MCU[14]	PD_CTL_DSP[14]	USB/USB_VPOUT	TEST/DO[14]		UART1/TXD1	(MISR_6)	MQSPI/SPI_CS10
Port D_15	1	00	010		TEST/DI31			PD_CTL_MCU[15]	PD_CTL_DSP[15]	USB/USB_VMOUT	TEST/DO[15]	DSM/DIS_L1T	UART1/CTS1	TX/TX_CLK	

2.5.5 Port E Mux Assignments

Table 2-38. Port E Mux Assignments

Port E	DDR	Alt In	Alt Out	IN 00	IN 01	IN 10	IN 11	OUT 000	OUT 001	OUT 010	OUT 011	OUT 100	OUT 101	OUT 110	OUT 111
Port E_0	0	00	000	UART1/URTS1				PE_CTL_MCU[0]	PE_CTL_DSP[0]	AEIM/DBG_EDEAD	DSP/PGAD0		AEIM/DBG_CSTATE[0]		
Port E_1	1	00	010					PE_CTL_MCU[1]	PE_CTL_DSP[1]	L1T/TOUT12 (Alt)		GPADC/SAMPLE	AEIM/DBG_CSTATE[1]		
Port E_2	1	00	010				A2DIGL/SRDX1	PE_CTL_MCU[2]	PE_CTL_DSP[2]	MQSPI/SPI_CS8 (Alt)		GPADC/CMP	AEIM/DBG_CSTATE[2]		
Port E_3	0	00	000	AEIM/EBW*	INT/INT4 (Alt)		TEST/SIO_DIN	PE_CTL_MCU[3]	PE_CTL_DSP[3]	DMAC/LCD_DATA[4] (Alt3)				BIST/IPT_DSP_BIST_DONE (IPT_DSP_BIST_DONE)	RXCPROC/RXCPROC_SDFS (Alt)
Port E_4	1	00	010					PE_CTL_MCU[4]	PE_CTL_DSP[4]	DMAC/LCD_DATA[0]		GPIO/PINTOUT[0] (Alt)	AEIM/DBG_CSTATE[3]		
Port E_5	1	00	010					PE_CTL_MCU[5]	PE_CTL_DSP[5]	DMAC/LCD_DATA[1]		GPIO/PINTOUT[1] (Alt)	AEIM/DBG_BCLK_CSTATE[0]		
Port E_6	1	00	010					PE_CTL_MCU[6]	PE_CTL_DSP[6]	DMAC/LCD_DATA[2]		GPIO/PINTOUT[2] (Alt)	AEIM/DBG_BCLK_CSTATE[1]		
Port E_7	1	00	010					PE_CTL_MCU[7]	PE_CTL_DSP[7]	DMAC/LCD_DATA[3]		GPIO/PINTOUT[3] (Alt)	AEIM/DBG_BCLK_CSTATE[2]		
Port E_8	1	00	010					PE_CTL_MCU[8]	PE_CTL_DSP[8]	DMAC/LCD_DATA[4]		GPIO/PINTOUT[4] (Alt)	AEIM/DBG_INS_WAIT		
Port E_9	1	00	010					PE_CTL_MCU[9]	PE_CTL_DSP[9]	DMAC/LCD_DATA[5]			AEIM/DBG_PRE_BURST		
Port E_10	0	00	000	GPIO/PE10	SAP/SCKA	TEST/SCAN_IN[30]		PE_CTL_MCU[10]	PE_CTL_DSP[10]	UART1/TXD1	DSP/PGAD10	UART1/RI1 (Alt)	AEIM/DBG_LATCH_DATA	BIST/IPT_DSP_BIST_FAIL (IPT_DSP_BIST_FAIL)	SAP/SCKA
Port E_11	0	00	000	GPIO/PE11	UART1/RXD1	TEST/SCAN_IN[31]	TEST/SIO_RD	PE_CTL_MCU[11]	PE_CTL_DSP[11]		DSP/PGAD11	UART1/DSR1 (Alt)	AEIM/DBG_LAT_EXTMEM	BIST/IPT_DSP_CFG (IPT_DSP_CFG)	SAP/STDA (Alt)
Port E_12	0	00	000	GPIO/PE12	UART1/RTS1 (Alt)	SAP/SRDA (Alt)	UART1/DTR1 (Alt)	PE_CTL_MCU[12]	PE_CTL_DSP[12]				TEST/SCAN_OUT[28]		
Port E_13	1	00	010				TEST/SIO_EN_SHIFT	PE_CTL_MCU[13]	PE_CTL_DSP[13]	UART2/CTS2	DSP/PGAD12 (Alt)	L1T/TOUT15 (Alt)	TEST/SCAN_OUT[29]		CCM/AP_WAKEUP_B
Port E_14	0	00	000	MQSPI/MISOB (Alt)	TEST/codec_bist_clk			PE_CTL_MCU[14]	PE_CTL_DSP[14]	DMAC/LCD_DATA[3] (Alt2)		TEST/SIO_DOUT	TEST/SCAN_OUT[30]		
Port E_15	0	00	000	UART2/RTS2			TEST/SIO_WR	PE_CTL_MCU[15]	PE_CTL_DSP[15]		DSP/PGAD0 (Alt)	TEST/TCM_GPIO_OUT[16]	TEST/SCAN_OUT[31]		CCM/AMARB_BG_CCM

Chapter 3
Clocks, Low Power Control and Reset (CLKRST)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	4/22/02	L.Hansford	PIG/JIG Modules changed to IP Modules Changed DCLKG control from CCM to DSP Adding PCTL1 bits for control of DSP low power divide and PCE, and MLM bits Changed DMAC clock to only source from PAT_REF_WAIT
0.2	5/1/02	L. Hansford	Added new modules and thier rrospective clock domains to Tables 3-1 & 3-3
0.3	5/24/02	L. Hansford	Add LRF flag to the DSP PCTL Remove all references to the MDI_SHIFT_CLK add DRA to Table 3-1 add reset out add uart2 to the clock domain Chnage the MSu to MCU_GRPA clock
0.4	07/19/02	Lamar Hansford, Shannon Osgood	Updated per DDTS# DSPhl14837. Removed reference to MMUX in table 3-1.
0.5	07/22/02	Lamar Hansford, Shannon Osgood	Updated per DDTS# DSPhl15007.
0.6	01/06/03	Lamar Hansford, Shannon Osgood	Updated per DDTS# DSPhl15736.
0.7	05/07/03		Updated for LTE specification release.

3.1 Clock Sources

Since the Neptune IC integrates a RISC microprocessor (MCU) and a general purpose digital signal processor (DSP) with many various peripherals connected to each of them on a single chip, the device integrates a complex clock scheme architecture to support the different needs of the system. The following is a summary of the basic routing of the Neptune clock sources. Refer to Figure 3-1 through Figure 3-2 for details.

The Neptune IC has two main clock sources: a 26 MHz reference provided by an internal oscillator and an external 26 MHz crystal and a 32.768 kHz square wave input (CKIL).

The digital subsystem will derive all its functional clocks from the Reference DPLL. This Reference DPLL will be frequency locked to the Wireless Base Station. From this, a corrected 13 MHz (PAT_REF) can be derived. The DSP and MCU clocks via integer dividers are also directly derived from the Reference DPLL.

The DSP clocks, the MCU clocks and other internal clock signals can be multiplexed on both the CKO and the CKOH pins which go through GPIO and are controlled by MCU software. CKO defaults to output the MCU GRPA clock upon reset. These outputs can be independently disabled under software control, in which case they remain low.

MCU peripherals have access to either the 32.768 kHz (CKIL), MCU clocks GRPA, GRPB, and GRPC, or one of three PAT_REF clocks. The use of different clocks allows some of the MCU peripherals to maintain a constant operational frequency even if the PAT_REF input signal is shut down or the MCU_CLK selector is changed. Changing the peripheral frequency or using different frequencies by different parts of the same peripheral implies severe constraints on clock frequencies and synchronization issues which should be addressed carefully (refer to each MCU peripheral for further details).

DSP peripherals interface with the DSP core, which is designed for a very fast, tightly coupled protocol without any possibility to halt the DSP core (i.e. no wait states permitted on core accesses), so the DSP peripherals receive the DSP core internally generated raw clock, which is synchronized with the DSP core clock and at the same frequency. Serial clocks generated internally by the DSP peripherals are generated from this clock or when possible from the PAT_REF clock under some restrictions (refer to each DSP peripheral for further details).

For test and debug purposes, there are many places in the system clocking chain, where an internally generated clock can be bypassed and replaced by an externally supplied source routed through the GPIO module. The GPIO insertion points are clearly defined in the subsequent system clocking diagrams.

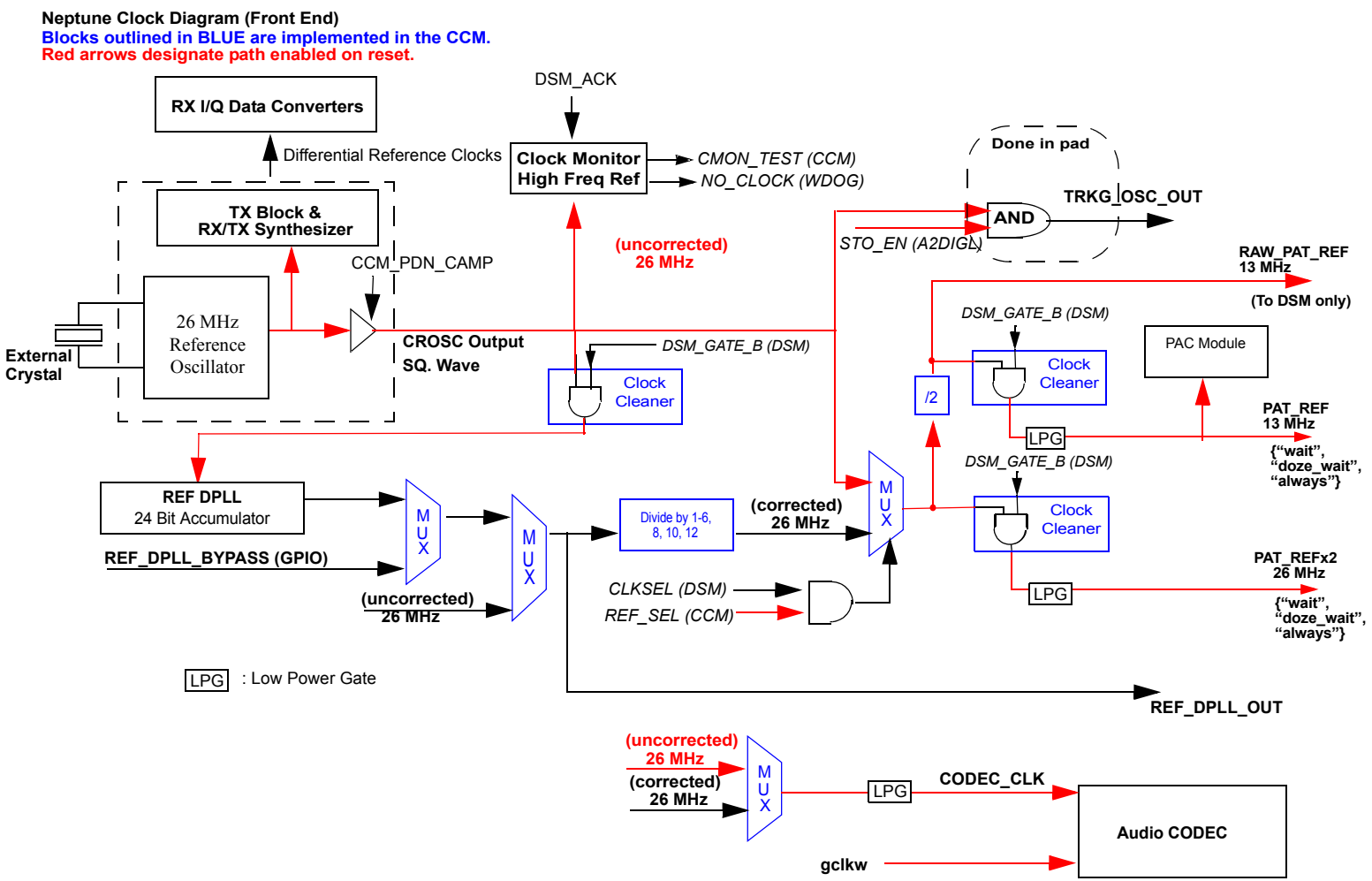


Figure 3-1. Neptune Clock Distribution (Page 1 of 2)

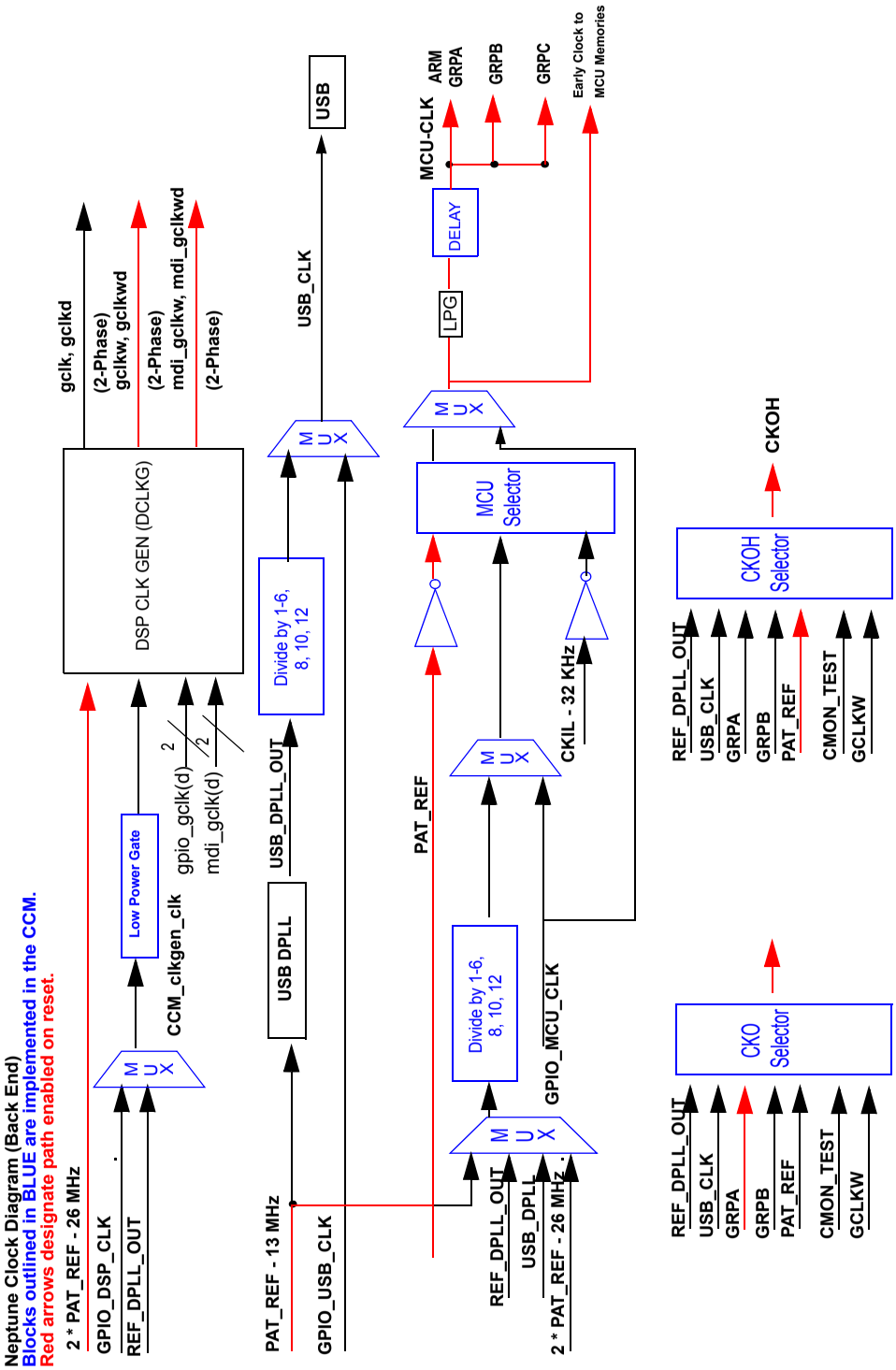


Figure 3-2. Neptune Clock Distribution (Page 2 of 2)

3.2 MCU Clock Generation

There is a single clock entering the ARM platform. This clock drives both the ARM module and the memories and peripherals inside the ARM platform. The dividers which provide lower frequency clocks to the DSP, MCU, and USB are not required to be 50% duty cycle for non-power of 2 divide ratios. They are required to have a minimum pulse width of the next lowest power of 2 divide.

The MCU clock is generated from one of the three clock sources: PAT_REF, the 32.768KHz input (CKIL), or the output of a mux of an external source (through GPIO) and a divided clock. The divided clock is the output of a mux of PAT_REF, the Reference DPLL, the USB DPLL, and 2*PAT_REF; which is then divided by a programmable divider and a fixed divide by 2. The three possible MCU clocks are selected by a programmable synchronous selector, which provides a smooth transition between the clocks. The selector output has three different branches, GRPA, GRPB, and GRPC. The functional characteristics of these three clocks are described in the CCM module chapter. There is also an early clock generated for the MCU memories. Each block generates its own clocks by a standard methodology, to achieve minimum skew between the clocks on different system parts which have common interfaces.

NOTE:

All clocks (except gclkw and gclkwd) are sourced by the CCM unless otherwise noted. See the CCM module chapter for functional details of these clocks.

Table 3-1. MCU Peripheral Interfaces and Clocks

Peripheral	Clock(s)	Interface
EGPT	grpc, ckil	IP
MQSPI	grpa, pat_ref_doze_wait	IP
SIM	grpa, pat_ref_always,ckil	IP
UART1	grpb, pat_ref_doze_wait	IP
UART2	grpb, pat_ref_doze_wait	IP
AWPT	grpa	AHB
DMAC	grpc	IP
DRA	grpc	IP
AMARB	grpc	IP - AHB
RTR	grpb, ckil	IP
IIM	no clock	IP
WDOG	grpb, ckil	IP
CCM	grpa, ckil (direct from pad), gclkw	IP - AHB
A2DIGL	grpa,pat_ref_always,ckil	IP
GPADC	pat_ref_doze_always,ckil	none
PAC_DIG	pat_refx_doze_wait	none
TUNEC	pat_ref_doze_wait	none

Table 3-1. MCU Peripheral Interfaces and Clocks

Peripheral	Clock(s)	Interface
AIPI	grpa	AHB
CARB	grpa	AHB
TCM	grpc(ip_platform)	IP - AHB
GEM	grpc	IP -AHB
DSM	grpa, raw_pat_ref(always), ckil	IP
L1 Timer	grpa, pat_ref_doze_wait, gclkw	IP - SPMBIF
USB	grpc,usb_clk, glckw	IP - SPMBIF
USB DPLL	pat_ref_always	none
KPP	grpa, ckil	IP
GPIO	grpb, gclkw_b	IP - SPMBIF
AEIM	grpa	AHB
AITC	grpa	AHB
INT	grpa	IP
MDI	grpb, gclkw	IP - SPMBIF
RTC	grpa,ckil (direct from pad)	IP
MCTL	grpa	IP - AHB
SJC	grpc, gclkw	none
MRAM	early_mcu_clk	AHB
MROM	early_mcu_clk	AHB
OWIRE	grpb, pat_ref_doze_wait	IP
HAC	grpa	IP
SCC	grpc	AHB
MSU	grpb, pat_ref_doze_wait	AHB
MDPI	grpa, gclkw	IP
STC	no clock	IP

3.3 DSP CLKGEN (DCLKG) - Block Diagram and Control

The CLOCK GENERATOR block diagram is shown in Figure 3-3. The components of the CLOCK GENERATOR are described in the following sections. **The DSP CLKGEN module is controlled via registers in the DSP register memory map.** In the diagram below, gclkw and its delayed version, gclkwd, is active when the DSP is in RUN or WAIT modes. The gclk and gclkd clocks are only active in DSP RUN mode. The mdi_gclkw and mdi_gclkwd clocks supply the part of DSP Data RAM that is shared with the MCU. The DSP control of the Low Power Divide will be controled through register mapped to the PCTL1 in the DSP X-memory.

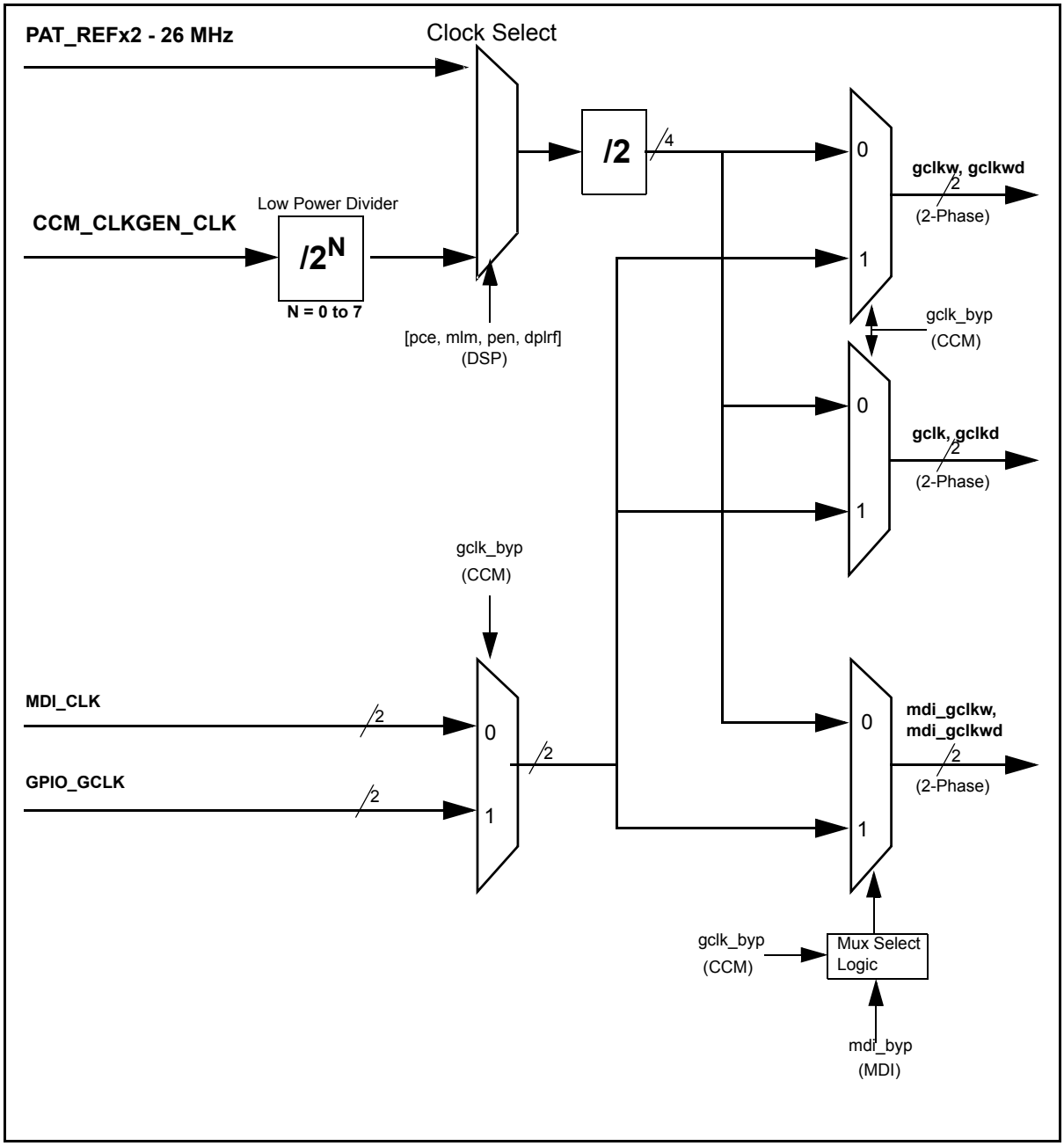


Figure 3-3. CLKGEN Block Diagram

PCTL1 DSP DPLL Control Register 1 **Addr X:$FFFC**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	DPLRF			MLM	PCE					PEN	PSTP			DF[2:0]		
TYPE	rw	r	r	rw	rw	r	r	r	r	rw	rw	r	r	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-2. PCTL1 Description

Name	Description	Settings
DPLRF Bit 15	**DPLL Lock Ready Flag**—This output signal, when set, indicates that the DPLL is in lock. If MF is integer and the DPLL is in the FPL mode, the DPLRF bit is primarily set when a phase error is reduced to zero. After that, the bit can be zeroed in the case that the phase error has increased over 25% of the chip clock period. For a non-integer MF or in the FOL mode, the DPLRF bit is set when the given number of reference clocks arrives after the DPLL starts.	0=unlock 1=lock
Bits 14-13	**Reserved Bits — These bits are read as 0 and should be written as 0 for future compatibility.**	N/A
MLM Bit 12	**Manual Lock Mode Bit —** The MLM bit controls the manner of switching the DSP-CLK from the PAT_REF/2 to the PLL output clock. When this bit is set, the switching will be performed by DSP software through the PCE bit of the PCTL1 register. When this bit is cleared, the DSP-CLK is replaced automatically after the PLL lock condition has been fulfilled. The bit cannot be changed if the PLL is enabled. The MLM bit is cleared by hardware reset.	<table><tr><th>MLM</th><th>PCE</th><th>CLKGEN input clock</th></tr><tr><td>0</td><td>x</td><td>PAT_REF before PLL lock / PLL Output after PLL lock</td></tr><tr><td>0</td><td>x</td><td>PAT_REF before PLL lock / PLL Output after PLL lock</td></tr><tr><td>1</td><td>0</td><td>PAT_REF</td></tr><tr><td>1</td><td>1</td><td>PLL Output</td></tr></table>
PCE Bit 11	**PLL Clock Enable Bit —** The PCE bit controls switching of the DSP-CLK from PAT_REF/2 to the PLL output clock in the manual mode, when the MLM bit is set. The DSP Core can force switching through this bit. When the PCE bit is set, the output clock is produced by the PLL. When the bit is cleared, the PAT_REF feeds the DSP CLKGEN block. If the MLM bit is cleared, the PCE bit is ignored. The PCE bit is cleared by hardware reset or if the PLL is disabled.	
Bits 10-7	**Reserved Bits — Those bits should be written as 0 for future compatibility.**	N/A
PEN Bit 6	**PLL Mux Enable bit**- This bit must be set in order to access the PLL derived clock	0- Path disabled 1- Path Enabled

Table 3-2. PCTL1 Description

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>PSTP
Bit 5</td><td>STOP Processing State Bit — The PSTP bit controls the behavior of the DSP-PLL during the stop processing state. When PSTP is set, the DSP-PLL will remain operating while the DSP56600 Core is in the stop processing state. When PSTP is cleared, the DSP-PLL will be disabled when the DSP56600 Core enters the stop processing state. For minimum power consumption during the stop state at the cost of longer recovery time, PSTP should be cleared. To enable rapid recovery when exiting the stop state, at the cost of higher power consumption, PSTP should be set. PSTP is cleared by hardware reset.</td><td>0 = DSP-PLL will be disabled when the DSP56600 Core enters the stop processing state. (default).
1 = DSP-PLL will remain operating while the DSP56600 Core is in the stop processing state.</td></tr>
<tr><td>Bit 4</td><td>Reserved Bit — This bit is read as 0 and should be written as 0 for future compatibility.</td><td>N/A</td></tr>
<tr><td>Bit 3</td><td>Reserved Bit — This bit should be written as 0 for future compatibility.</td><td>N/A</td></tr>
<tr><td>DF[2:0]
Bits 2-0</td><td>Division Factor Bits— The Division Factor bits DF2-DF0 define the divide factor DF of the low power divider. These bits specify any power of two divide factor in the range from 2^0 to 2^7. They are used also in order to generate the needed DSP-CLK frequency from the DPLL output frequency which must lie between 40-130MHz MHz. The table to the right shows the programming of the DF2-DF0 bits. Changing the value of the DF2-DF0 bits will not cause a loss of lock condition. Whenever possible, changes of the operating frequency of the DSP56600 Core (for example, to enter a low power mode) should be made by changing the value of the DF2-DF0 bits rather than changing the MF bits. Changing DF2-DF0 may lengthen the instruction cycle following the PLL control register update; this is done in order to keep synchronization between PAT_REF and the internal chip clock. These bits are cleared (division by one) by hardware reset.</td><td>
<table>
<tr><th>DF2-DF0</th><th>Division Factor DF</th></tr>
<tr><td>$0</td><td>$2^0$</td></tr>
<tr><td>$1</td><td>$2^1$</td></tr>
<tr><td>$2</td><td>$2^2$</td></tr>
<tr><td>•</td><td>•</td></tr>
<tr><td>$7</td><td>$2^7$</td></tr>
</table>
</td></tr>
</table>

3.3.1 Low Power Divider (LPD)

The Low Power Divider (LPD) divides the output frequency of the REF_DPLL by any power of 2 from 2^0 to 2^7. Since the LPD is not in the closed loop of the REF_DPLL, changes in the divide factor do not cause a loss of lock condition. This fact is particularly useful for utilizing the LPD in low power consumption modes when the chip is not involved in intensive calculations. This can result in significant power savings. When the chip is required to exit the low power mode, it can immediately do so with no time needed for clock recovery or REF_DPLL lock.

3.3.2 Clock Select

The clock select block in the DCLKG module selects via the bit between PAT_REFx2 and the output of the CCM mux CCM_CLKGEN_CLK. **Please note the following:**

- Before CCM_CLKGEN_CLK stops, PCE should be selecting PAT_REFx2 or the S-OnyxU subsystem should be in STOP mode.
- Before the REF_DPLL is restarted (and locked), PCE should be selecting PAT_REFx2 or the S-OnyxU subsystem should be in STOP mode.
- The maximum frequency of CCM_CLKGEN_CLK is 260 MHz.

3.3.3 Divide by 2

The PAT_REFx2 clock and the output of the Low-Power Divider are selected according to the PCE bit in the CCM DSP CLKGEN control register (CGCTL). The selected clock frequency is divided by two and is driven to the DSP core, memories and peripherals.

3.3.4 DSP Operating Frequency

The operating frequency of the DSP clock is governed by the Reference DPLL output frequencyset in the Clock Control Module (CCM) accessible from the MCU memory map, and the DSP divide factors located in the PCTL1 in the DSP memory map. Please note that the PEN bit in the PCTL1 (DCLKG) register MUST be set in order to obtain the PLL derived clock.

3.3.5 DSP Peripherals Clock Sources

Table 3-3. DSP Module Clock Sources

Module	Module Clock Source
DSP Clock Generator (DCLKG)	grpb, pat_refx2_doze_wait, dsp_clkgen_clk (see CCM module chapter for descriptions), gpio_gclkw,gpio_gclkwd
DSP Core (S-OnyxU)	gclkw
DSP Memories (DMEM)	gclkw,gclkwd,mdi_gclkw,mdi_gclkwd
Layer 1 Encryption Module (LEM)	gclkw
Base Band Port (BBP)	gclkw, external
Triple Timer (DTIMER)	gclkw, external
Serial Audio Port (SAP)	gclkw, pat_ref_doze_wait, external
Direct Memory Access (DMA1)	gclkw
Direct Memory Access (DMA2)	gclkw
Midrect Memory Access Arbiter (DMA_ARB)	gclkw
Viterbi Accelerator (VIAC)	gclkw

Table 3-3. DSP Module Clock Sources

Module	Module Clock Source
DSP Debug (PDDM)	gclkw
Special Interrupt Handler (DSIH)	gclkw
DSP OnCE	gclkw, tck
Voice Codec (VOCOD)	gclkw, codec_clk

3.4 Low Power Modes

3.4.1 Introduction

The Neptune chip supports various power modes and features, including:

- MCU STOP mode (independent of DSP).
- MCU DOZE mode (independent of DSP).
- MCU WAIT mode (independent of DSP).
- MCU RUN mode (independent of DSP).
- DSP STOP mode (independent of MCU).
- DSP WAIT mode (independent of MCU).
- DSP RUN mode (independent of MCU).
- Ability to shut down all peripherals (DSP and MCU) independently.
- Ability to shut down the REF_DPLL.
- Ability to shut down the USB PLL.
- Ability to shut down the crystal osciallator output driver.

For peripheral clock control details refer to Chapter 5, “Clock Control Module (CCM),” and each module chapter.

3.4.2 MCU Low Power Modes

The MCU system supports three different low-power modes: WAIT, DOZE, and STOP. The ARM has no low power mode instructions. Therefore any low power modes must be supplied externally. The low power modes are provided by a combination of CCM and CARB functionality. Refer to Section 3.4.5, “MCU Low Power Modes operation,” and the individual MCU peripheral specifications for more details on the low power features.

3.4.3 DSP Low Power Modes

The DSP supports two different low-power modes: WAIT, and STOP. Refer to the DSP56600 Users Manual for a detailed description of these modes. Refer to the individual DSP peripheral specifications for details on their low power features.

3.4.4 General Low Power Features

3.4.4.1 REF_DPLL Shut Down

The REF_DPLL may be shut down by software to reduce the current consumption.

3.4.4.2 Crystal Oscillator (CROSC) Output Driver Shut Down

The clock driver in the crystal oscillator may be shutdown by MCU software or the DSM. In such a case, the DSP and its peripherals are disabled. The MCU should change its clocking source to the CKIL prior to the shut down sequence to ensure that it can gracefully come out of stop mode. It is strictly required to put the DSP in its STOP mode prior to shut down, for proper operation. At reset, this driver is enabled to ensure proper reset of the DSP. If the MCU resets the DSP through the MDI, it should assure that the crystal oscillator driver is active and the DSP receives an operating clock.

3.4.5 MCU Low Power Modes operation

The CCM has the capability to disable the MCU GRPA, GRPB, and GRPC clocks during low power modes for power conservation. A disabled clock is held high. Since the ARM has no low power mode instructions, any MCU low power modes must be supplied externally. The MCU low power modes are provided by a combination of CCM and CARB functionality. The procedure for entering and exiting an MCU low power mode is described in the following sections.

3.4.5.1 Entering Low Power

To initiate low power modes, the MCU will write to low power enable bits in the CCM. There are bits to define STOP, WAIT and DOZE modes. A polling loop on the low power bits set should follow. The CCM will send a bus request to the Alternate Master arbiter. The Alternate Master arbiter will consider the CCM the lowest priority alternate bus master and consider it always at an instruction boundary. The CCM will drive TRANS=0 when it is a master (no actual bus cycle request), this is only to guarantee that there are no pending transactions. When there are no outstanding transactions, the arbiter in the ARM platform will grant the bus to the Alternate Master arbiter with the ap_bg signal. This action alone will disable the CPU clocks for DOZE and WAIT modes. The CCM will then disable the clocks appropriate to the low power mode. This will always include the single clock going to the ARM platform. (for alternate master access in WAIT, see Section 3.4.5.3) This action will shut off clock to the memories, reducing power drain.

3.4.5.2 Exiting Low Power

If exiting by reset, then the RESETB signal will reset all modules and this flow is not used. Otherwise, the CCM will monitor the ap_wakeup_b signal, which is an OR of interrupt request, debug request, and alt bus master request. An interrupt request will clear the low power mode bits in the CCM. Clearing the low power mode bits shall release the CCM bus request to the Alternate Master arbiter and startup the clock to the ARM platform. Clearing the request shall clear the bus grant from the Alternate Master arbiter to the CCM. With no alternate master now having a bus grant, the MCU will continue executing. Due to the interrupt request, the appropriate interrupt service routine will be the first thing executed by the MCU.

3.4.5.3 Alternate Bus Master During Low Power

An alternate bus master that was receiving clock cycles will send a bus request to the Alternate Master arbiter. The Alternate Master arbiter will see that this is a higher priority alternate master and remove the bus grant for the CCM. The CCM will apply the clock to the ARM platform. The Alternate Master arbiter

will send a bus grant to the alternate master which requested it. After the alternate master removes bus request, the Alternate Master arbiter will remove its bus grant. This is possible because the CCM bus grant is always considered to be at an instruction boundary. The Alternate Master arbiter will reassert the bus grant to the CCM The CCM will disable the clocks appropriate to the low power mode. See CCM module chapter for details.

3.5 Reset

3.5.1 Physical Reset Pin(s)

An input reset pin (RESET_IN) is provided on Neptune. The external reset input signal is active-low, and is typically connected to external power-on/low voltage detection circuitry. The external RESET_IN input signal is qualified as a valid reset if held low for at least four CKIL clock cycles. Upon de-assertion of the RESET_IN signal, reset output is used to reset external peripheral devices. The Reset_Out pin is driven low when either qualified external resets are detected, or an on-chip reset source (watchdog timer, power-up-reset circuitry, clock monitor) is generated. The minimum reset pulse width is eight CKIL clock cycles. functions with the same sequence as the power-up sequence detailed in the previous paragraph. The watch-dog or clock monitor reset is propagated to the internal reset circuitry, bypassing the four cycle reset qualifier.

3.5.2 Other Reset sources

Five(5) other sources are capable of generating a reset condition:

1. Internal Watchdog timer.
2. Internal power-on reset circuitry.
3. Internal clock monitor.
4. DSP OnCE reset - initiated by external JTAG command
5. Software reset

See the CCM module chapter for a detailed description of the system reset function.

3.5.3 Reset function/sequence

The reset sequence depends on the operational state of the chip when the reset occurs.

An internal POR circuit detects the power up condition and provides immediate reset to the “stretcher” input (see the CCM module chapter for details on the stretcher function), ensuring that all the on-chip circuits are properly initialized (i.e. all the MCU peripherals, DSP peripherals, MCU core and DSP core are at their reset state), and that the crystal oscillator is running properly. The MCU is clocked by the uncorrected PAT_REF and the DSP is clocked by the uncorrected 2*PAT_REF.

The negation of the external reset source is propagated by the reset circuit, and after being delayed by the reset pulse stretcher for the eight CKIL cycles and then sixteen cycles of the crystal oscillator output, it negates the internal MCU portion reset. This negation switches the 2*PAT_REF clock to be the input clock to the DSP portion. The reset for the DSP portion is held for sixteen more PAT_REF clock cycles and then it is released.

Under normal operating conditions, the POR circuit is not activated, thus an external reset trigger is first qualified for four CKIL clock cycles and only then is propagated to the internal reset circuitry, where it functions with the same sequence as the power-up sequence detailed in the previous paragraph.

The watch-dog, clock monitor, and software reset is propagated to the internal reset circuitry, bypassing the four cycle reset qualifier.

3.5.3.1 MCU Reset

An MCU reset performs a hardware reset of the core and resets all peripherals to their default states. The state of the MOD pin is latched when RESET_IN is negated and is used by the MCU to determine where to fetch the first memory value which is loaded onto the PC for the starting program address. The memory access is from either the internal ROM first address or the external memory connected to CS0. If the MOD pin is driven at a logic-low at least four CKIL clock cycles after RESET_IN negation, then the internal MCU ROM will be disabled and CS0 will be asserted for the first MCU cycle following reset negation. If MOD is driven logic-high at least four CKIL clock cycles before RESET_IN negation, then the internal ROM will be enabled, and the MCU will fetch the first word from internal ROM. MOD is configured with an internal pull-up resistor. Internal ROM will be configured to 0x0000_0000 address space if MOD is unconnected. See the MCU memory map configuration in Chapter 6 for more details.

3.5.3.2 DSP Reset

Any qualified MCU reset also resets the DSP section and all related peripherals to their default states. Any reset to the DSP should be accompanied by a clock source, otherwise a large amount of current may be consumed by the DSP until internal circuitry is reset to a convergent state. At hardware reset, the 2*PAT_REF clock is supplied to the DSP. Only after assuring a stable clock at the DSP clock input (sixteen PAT_REF clock cycles), the reset to the DSP portion is released.

When exiting reset, the DSP begins execution at $8000 (reset vector) in the DSP program memory.

The MCU can perform a hardware reset of the DSP section and related peripherals independent of the MCU section by asserting the DHR bit in the MCU-DSP interface (MDI).

The Command Vector may be used by the MCU to cause the DSP to jump to the reset vector to implement a soft reset if required. A soft reset does not reset the DSP peripherals.

Reset deassertion will have a signal dedicate in the pinlist so that a valid deassertaion may be seen N-clock cycles after reset.

3.5.3.3 REF_DPLL operation after reset

The REF_DPLL will be off when coming out of reset.

Chapter 4
USB Digital Phase Lock Loop (DPLL)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	05/07/03		Updated for LTE specification release.

The USB DPLL is a second instantiation of the PLL described in Chapter 34, “Digital Phase Lock Loop (DPLL).” The programming model for the USB DPLL is documented in Chapter 5, “Clock Control Module (CCM).”

Chapter 5
Clock Control Module (CCM)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
1.0	02/25/02	Satya Akella	Updated for CKIL monitor updated clock dividers removed dsp_cg_ctl register
2.0	04/18/02	Satya Akella	added notes for clock cleaner added notes for synchronous pat_ref generation updated timing diagrams for Low power modes
2.1	05/01/02	Satya Akella	added updates for qclk_byp for dclkg added sog_reset_b generation per DDTS DSPhl14037
2.2	05/07/02	Satya Akella	updated clock diagrams
2.3	07/22/02	Satya Akella	updated for dsm_int_holdoff logic
2.4	08/26/02	Satya Akella	DDTS: DSPhl15001, added clarification for MOD pin logic in the RESET LOGIC section
2.5	09/04/02	Mark Babcock, Shannon Osgood	Updated per DDTS# DSPhl15190. Updated references to CMON register.
2.6	09/30/02	Satya Akella	Added desense ckt functionality per DSPhl15120, DSPhl15635, DSPhl15369, DSPhl15316.
2.7	6 Nov 2002	Mark Babcock	Added notes to distinguish between Neptune LTS and Neptune LTE/ULS functionality. Corrected references to MFN_TOG_CNT to be consistent throughout.
2.8	12/06/02	Satya Akella	updated section 5.10, synchronous pat_ref generation per DSPhl15747

Revision History Table

Revision	Date	Author	Changes
2.9	02/05/03	Satya Akella	updated section 5.10, synchronous pat_ref generation
3.0	05/07/03		Updated for LTE specification release.

5.1 Overview

The major functions of the Clock Control Module (CCM) can be broken down as follows:

- Clock Muxing - Provides alternate clock sources for DSP/MCU platforms and peripherals.
- Clock Observability- Internal clocks can be routed to the CKO(H) pins for external visibility.
- Provides controls for powering down the CMON and ROSC (reference oscillator clock driver).
- Maintains control/operation registers for USB DPLL, REF DPLL
- Maintains post-dividers for generation of USB, MCU and REF clocks.
- Generates a system reset based on internal and external stimuli.
- Manages MCU low power modes (DOZE, STOP, WAIT), disabling peripheral clocks for power conservation.
- The CCM uses a 32-bit IP bus interface, and all its registers are byte-accessible.
- CKIL Clock Monitor - detects the disappearance of the CKIL clock.
- Provide quadrature clock bypass to the DSP clock generator (DCLKG).
- Provide a MFN value to the refpll that toggles between MFN_PLUS and MFN_MINUS for a programmed toggle rate when the desense function is enabled.

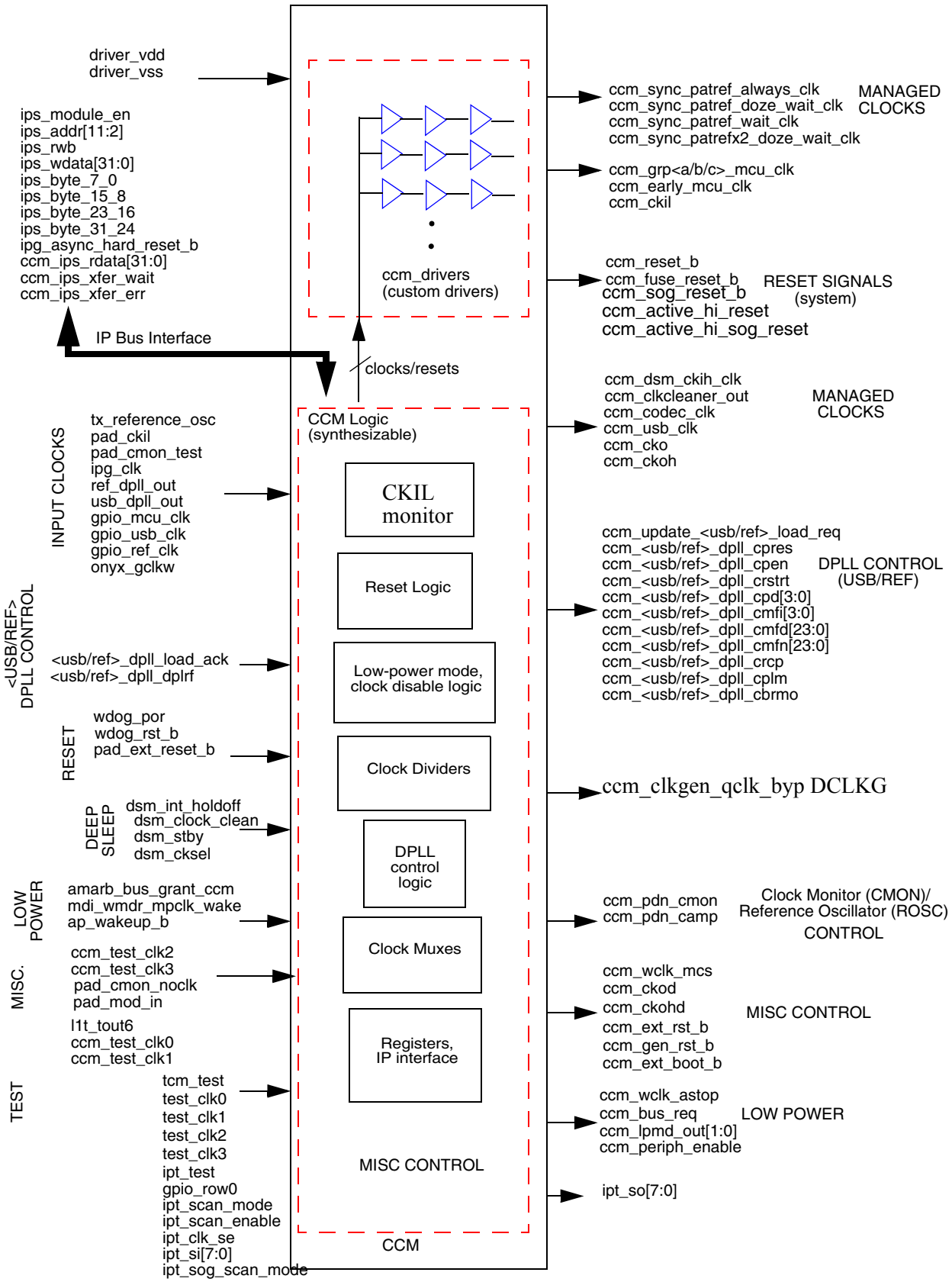


Figure 5-1. CCM Block Diagram

Table 5-1. CCM Module Pin List

Pin Name	Direction	Description
input clocks		
tx_reference_osc	input	26 MHz Crystal Reference oscillator clock (ROSC)
pad_ckil	input	32.768kHz clock
pad_cmon_test	input	Clock Monitor (CMON) digital clock
ipg_clk	input	IP module clock
ref_dpll_out	input	REF DPLL output
usb_dpll_out	input	USB DPLL output
gpio_mcu_clk	input	MCU bypass clock (GPIO signal)
gpio_usb_clk	input	USB bypass clock (GPIO signal)
gpio_ref_clk	input	REF bypass clock (GPIO signal)
onyx_gclkw	input	DSP clock. Used only for observability through CKO(H).
managed clocks		
ccm_sync_patref_always_clk	output	PAT_REF* (13 MHz) synchronized with ccm_grp(a,b,c)_mcu_clk, always on regardless of Low-Power mode,
ccm_sync_patref_doze_wait_clk	output	PAT_REF* (13MHz) synchronized with ccm_grp(a,b,c)_mcu_clk, off during STOP mode only
ccm_sync_patref_wait_clk	output	unsychronized pat_ref clock used as the source for usb dpll
ccm_sync_patrefx2_doze_wait_clk	output	PAT_REFx2* (26 MHz) off during STOP mode only
ccm_dsm_ckih_clk	output	PAT_REF*(13 MHz) synchronized with mcu_clk and not gated by low power modes (for DSM only)
ccm_grpa_mcu_clk	output	Group A MCU clock (see Table 5-15 for a description of group A)
ccm_grpb_mcu_clk	output	Group B MCU clock (see Table 5-15 for a description of group B)
ccm_grpc_mcu_clk	output	Group C MCU clock (see Table 5-15 for a description of group C)
ccm_early_mcu_clk	output	“Early” MCU clock for memories (leads group A,B,C mcu clocks)
ccm_clkcleaner_out	output	Output of clock cleaner (ROSC gated by DSM_STBY)
ccm_codec_clk	output	Clock source for CODEC module
ccm_usb_clk	output	48 MHz USB clock
ccm_cko	output	CKO mux selector output
ccm_ckoh	output	CKOH mux selector output
ccm_ckil	output	Buffered version of CKIL from pad

Table 5-1. CCM Module Pin List

Pin Name	Direction	Description
USB PLL control		
usb_dpll_load_ack	input	Indicates usb DPLL has acknowledged the CMFN update request
usb_dpll_dplrf	input	lock ready flag, Indicates that the usb DPLL is in lock.
ccm_update_usb_load_req	output	Request for usb DPLL to update CMFN
ccm_usb_dpll_cpres	output	usb DPLL reset
ccm_usb_dpll_cpen	output	usb DPLL enable
ccm_usb_dpll_crstrt	output	usb DPLL restart
ccm_usb_dpll_cpd[3:0]	output	usb DPLL Pre-division factor.
ccm_usb_dpll_cmfi[3:0]	output	usb DPLL Multiplication factor, Integer part
ccm_usb_dpll_cmfd[23:0]	output	usb DPLL Multiplication factor, Denominator part.
ccm_usb_dpll_cmfn[23:0]	output	usb DPLL Multiplication factor, Numerator part.
ccm_usb_dpll_crcp	output	usb DPLL Reference clock polarity
ccm_usb_dpll_cplm	output	usb DPLL Phase Lock Mode
ccm_usb_dpll_cbrmo	output	usb DPLL BRM Order
REF PLL control		
ref_dpll_dplrf	input	lock ready flag, Indicates that the ref DPLL is in lock.
ref_dpll_load_ack	input	Indicates ref DPLL has acknowledged the CMFN update request
ccm_update_ref_load_req	output	Request for ref DPLL to update CMFN
ccm_ref_dpll_cpres	output	ref DPLL reset
ccm_ref_dpll_cpen	output	ref DPLL enable
ccm_ref_dpll_crstrt	output	ref DPLL restart
ccm_ref_dpll_cpd[3:0]	output	ref DPLL Pre-division factor.
ccm_ref_dpll_cmfi[3:0]	output	ref DPLL Multiplication factor, Integer part.
ccm_ref_dpll_cmfd[23:0]	output	ref DPLL Multiplication factor, Denominator part.
ccm_ref_dpll_cmfn[23:0]	output	ref DPLL Multiplication factor, Numerator part.
ccm_ref_dpll_crcp	output	ref DPLL Reference clock polarity
ccm_ref_dpll_cplm	output	ref DPLL Phase Lock Mode
ccm_ref_dpll_cbrmo	output	ref DPLL BRM Order

Table 5-1. CCM Module Pin List

Pin Name	Direction	Description
clock monitor/amplifier control		
pad_cmon_noclk	input	“No Clock” signal from CMON; indicates loss of PAT_REF* clock
ccm_pdn_cmon	output	Power down CMON
ccm_pdn_camp	output	Power down reference oscillator (ROSC)
bus interface signals		
ips_module_en	input	IP module enable
ips_addr[11:2]	input	IP module address bus
ips_rwb	input	IP module read/write_b
ips_wdata[31:0]	input	IP module write-data bus
ips_byte_7_0	input	IP module byte enable for data[7:0]
ips_byte_15_8	input	IP module byte enable for data[15:8]
ips_byte_23_16	input	IP module byte enable for data[23:16]
ips_byte_31_24	input	IP module byte enable for data[31:24]
ipg_async_hard_reset_b	input	IP module asynchronous reset
ccm_ips_rdata[31:0]	output	IP module read-data bus
ccm_ips_xfr_wait	output	IP module wait-state request
ccm_ips_xfr_err	output	IP module invalid register access
reset signals		
wdog_por	input	Power-on reset signal from watchdog
wdog_rst_b	input	Internal reset signal from watchdog
pad_ext_reset_b	input	External reset from pad RESET_N
ccm_gen_rst_b	output	CCM-generated system reset
ccm_reset_b	output	system reset (generated or bypass) distributed to modules
ccm_fuse_reset_b	output	“early” reset for memories
ccm_sog_reset_b	output	reset for modules in sea-of-gates (SOG modules)
ccm_active_hi_reset	output	active hi system reset (inverse of ccm_reset_b)
ccm_active_hi_sog_reset	output	active hi sog reset (inverse of ccm_sog_reset_b)
ccm_ext_rst_b	output	Indicates that the external reset is asserted and qualified
low-power mode control		

Table 5-1. CCM Module Pin List

Pin Name	Direction	Description
amarb_bus_grant_ccm	input	Bus grant from Alternate Master Arbitrator
mdi_wmdr_mpclk_wake	input	from MDI, to enable peripheral clocks in low power mode
ap_wakeup_b	input	ARM Platform enable clock request
ccm_wclk_astop	output	peripheral clock disabled
ccm_bus_req	output	Alternate Master bus request
ccm_lpmd_out[1:0]	output	Low-Power Mode seen by the peripherals
ccm_periph_enable	output	Enable for peripheral clocks
misc. control signals		
dsm_clock_clean	input	Enable for clock cleaner
dsm_stby	input	DSM standby (power down)
dsm_clksel	input	DSM selection for corrected or uncorrected clock for PAT_REF*
dsm_int_holdoff	input	DSM interrupt holdoff - when asserted indicates CCM to not exit low power mode when ap_wakeup_b is asserted
pad_mod_in	input	mode pin (MOD) input
l1t_tout6	input	desense circuit enable from the L1timer
ccm_wclk_mcs	output	To watchdog, is low when ckil is selected as mcu clock
ccm_ext_boot_b	output	latched mode pin
ccm_ckod	output	CKO clock disable to GPIO
ccm_ckohd	output	CKOH clock disable to GPIO
ccm_clkgen_qclk_byp	output	quadrature clock bypass to DCLKG.
test signals		
ipt_test	input	asserts when in any test mode. Used to reset PAT_REF* clock
gpio_row0	input	ROW0 pin input. Used to reset PAT_REF* clock
tcm_test	input	ipt_test signal latched in TCM (should be driven from rising edge of mcu_clk)
ipt_scan_mode	input	scan mode signal for CCM
test_clk0	input	scan test clock 13MHz
test_clk1	input	scan test clock 26MHz
test_clk2	input	scan test clock 52MHz
test_clk3	input	scan test clock 130 MHz

Table 5-1. CCM Module Pin List

Pin Name	Direction	Description
ipt_scan_enable	input	scan enable for CCM
ipt_sog_scan_mode	input	sea of gates scan mode
ipt_clk_se	input	scan enable for clock gating cells
ipt_si[7:0]	input	scan chain inputs from STC
ipt_so[7:0]	output	scan chain outputs from CCM
ccm_test_clk3	input	130 MHz test clock for CCM flops
ccm_test_clk2	input	52 MHz test clock for CCM flops
ccm_test_clk1	input	26 MHz test clock for CCM flops
ccm_test_clk0	input	13 MHz test clock for CCM flops
CCM drivers signals		
driver_vdd	input	analog vdd to the clock drivers
driver_vss	input	analog vss to the clock drivers

*** PAT_REF is generated from ROSC clock or ref_pll output. Refer to figure 5-6, clock controls diagram**

5.2 Register Summary

The Register Summary lists all registers of the module by ascending address. The absolute address of each register is given, and the value of each bit for reads and writes is given using the conventions in the Legend.

NOTE:

New control and status registers REF_PLL_MFN_MINUS, REF_PLL_MFN_PLUS, MFN_TOG_CNT, and DESENSE_STAT describe enhanced functionality included on Neptune LTE.

KEY: Always Reads One (1) | Always Reads Zero (0) | Read/Write Bit | Read-Only Bit | Write-Only Bit | Write 1 to Clear (W1C) | Self-Clear Bit (0) | N/A

Table 5-2. CCM Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xxx_DPLL_CTL USB:($2484_5000) REF:($2484_5010)	R	0															
	W																
	R	0							LDREQ	0	PRE	PEN	RST	RCP	PLM	BRMO	LRF
	W																
xxx_DPLL_OP USB:($2484_5004) REF:($2484_5014)	R	0															
	W																
	R	0								MFI[3:0]				PDF[3:0]			
	W																
xxx_DPLL_MFN USB:($2484_5008) REF:($2484_5018)	R	0								MFN[23:0]							
	W																
	R	MFN[15:0]															
	W																
xxx_DPLL_MFD USB:($2484_500C) REF:($2484_501C)	R	0								MFD[23:0]							
	W																
	R	MFD[15:0]															
	W																
DFR ($2484_5020)	R	0															
	W																
	R	0			0	RDD[3:0]				UDD[3:0]				MCD[3:0]			
	W				RST												
CSR ($2484_5024)	R	0															
	W																
	R	0			DIS_GL	DPLL_BYP	REF_BYP	USB_BYP	MCU_BYP	DBG EN	QCLK BYP	CODS	REF_SEL	DCS[1:0]		MCS[1:0]	
	W																
CMON ($2484_5028)	R	0															
	W																
	R	CKOHD	CKOHS[2:0]			CKOD	CKOS[2:0]			CKIL MON EN	NO CKIL	STBY_CTL	NOCK	PDN CMON	PDN AMP	0	
	W																
LPMDR ($2484_502C)	R	0															
	W																
	R	0														LPMD[1:0]	
	W																

Table 5-2. CCM Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REF_PLL_MFN_MINUS REF:($2484_5030)	R	0								MFN_MINUS[23:16]							
	W																
	R	MFN_MINUS[15:0]															
	W																
REF_PLL_MFN_PLUS REF:($2484_5034)	R	0								MFN_PLUS[23:16]							
	W																
	R	MFN_PLUS[15:0]															
	W																
MFN_TOG_CNT REF:($2484_5038)	R	0								0			0	0			TOG_EN
	W																
	R	TOG_CNT[15:0]															
	W																
DESENSE_STAT :($2484_503C)	R	0	0	0	0	0	0	0	tog_sel	REF_PLL_MFN[23:16]							
	W																
	R	REF_PLL_MFN[15:0]															
	W																

5.3 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various Clock Control registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

5.3.1 Clock and PLL Registers

		Addr
USB_DPLL_CTL	DPLL Control Register	**$2484_5000**
REF_DPLL_CTL		**$2484_5010**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
								LDREQ		PRE	PEN	RST	RCP	PLM	BRMO	LRF
TYPE	r	r	r	r	r	r	r	rwm	r	rw	rw	rw	rw	rw	rw	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 5-3. DPLL_CTL Description

Name	Description	Settings
LDREQ Bit 8	**Load Request--** Notifies the DPLL that the numerator factor of the fractional part (MFN) has changed. This bit is cleared by an acknowledge from the DPLL. Writing a '0' to it has no effect. NOTE: Writing to this bit does not generate an automatic 'restart', whereas writing to any other bit in this register would. See Section 5.4.1, "x_CTL, x_OP, x_MFD Register Update,"	0 = DPLL has finished updating MFN(default). 1 = request that the DPLL updates MFN.
Bit 7	**RESERVED**	N/A
PRE Bit 6	**Power Up Reset --** Asserts a soft reset.	0 = reset cleared (default). 1 = reset asserted
PEN Bit 5	**PLL Enable —** Enables DPLL operation.	0 = DPLL disabled (default). 1 = DPLL enabled.
RST Bit 4	**Restart —** Restarts DPLL.	0 = DPLL not in restart (default). 1 = DPLL restarted.
RCP Bit 3	**Reference Clock Polarity —** Selects which edge the chip clock is adjusted to.	0 = positive edge of reference clock (default). 1 = negative edge of reference clock.
PLM Bit 2	**Phase Lock Mode**. **—** Selects if phase is to be considered (in addition to frequency) during lock-in. NOTE: For Neptune configuration the PLM bit must remain clear. Setting this bit will cause the DPLL output to be gated off when the phase lock is being lost.	0 = DPLL in frequency only lock mode (default). 1 = DPLL in frequency and phase lock mode.
BRMO Bit 1	**BRM Order Bit—** Binary Rate Modulator order	0 = BRM in first order. 1 = BRM in second order. (default)
LRF Bit 0	**Lock Ready Flag —** Shows if the DPLL is in lock.	0 = DPLL not locked. 1 = DPLL locked.

Registers DPLL_OP, DPLL_MFN, and DPLL_MFD calculate the output frequency by the formula:

$$f_{dck} = 2 \cdot f_{ref} \cdot \left[\frac{MF}{PDF}\right] \qquad \text{Where: } MF = MF_I + \frac{MF_N}{MF_D}$$

Please refer to Chapter 34, “Digital Phase Lock Loop (DPLL),” for further details.

USB_DPLL_OP
REF_DPLL_OP

DPLL Operation Register

$2484_5004
$2484_5014

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
									MFI[3:0]				PDF[3:0]			
TYPE:	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-4. DPLL_OP Description

Name	Description	Settings
MFI[3:0] Bits 7–4	**Multiplication Factor, Integer part (MF_I) -** If MFI[3:0] is written with a value of less than 5,then MF_I will default to 5.	**MFI[3:0]** / **MF_I** 0000 / 5 (default) . . / 5 0101 / 5 0110 / 6 . . . / . . . 1111 / 15
PDF [3:0] Bits 3–0	**Pre-division Factor (PDF) -** The value of PDF[3:0] is equal to the Pre-Divider Factor minus one.	**PDF[3:0]** / **PDF** 0000 / 1 0001 / 2 0010 / 3 . . . / . . . 1110 / 15 1111 / 16

Registers DPLL_OP, DPLL_MFN, and DPLL_MFD calculate the output frequency by the formula:

$$f_{dck} = 2 \cdot f_{ref} \cdot \left[\frac{MF}{PDF}\right] \qquad \text{Where: } MF = MF_I + \frac{MF_N}{MF_D}$$

Please refer to Chapter 34, “Digital Phase Lock Loop (DPLL),” for further details.

		Addr
USB_DPLL_MFN	DPLL MFN Register	**$2484_5008**
REF_DPLL_MFN		**$2484_5018**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
									MFN[23:16]							
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	MFN[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-5. DPLL_MFN Description

Name	Description	Settings
MFN [23:0] Bits 23–0	**Multiplication Factor Numerator Bits —** The MFN[23:0] bits, in a 2’s compliment format, give the numerator of the fractional part. NOTE: MF_N should be in a range from -8,388,606 to 8,388,606. If the absolute value of the MF_N is larger than MF_D, the output clock frequency will differ from the desired frequency.	

MFN[23:0]	MF_N
800000h	-8,388,608d
800001h	-8,388,607d
800002h	-8,388,606d
800003h	-8,388,605d
. .	. .
FFFFFFh	-1
000000h	0 (default)
000001h	1
. .	. .
7FFFFEh	8,388,606d
7FFFFFh	8,388,607d

Registers DPLL_OP, DPLL_MFN, and DPLL_MFD calculate the output frequency by the formula:

$$f_{dck} = 2 \cdot f_{ref} \cdot \left[\frac{MF}{PDF}\right] \qquad \text{Where: } MF = MF_I + \frac{MF_N}{MF_D}$$

Please refer to Chapter 34, “Digital Phase Lock Loop (DPLL),” for further details.

USB_DPLL_MFD
REF_DPLL_MFD

DPLL MFD Register

$2484_500C
$2484_501C

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
									MFD[23:16]							
TYPE	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	MFD[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-6. DPLL_MFD Description

Name	Description	Settings
MFD[23:0] Bits 23-0	**Multiplication Factor Denominator Bits—** The MFD[23:0] bits, in a 2's compliment format, give the denominator of the fractional part. If MFD[23:0] is written with 0, then MF_D defaults to 1. NOTE: Bit 23 is always 0. MF_D should be in a range from 1 to 8,388,606, otherwise the output clock frequency will differ from the desired frequency.	

MFD[23:0]	MF_D
000000h	1 (default)
000001h	1
000002h	2
. .	. .
7FFFFEh	8,388,606d
7FFFFFh	8,388,607d

Registers DPLL_OP, DPLL_MFN, and DPLL_MFD calculate the output frequency by the formula:

$$f_{dck} = 2 \cdot f_{ref} \cdot \left[\frac{MF}{PDF}\right] \qquad \text{Where: } MF = MF_I + \frac{MF_N}{MF_D}$$

Please refer to Chapter 34, "Digital Phase Lock Loop (DPLL)," for further details.

DFR Division Factor Register **$2484_5020**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
				RST	RDD[3:0]				UDD[3:0]				MCD[3:0]			
TYPE:	r	r	r	slfclr	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0

Table 5-7. DFR Description

Name	Description	Settings
RST Bit 12	**Restart Dividers -** Restarts all 3 divided clocks from the same state. This affects the divided clocks only (not divide-by-1), and would not affect the division factor settings. This bit is self-clearing. This bit can only be activated in TEST mode. NOTE: When writing to this register to set this bit, the divided clocks will be held at '0' for the duration of the write cycle.	0 = (default) 1 = Restart clock division.
RDD [3:0] Bits 11-8	**REF DPLL Divide -** Divides output of REF DPLL	
UDD [3:0] Bits 7-4	**USB DPLL Divide -** Divides output of USB DPLL	
MCD [3:0] Bits 3-0	**MCU Clock Source Divide-** Divides output of the selected MCU clock source to create MCU_CLK_IN.	

[3:0]	divide by
1XXX	1 (default)
0000	2
0001	3
0010	4
0011	5
0100	6
0101	8
0110	10
0111	12

CSR Clock Selection Register **$2484_5024**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
				DIS_GL	DPLL_BYP	REF_BYP	USB_BYP	MCU_BYP	DBGEN	QCLK BYP	CODS	REF_SEL	DCS[1:0]		MCS[1:0]	
TYPE:	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

Table 5-8. CSR Description

Name	Description	Settings
Bits 15-13	**RESERVED**	N/A
DIS_GL Bit 12	**Disable Glitchless Muxes-** Disables the glitchless feature of switching clock muxes. When set, the select line(s) will cause the mux to switch immediately regardless of the state of the input clocks.	0 = Glitchless muxes will function normally, synchronously disabling and re-enabling the output while the mux switches.(default). 1 = Muxes will switch immediately when select line changes.
DPLL_BYP Bit 11	**DPLL Bypass-** Functional test bit to test the ref, usb dividers.	0 = DPLL clock is passed to dividers (default). 1 = Clock synchronous to XTAL is passed to dividers.
REF_BYP Bit 10	**REF DPLL Bypass-** Selects the source for CCM_REF_DPLL_OUT.	0 = REF_DPLL_OUT comes from REF_DPLL (default). 1 = REF_DPLL_OUT comes from GPIO_REF_CLK.
USB_BYP Bit 9	**USB DPLL Bypass-** Selects if USB_CLK comes from USB_DPLL or from GPIO	0 = USB_CLK comes from USB_DPLL (default). 1 = USB_CLK comes from GPIO_USB_CLK.
MCU_BYP Bit 8	**MCU_CLK_IN Bypass-** Selects if MCU_CLK_IN comes from the divided output of div_clk mux or from GPIO	0 = MCU_CLK_IN comes from divided MCU clock source (default). 1 = MCU_CLK_IN comes from GPIO_MCU_CLK.
DBGEN Bit 7	**Debug Enable -** When set, this will keep the group A mcu_clk active during all MCU low-power modes. This will keep the ARM clock active during low-power, although the ARM still will not have control of the bus.	0 = grpa_mcu_clk will be gated off during low-power modes (default). 1 = grpa_mcu_clk will remain active during low-power modes.
QCLK_BYP	**Quadrature clock bypass-** Controls whether the quadrature clocks are generated within the DCLKG module or if the generation is bypassed and the quadrature clocks are provided from an external source through the GPIO	0 = Quadrature clocks are generated within the DCLKG module (default). 1 = Quadrature clocks are provided by an external source through the GPIO
CODS Bit 5	**CODEC Clock Select-** Selects if the source for CODEC_CLK is the uncorrected clock (clk_cleaner_out - 26 MHz) or the corrected clock (REF_DPLL_OUT divided down to 26 MHz).	0 = CODEC_CLK is the corrected clock. 1 = CODEC_CLK is the uncorrected clock.(default)

Table 5-8. CSR Description

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>REF_SEL
Bit 4</td><td>PAT_REF Source Select- Selects if the source for PAT_REFx2 is the “raw” uncorrected 26 MHz clock, or the corrected clock (REF_DPLL_OUT divided down to 26 MHz).
NOTE: DSM CKSEL output contributes to this selection</td><td>0 = PAT_REFx2 is the uncorrected clock.
1 = PAT_REFx2 is controlled by DSM (default). If CKSEL (DSM) =0 (default), then the uncorrected clock is selected. If CKSEL=1, then the corrected clock is selected.

NOTE: The clock out of this mux will go through a 1/2 divider to create PAT_REF.</td></tr>
<tr><td>DCS [1:0]
Bits 3-2</td><td>Divided MCU Clock Source Select - Selects
the source for the divided clock that provides MCU_CLK_IN.</td><td><table><tr><th>DCS[1:0]</th><th>Select Source</th></tr><tr><td>00</td><td>PAT_REF (13 MHz)</td></tr><tr><td>01</td><td>REF_DPLL_OUT</td></tr><tr><td>10</td><td>USB_DPLL_OUT</td></tr><tr><td>11</td><td>PAT_REFx2 (26 MHz)</td></tr></table></td></tr>
<tr><td>MCS [1:0]
Bits 1-0</td><td>MCU Clock Select - Selects the source clock for the MCU_CLK groups.

NOTE: When TEST is asserted, MCU_CLK = GPIO_MCU_CLK, and MCS is ignored</td><td><table><tr><th>MCS[1:0]</th><th>Select Clock</th></tr><tr><td>00</td><td>Inverted PAT_REF</td></tr><tr><td>01</td><td>MCU_CLK_IN*</td></tr><tr><td>1X</td><td>Inverted CKIL</td></tr></table></td></tr>
</table>

* MCU_CLK_IN = GPIO_MCU_CLK when MCU_BYP is asserted. In normal operation MCU_CLK_IN should be programmed for the divided output of the Divided MCU Clock Source mux (DCS[1:0]).

CMON Clock Monitor Register **Addr $2484_5028**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	CKOHD	CKOHS[2:0]			CKOD	CKOS[2:0]			CKILMONEN	NOCKILCLK	STBY_CTL	NOCK	PDNCMON	PDNAMP		
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rwm	rw	r	rw	rw	r	r
RESET	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0

xxxxx

Table 5-9. CMON Description

Name	Description	Settings
CKOHD Bit 15	**CKOH Disable -** Gates the CKOH signal.	0 = CKOH is enabled(default). 1 = CKOH is disabled.
CKOHS [2:0] Bits 14-12	**CKOH Clock Select -** Selects the clock for the CKOH pin. NOTE: CKOH default is PAT_REF	**CKOHS[2:0]** / **Select Clock** 000 — MCU_CLK (GRPA) 001 — MCU_CLK (GRPB) 010 — USB clock 011 — REF_DPLL_OUT 100 — n/a 101 — PAT_REF 110 — CMON_TEST 111 — DSP_CLK (gclkw)
CKOD Bit 11	**CKO Disable -** Gates the CKO signal	0 = CKO is enabled(default). 1 = CKO is disabled.

Table 5-9. CMON Description

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>CKOS [2:0]
Bits 10-8</td><td>CKO Clock Select - Selects the clock for the CKO pin.

NOTE: CKO default is GRPA_MCU_CLK.</td><td>
<table>
<tr><th>CKOHS[2:0]</th><th>Select Clock</th></tr>
<tr><td>000</td><td>MCU_CLK (GRPA)</td></tr>
<tr><td>001</td><td>MCU_CLK (GRPB)</td></tr>
<tr><td>010</td><td>USB clock</td></tr>
<tr><td>011</td><td>REF_DPLL_OUT</td></tr>
<tr><td>100</td><td>n/a</td></tr>
<tr><td>101</td><td>PAT_REF</td></tr>
<tr><td>110</td><td>CMON_TEST</td></tr>
<tr><td>111</td><td>DSP_CLK (gclkw)</td></tr>
</table>
</td></tr>
<tr><td>CKIL_MON_EN
Bit 7</td><td>CKIL monitor enable - Enables the CKIL clock monitor</td><td>0 = CKIL clock monitor is disabled(default).
1 = CKIL clock monitor is enabled.</td></tr>
<tr><td>NO_CKIL_CLK
Bit 6</td><td>NO CKIL clock flag - This flag is set by the CKIL clock</td><td>0 = CKIL clock monitor is disabled or flag is cleared by software(default).
1 = CKIL clock is not present.</td></tr>
<tr><td>STBY_CTL
Bit 5</td><td>DSM STANDBY Control--Allows for a “psuedo-Deep Sleep Mode” in which DSM is asserted, but does not shut down ROSC (reference oscillator clock) or the Clock Cleaner output.</td><td>0 = DSM_STBY is passed to ROSC (default).
1 = DSM_STBY is not passed on to ROSC.</td></tr>
<tr><td>NOCK
Bit 4</td><td>No PAT_REF clock</td><td>0 = PAT_REF present
1 = PAT_REF not present</td></tr>
<tr><td>PDN_CMON
Bit 3</td><td>Power Down CMON</td><td>0 = CMON on
1 = CMON off (default)</td></tr>
<tr><td>PDNAMP
Bit 2</td><td>Power Down Clock Amplifier</td><td>0 = Reference oscillator clock available (default).
1 = Reference oscillator disabled.</td></tr>
<tr><td>Bit 1-0</td><td>RESERVED</td><td>N/A</td></tr>
</table>

x xxx x x x x

LPMDR Low-Power Mode Register **Addr $2484_502C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
															LPMD[1:0]	
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rwm	rwm
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

xxxx

Table 5-10. LPMDR Description

Name	Description	Settings
Bits 15-2	**RESERVED**	RESERVED
LPM D[1:0] Bits 1-0	**Low-Power Mode -** Indicates the current operating mode of the mcu and peripheral clocks. Refer to **Chapter 3, "Clocks, Low Power Control and Reset (CLKRST),"** for a description of each mode	

LPMD[1:0]	Mode
00	STOP
01	WAIT
10	DOZE
11	RUN

NOTE:

REF_PLL_MFN_MINUS Register describes enhanced functionality included on Neptune LTE.

REF DPLL MFN MINUS Register

Addr $2484_5030

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
									MFN_MINUS[23:16]							
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	MFN_MINUS[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-11. DPLL_MFN MINUS Description

Name	Description	Settings
MFN_MINUS [23:0] Bits 23–0	**Multiplication Factor Numerator Bits —** The MFN_MINUS[23:0] bits, in a 2's compliment format, give the numerator of the fractional part. NOTE: MF_N should be in a range from -8,388,606 to 8,388,606. If the absolute value of the MF_N is larger than MF_D, the output clock frequency will differ from the desired frequency.	**MFN_MINUS[23:0]** / **MF_N** 800000h / -8,388,608d 800001h / -8,388,607d 800002h / -8,388,606d 800003h / -8,388,605d . / . FFFFFFh / -1 000000h / 0 (default) 000001h / 1 . / . 7FFFFEh / 8,388,606d 7FFFFFh / 8,388,607d

NOTE:

REF_PLL_MFN_PLUS Register describes enhanced functionality included on Neptune LTE.

REF DPLL MFN PLUS Register

Addr $2484_5034

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
									MFN_PLUS[23:16]							
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	MFN_PLUS[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-12. MFN PLUS register

Name	Description	Settings
MFN_PLUS [23:0] Bits 23–0	**Multiplication Factor Numerator Bits —** The MFN_PLUS[23:0] bits, in a 2's compliment format, give the numerator of the fractional part. NOTE: MF_N should be in a range from -8,388,606 to 8,388,606. If the absolute value of the MF_N is larger than MF_D, the output clock frequency will differ from the desired frequency.	

MFN_PLUS[23:0]	MF_N
800000h	-8,388,608d
800001h	-8,388,607d
800002h	-8,388,606d
800003h	-8,388,605d
. .	. .
FFFFFFh	-1
000000h	0 (default)
000001h	1
. .	. .
7FFFFEh	8,388,606d
7FFFFFh	8,388,607d

NOTE:

MFN_TOG_CNT Register describes enhanced functionality included on Neptune LTE.

MFN TOG CNT Register — **Addr $2484_5038**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
																TOG_EN
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	TOG_CNT[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-13. MFN TOG CNT Register

Name	Description	Settings
Bits 31-17	**RESERVED**	RESERVED
TOG_EN Bit 16	TOG_EN- desense functionality enable	tog_en = 1, desense ckt is enabled tog_en = 0, desense ckt is disabled.
TOG_CNT[15:0] Bits 15-0	toggle counter value.	

TOGCNT[15:0]	count value
0000	0
0001	1
0002	2
0003	3
...	...

NOTE:

DESENSE_STAT Register describes enhanced functionality included on Neptune LTE.

DESENSE STAT Register — **Addr $2484_503C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
								TOG_SEL	MFN[23:16]							
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	MFN[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-14. DESENSE STAT Register

Name	Description	Settings
TOG_SEL BIT 24	Toggle sel status	**tog_sel = 1,** desense ckt is enabled **tog_sel = 0**, desense ckt is disabled
MFN [23:0] Bits 23–0	MFN value that is sent to the ref_dpll.	

5.4 DPLL Control

5.4.1 x_CTL, x_OP, x_MFD Register Update

The USB and REF dpll require special timing considerations when updating their ports. When the dpll control register x_CTL (with exception to LDREQ), x_OP, or x_MFD is written (even if the bits don't change), a restart is issued to the dpll before the actual control bits are sent to the DPLL. The restart assertion and port update will be synchronous to the reference input clock of the dpll (REF_DPLL:= pat_refx2, USB_DPLL:=pat_ref). Writing to any of these registers will cause following procedure to come into effect:

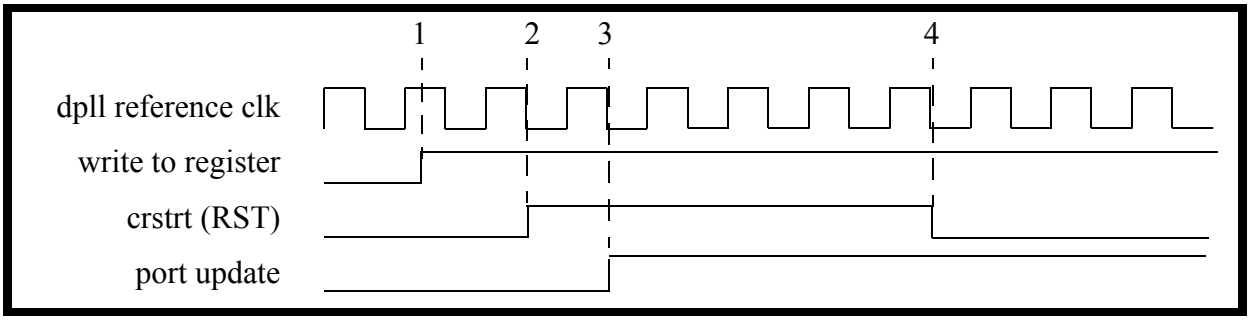


Figure 5-2. DPLL Port Timing

1. Register is written
2. One clock cycle later restart (crstrt) is asserted
3. One clock cycle later the appropriate port is updated
4. Four clock cycles later restart is released

NOTE: Setting the RST bit will cause the crstrt signal to immediately assert.

5.4.2 x_MFN Register Update

The MFN bits (in register DPLL_MFN) are the only bits in the DPLL that can be changed after the DPLL was locked without reset. After writing a new value to MFN, the procedure to update the DPLL is:

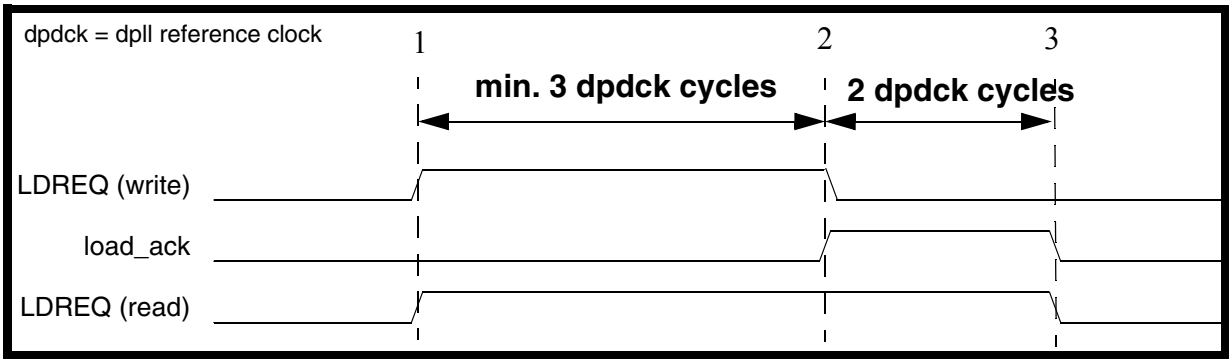


Figure 5-3. MFN Load Request Timing

1. The control bit LDREQ is set. This sends a request signal to the DPLL. NOTE: Must only write to byte1 of DPLL; if byte0 is also written to, a restart will be asserted.
2. When the DPLL loads the new MFN, its sends an acknowledge signal back to the CCM and clears the write value of LDREQ. At this point LDREQ becomes a status bit, reflecting the state of the acknowledge signal. NOTE: This write value of LDREQ is a “sticky” bit. When set, it will remain at ‘1’ until a load_ack pulse is received.
3. The read value of LDREQ is automatically cleared when the DPLL has completed updating MFN and has removed the acknowledge signal, allowing the next load request to take place.

Please refer to Chapter 34, “Digital Phase Lock Loop (DPLL),” for further details.

5.4.3 DPLL Reset Logic

In a standard operating mode the DPLL can be reset from two sources. A soft reset by setting PRE, or a hard reset by asserting RESET_IN. During a scan mode reset can be set by asserting RESET_IN . CCM provides the software reset PRE to the DPLL. The logic to reset the DPLL resides in the DPLL.

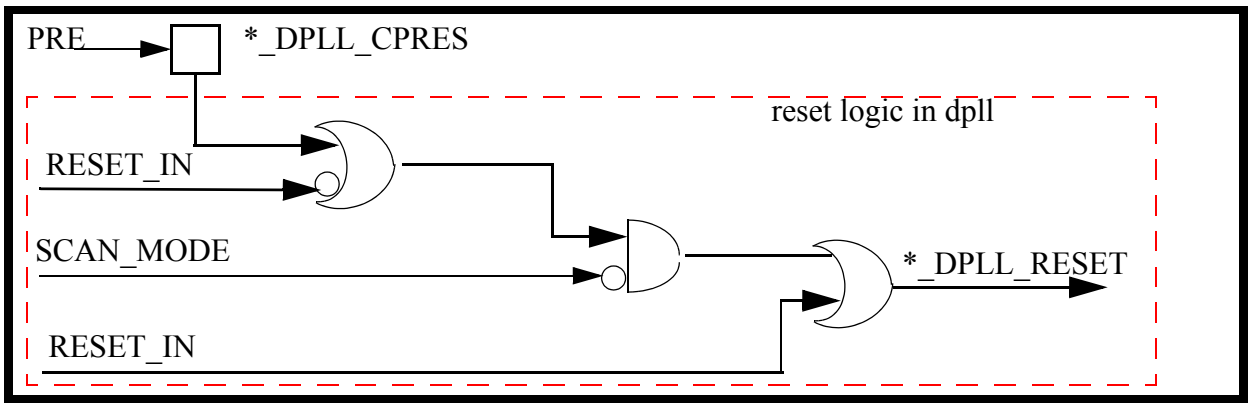


Figure 5-4. DPLL Reset Logic

5.5 Clock Control

5.5.1 Clock Cleaner - Deep Sleep Mode

The clock cleaner provides a 26 MHz reference clock used by the digital subsystem of the chip. This reference clock is a gated version of the crystal reference oscillator (ROSC). When Deep Sleep mode is entered, the clock cleaner output will be synchronously gated off by dsm_gate_b ("dsm_clock_clean" in DSM spec.) and the reference oscillator will be disabled. Waking up from Deep Sleep will release the clock. **All clocks derived from the output of the clock cleaner will be stable during shutdown and wakeup**. It is possible to create a "psuedo-deep sleep" mode where the chip thinks it is in Deep Sleep, but the clocks remain active. This is accomplished by setting stby_ctl (bit 5 of CMON) which prevents the ROSC or the Clock Cleaner from being disabled.

Alternatively, the oscillator can be immediately disabled by setting pdn_amp (bit 2 of CMON). This bit does not affect the clock cleaner other than disabling the reference oscillator (ROSC).

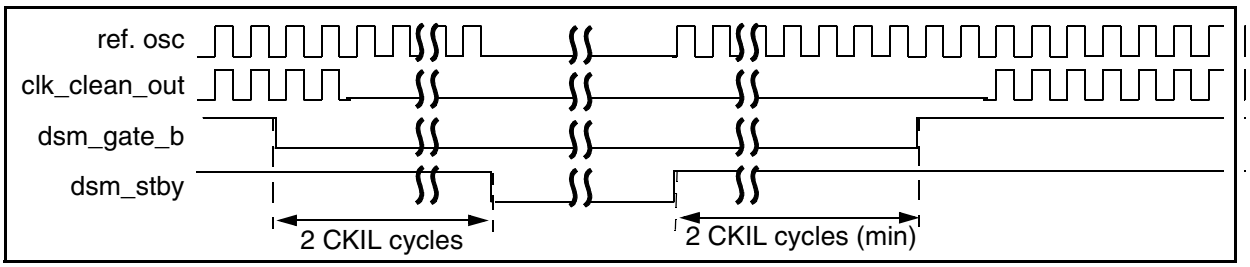
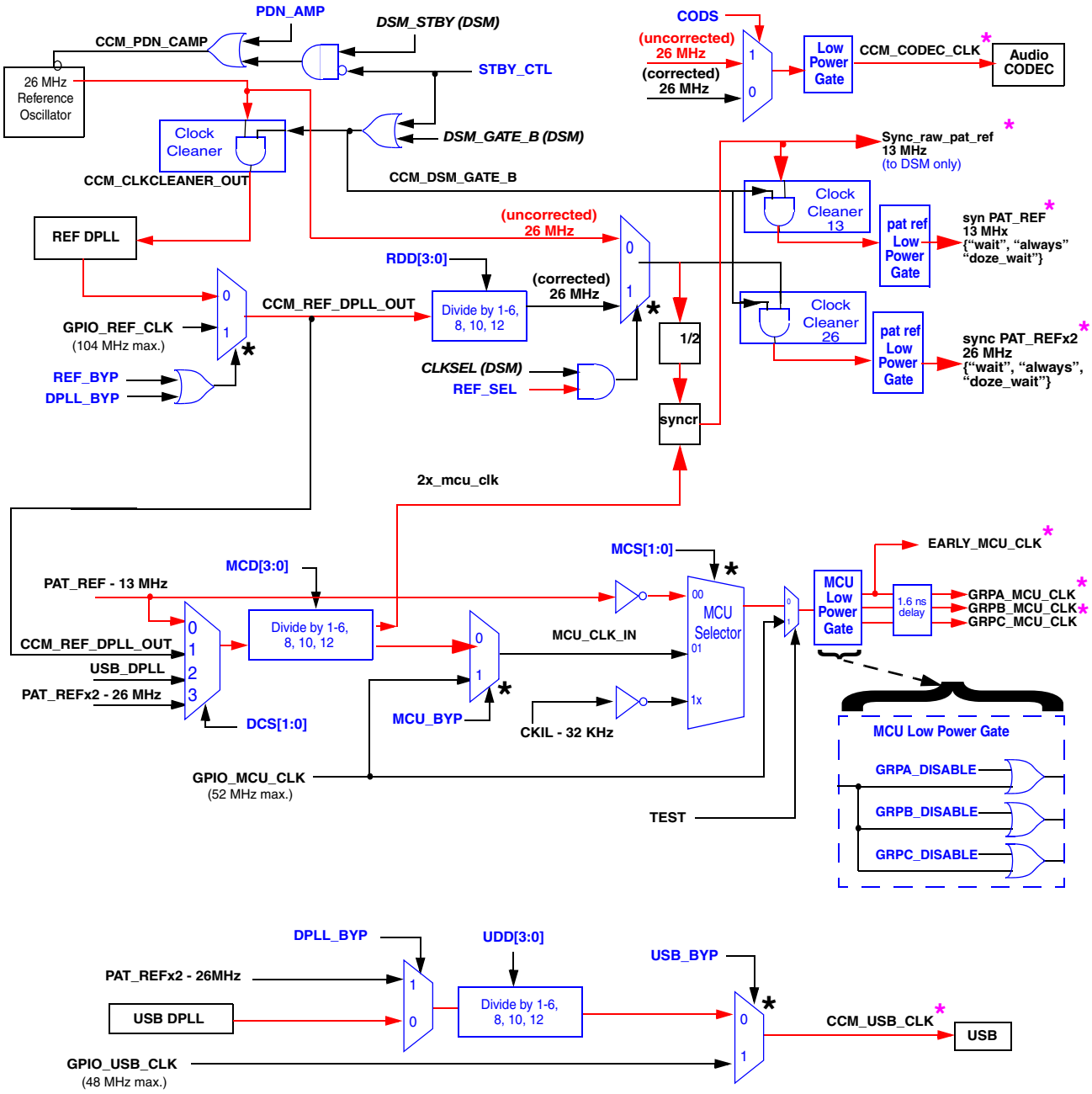


Figure 5-5. Deep Sleep/Clock Cleaner Relationship

NOTE: The dsm receives raw_pat_ref clock instead of the pat_ref_always clock. pat_ref_always clock is disabled by the DSM before entering the Deep Sleep Mode.On entering the deep sleep mode, the oscillator and hence the raw_pat_ref clock are disabled. DSM cannot work if it uses the clock (pat_ref_always) that it disables. Therefore, DSM receives the raw_pat_ref clock so that it can continue to work and be able to restore the pat_ref clock for the rest of the chip.

Please refer to Chapter 52, "Deep Sleep Module (DSM)," for more details.



* All the functional clocks provided by the CCM are muxed with the corresponding test_clk within the CCM. The test muxes within peripherals which muxed the test clock with the functional clock from CCM no longer exist. For clocks generated within modules, the muxing would be done within the modules.

Asterisk (*) denotes glitchless mux.
Blocks/controls in BLUE are implemented by CCM.
RED arrows designate path enabled on reset.

Figure 5-6. Clock Controls Diagram

5.5.2 Clock Muxes

There are a multitude of muxes available throughout the clock routes, providing alternate clock sources for the system clocks controlled by the CCM. These alternate clock sources can come from internally or externally (through GPIO) generated clocks.

The CCM utilizes several glitchless clock muxes. The definition of Glitchless for the CCM is that between transition of two clocks, there will be no pulses that are of a frequency higher than either clock. During this transition, the output of the mux will be held high for a maximum of two cycles of ROSC (reference oscillator clock, tx_reference_osc) and two-and-a-half cycles of the newly selected clock

In order for the glitchless mux to function, both the current clock and the clock to be selected must remain active during the entire selection process. **Not doing so may cause a loss of clock with no recovery!** Any clock may be inactive anytime other than when changing from one clock to another. When disabling pll, programmer must switch to ckil, or another active clock prior to powering down.

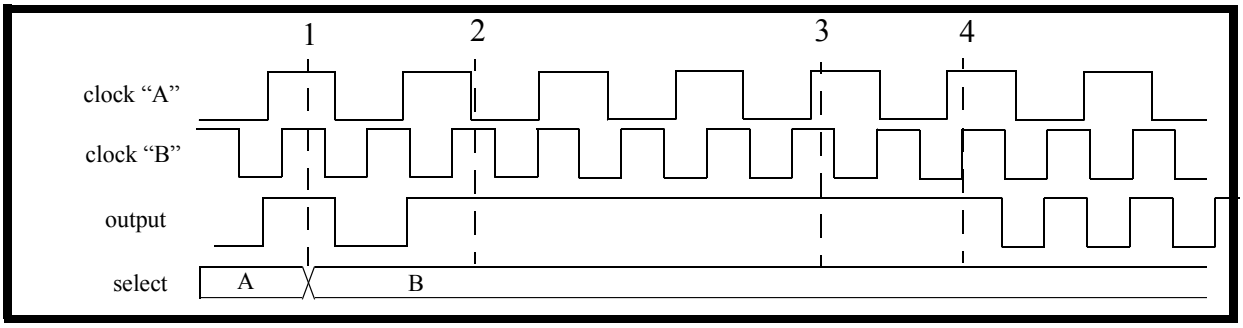


Figure 5-7. Glitchless Clock Mux Timing

1. New clock is requested
2. Current clock is gated off. T_2-T_1 = maximum 2 clock "A" cycles (1 cycle min.)
3. Mux switches and allowed to stabilize. T_3-T_2 = maximum 2 ROSC(tx_reference_osc clokc) cycles (1 cycles min.)
4. Selected clock appears at output. T_4-T_3 = maximum 2.5 clock "B" cycles (1.5 cycle min.)

NOTE: All input clocks (clocks "A", "B", and ROSC(tx_reference_osc)) must be active for the glitchless muxes to switch. If a clock loss occurs due to switching a glitchless mux when an input clock was inactive, it is possible to recover the output clock by disabling the glitchless feature of the mux and switching the mux back to its previous setting. Setting bit 12 (dis_gl) of the CCM Clock Selection Register (CSR) will immediately disable the synchronizers and transform all glitchless muxes in the CCM into 'standard' muxes where the clock would switch immediately after the mux select changed and the output clock could glitch. However, if mcu_clk was the clock that was lost, it cannot be recovered, since mcu_clk would be needed to set dis_gl. It is also not possible to enter TEST mode if mcu_clk is lost, since the "TEST" signal provided to the CCM doesn't come directly from the pad; it is first registered by the TCM using mcu_clk.

5.5.3 Test Mode

When any test mode is active (ipt_test asserted) the MCU_CLK is replaced by GPIO_MCU_CLK. While in test mode, changing MCU_BYP or the mcs[1:0] bits will not affect MCU_CLK. The low power gating of the MCU_CLK groups are also ignored during test mode.

During any scan test (ipt_scan_mode active), the quadrature clock mux select going to DCLKG (ccm_clkgen_qclk_byp) will automatically be asserted to ensure that the DCLKG can provide the bypass clocks when the CCM is held in reset by the STC(scan test controller block).

Another feature during test mode is to provide direct control for resetting the flip-flop that divides pat_refx2 to create pat_ref. The objective is to give the tester control over the phase of pat_ref when coming out of reset. Normally, this flop is reset by only power-on reset. But in test mode (pin ipt_test asserted), if the pin ROW0 is held low during the reset sequence, then the reset for the pat_ref flop is directly controlled by the pin RESET_IN.

NOTE: To return to functional mode without executing a system reset, the GPIO needs to be configured to continue to provide GPIO_MCU_CLK prior to exiting test mode. Otherwise the value of tcm_test will not be updated after test mode is exited, and the CCM will still think it's in test mode but MCU_CLK will NOT be available.

5.5.4 Clock Dividers

The CCM has three glitchless clock dividers. They are located at:

- USB_DPLL. Divides output of the usb dpll. Can be used for the usb clock.
- REF_DPLL. Divides the output of the ref dpll mux. Can be used for pat_ref.
- MCU_CLK_SRC. Divides the output of the div_clk_src mux. Can be used for the mcu_clk.

Frequencies of divide by 1, 2, 3, 4, 5, 6, 8, 10 or 12 of the input clock frequency can be achieved. After a change occurs in the clock selection signals, a change of frequency will occur at a maximum of four input clock cycles (when switching from a divide-by-1), or three output clock cycles (when switching from a divide-by-n, n>1).

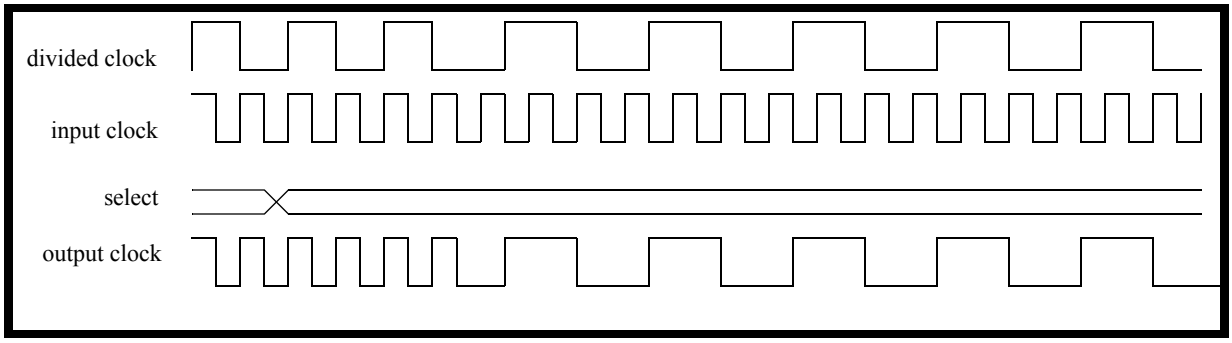


Figure 5-8. Divide-by-1 to Divide-by-3

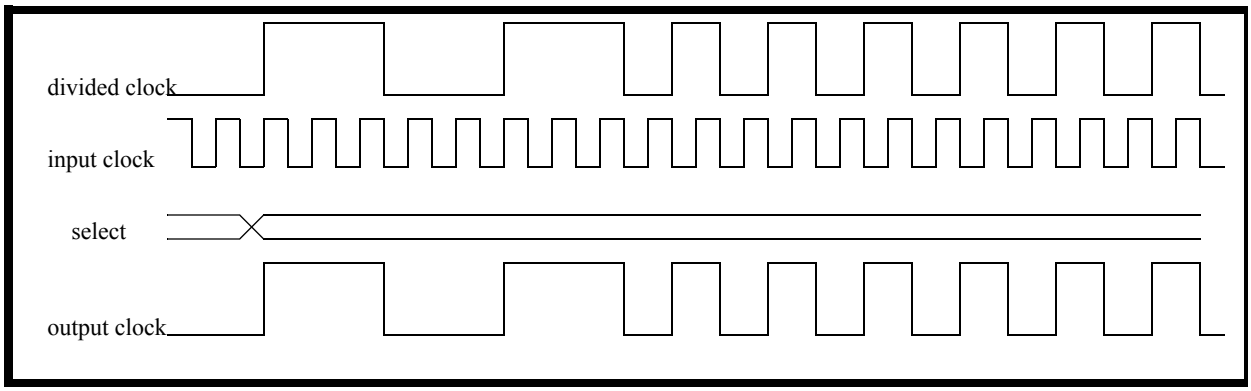


Figure 5-9. Divide-by-5 to Divide-by-2

5.6 Clock Characteristics

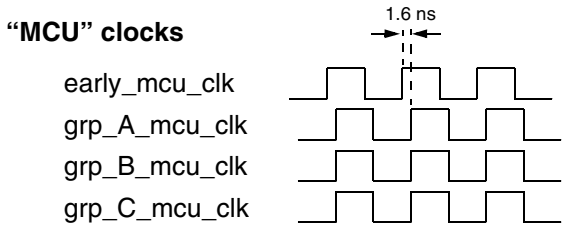


Table 5-15. Characteristics of CCM Clocks During System Events

CLOCK (actual net name is "ccm_<CLOCK>")	Reset Freq.	Clock Source During TEST	MCU Low-Power Mode (all clocks restored on int, debug, or alt-mas req.)				Deep Sleep
			STOP	DOZE	WAIT	restore on MDI req	
ccm_early_mcu_clk	13 Mhz	gpio	high	high	high	no	off[3]
ccm_grpa_mcu_clk	13 Mhz	gpio	high	high	high	no	off[3]
ccm_grpb_mcu_clk	13 Mhz	gpio	high	high	high	yes	off[3]
ccm_grpc_mcu_clk	13 Mhz	gpio	high	high	running	no	off[3]
clk_cleaner_out	26 Mhz	n/a	n/a	n/a	n/a	n/a	low
sync_patrefx2_doze_wait_clk	26 Mhz	n/a	high	running	running	yes	low[3]
ccm_dsm_ckih_clk	13 Mhz	n/a	running	running	running	n/a	low
ccm_codec_clk	26 Mhz	n/a	low	running	running	yes	off[3]
sync_patref_always_clk	13 Mhz	n/a	running	running	running	n/a	off
sync_patref_doze_wait_clk	13 Mhz	n/a	high	running	running	n/a	off
sync_patref_wait_clk	13 Mhz	n/a	high	high	running	yes	off[3]
ccm_usb_clk	low[1]	n/a	n/a[3]	n/a[3]	n/a[3]	n/a	off[3]
dsp clocks[2]	13 Mhz	n/a	off[3]	running[3]	running[3]	yes	off[3]

[1]This is due the DPLL being disabled.

[2]Not a part of CCM, but wanted to show anyway for reference

[3]Unless a bypass clock source is selected, in which case the off state cannot be pre-determined.

5.7 Low Power Mode

The CCM has the capability to disable the mcu clocks (groups A,B,C,early), pat_ref clocks, codec and clkgen clocks during low-power modes for power conservation. The three modes in which this can happen are STOP, DOZE and WAIT, and the clocks affected are described in Section 5.7.1 through Section 5.7.3.

Since the ARM has no low power mode instructions, any low power modes must be supplied externally. The low power modes will be provided by a combination of CCM and CARB functionality. The procedure for entering and exiting a low power mode is described in the following sections.

5.7.1 Entering a low power mode

The MCU will write the desired low-power mode to the CCM register LPMDR. The CCM will then issue a bus_request to the AMARB and wait for bus_grant_ccm to be asserted. When bus grant is received, the CCM will update ccm_lpmd, the low-power mode signals seen by the rest of the system, and disable the appropriate mcu_clk groups no sooner than 7 cycles later. (see Figure 5-10). Low-power mode cannot be entered if the chip is currently servicing a debug or interrupt request (ap_wakeup_b asserted).

If a low-power mode is requested when the MDI is requesting the clocks, then low-power mode will still be entered, indicated by a change in ccm_lpmd[1:0], but the clocks that can be restored by an MDI request (see Table 5-15) will remain active until after the MDI request is removed.

If a low-power mode is requested by an alternate master, that mode will not be entered until the alternate master relinquishes the control of the bus.

NOTE: wclk_astop asserts one cycle before mcu_clk is disabled (or at lease 6 cycles after low-power mode is entered). It is asserted when entering ANY low-power mode and stays asserted until the low-power mode has been exited.

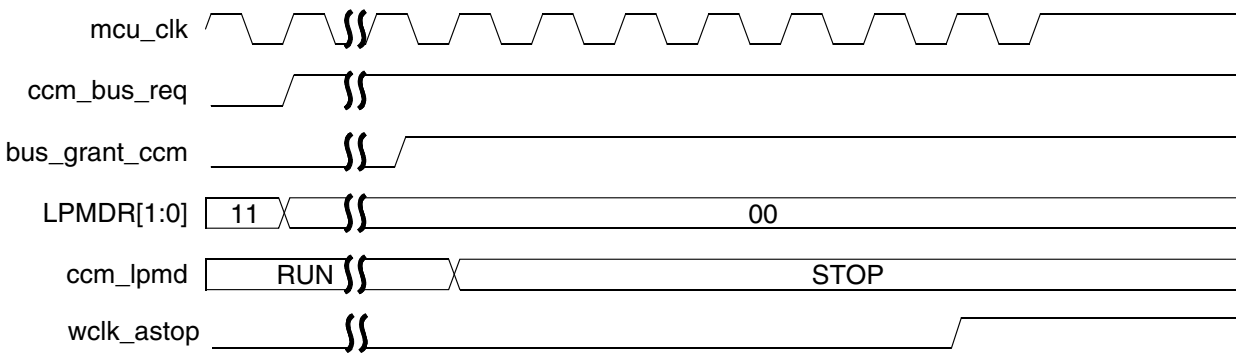


Figure 5-10. Entering Low-Power mode

5.7.2 Restoring Clocks while in low-power mode

Once the MCU is in a low-power mode, a wakeup request from the MDI (mpclk_wake assertion), or an alternate master request (bus_grant_ccm de-assertion) can restore clocks that have been disabled. (See Table 5-15). These requests will not change the status of the low-power mode (ccm_lpmd[1:0] will not change), nor will they affect wclk_astop. The clocks will be re-enabled no more than 2 1/2 mcu_clk cycles following an mpclk_wake assertion (if MDI requested clocks) or 1 1/2 mcu_clk cycles following a bus_grant_ccm deassertion (if alternate master requested clocks). The clocks will stay enabled at least for 4 mcu_clk cycles after the request is removed (mpclk_wake deassertion, or bus_grant_ccm assertion). (See Figure 5-12)

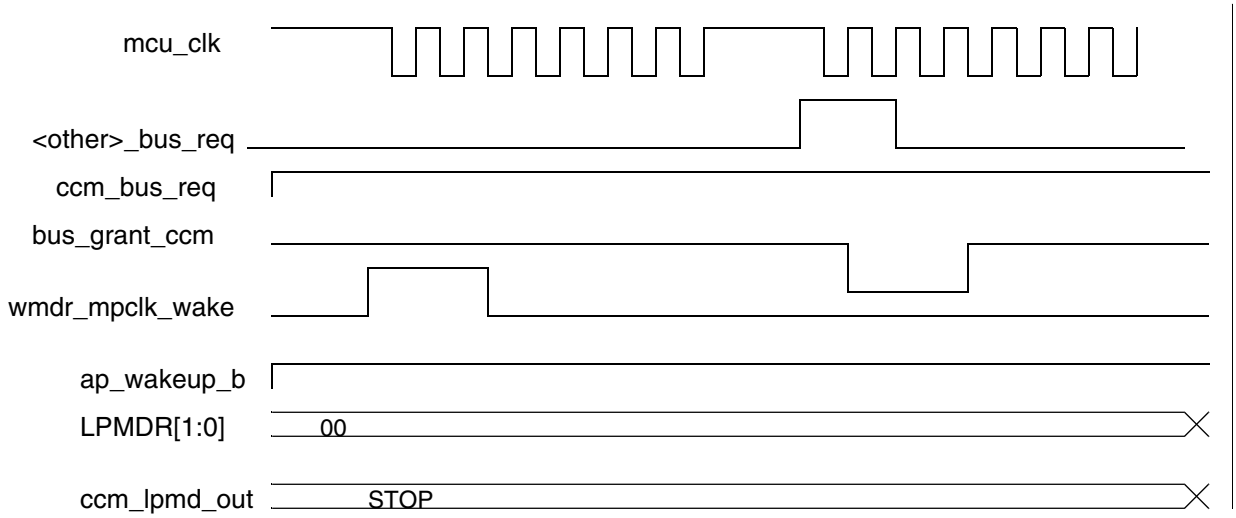


Figure 5-11. Restoring Clocks while in low power mode

NOTE: During low-power mode, the CCM is similar to an alternate master, and the mcu_clk going to the ARM core will be gated off by the CARB. This clock will remain gated off as long as an alternate master has control of the bus. This means that even though the clocks provided by the CCM can be restored while in low-power mode (by MDI or another alternate master), the ARM core will not receive a clock until low-power mode is actually exited.

5.7.3 Exiting a low power mode

All the disabled clocks will be restored when low-power mode is exited. Setting the low-power mode back to RUN (ccm_lpmd[1:0]='11') can be accomplished by any of the following methods.

- interrupt request
- debug request
- alternate master writing '11' to LPMDR
- system reset

If an interrupt or debug request has been made, indicated by the assertion of ap_wakeup_b(and dsm_int_holdoff is de-asserted), then the clocks will be re-enabled no more than 3 1/2 mcu_clk cycles later. An interrupt or debug request (ap_wakeup_b asserted) cannot re-enable the clocks if the dsm_int_holdoff signal is asserted. To ensure proper exit from low-power mode, ap_wakeup_b should stay asserted for at least 2 clock cycles after the mcu clocks are restored. This will be followed by returning ccm_lpmd back to RUN mode. RUN mode can also be established by a system reset, or by writing '11' to LPMDR via alternate master.

NOTE: When writing to LPMDR to change the low-power mode, only changes from RUN to [STOP,DOZE, or WAIT], or from [STOP, DOZE, or WAIT] to RUN are allowed. For example, going from WAIT to STOP is not allowed.

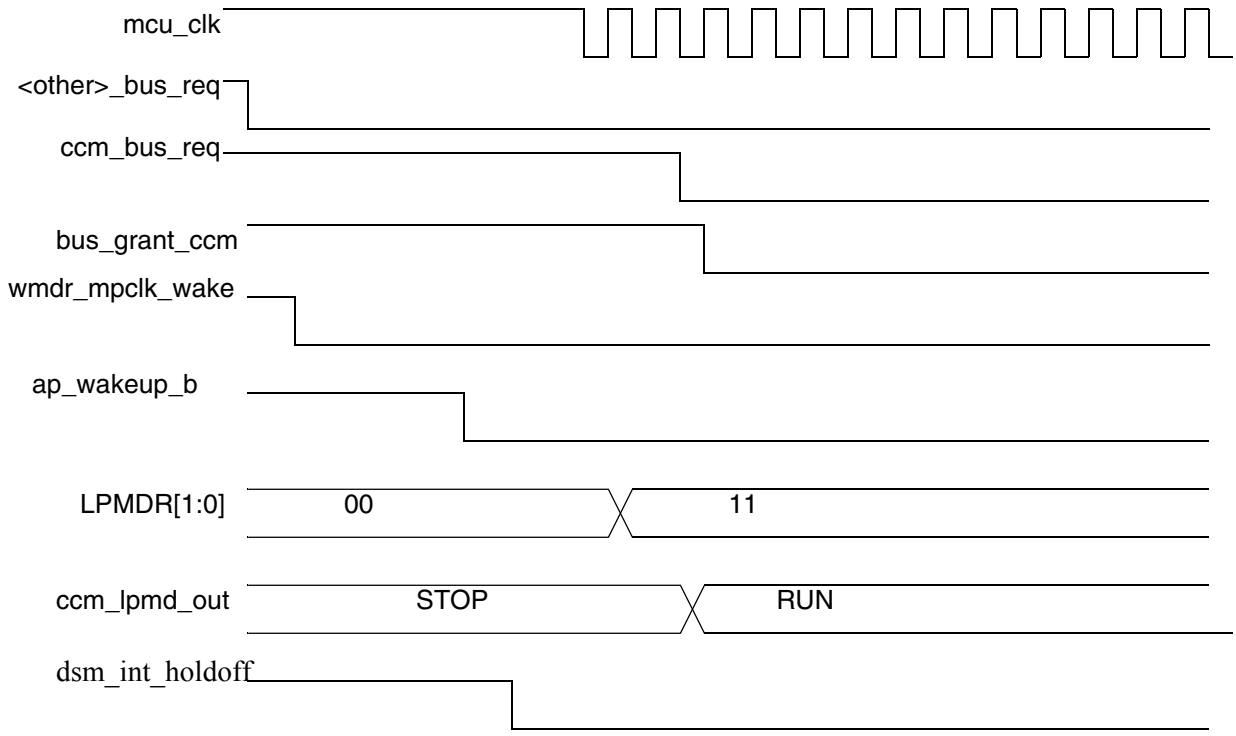


Figure 5-12. Re-enabling clocks and exiting low-power mode

NOTE: MCU Low-power mode does not directly affect the DPLL output clocks, nor does it affect ROSC (crystal reference oscillator clock, tx_reference_osc).

5.8 CKO(H) Selection

CKO and CKOH are external pins which will give visibility to several clocks. The muxes for each pin, as well as the selection options, are identical.The clock selected can be disabled by setting its respective disable bit in the Clock Monitor Register (CCM_CMON).

The allowed clock sources to be routed to CKO(H):

- GRPA_MCU_CLK
- GRPB_MCU_CLK
- PAT_REF
- CCM_REF_DPLL_OUT
- USB_CLK
- CMON_TEST
- DSP_CLK (gclkw)

The CKO(H) selectors are controlled by CKO(H)S[2:0] which are in the Clock Monitor (CMON) register. The CKO(H) selected outputs can be disabled at the pad by setting CKO(H)D

NOTE: The CKO(H) selectors are not glitchless.

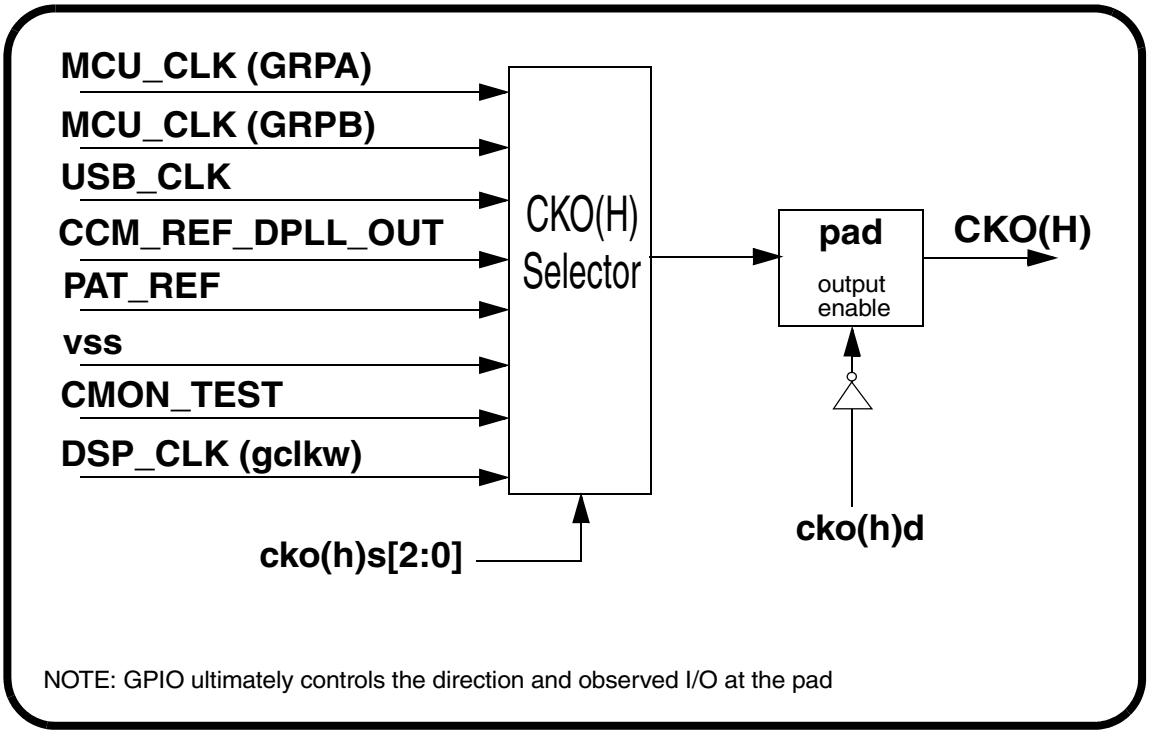


Figure 5-13. CKO(H) Selection

5.9 Reset Logic

The CCM supplies reset to the chip as CCM_RESET_B. There are two categories of reset:

- Qualified reset - These resets will not be recognized unless asserted for at least 4 CKIL cycles (the counter starts from the next CKIL rising edge after reset is asserted). There is only one qualified reset:
  - External reset pin (RESET_IN)
- Immediate reset - These resets are recognized immediately after assertion. There are five possible immediate resets acknowledged by the CCM:
  - Power on reset
  - Loss of clock from clock monitor
  - Watchdog time-out
  - Software reset
  - Reset due to JTAG instruction

The system reset (ccm_reset_b) will assert itself immediately after any reset source is recognized (any_valid_reset_b). After all of the reset sources are released, the CCM will start a reset-stretching sequence that causes the system resets to release at a maximum 9 CKIL and 16 ROSC(tx_reference_osc clock) cycles later (ccm_fuse_reset_b will release at least one CKIL cycle prior to system reset deassertion). The system reset is guaranteed to assert for a minimum of 8 CKIL cycles and 15 ROSC (tx_reference_osc) cycles after a reset source is recognized.(see table 5-13, reset deassertion timings)

The Reset Logic is also responsible for latching the mod pin (MOD). The latch is transparent and reflects the value of the MOD pin when any valid reset is asserted. The value of the MOD pin will be latched four CKIL cycles after all of the reset sources are released. This signal is used to determine if the core should boot from internal or external memory.

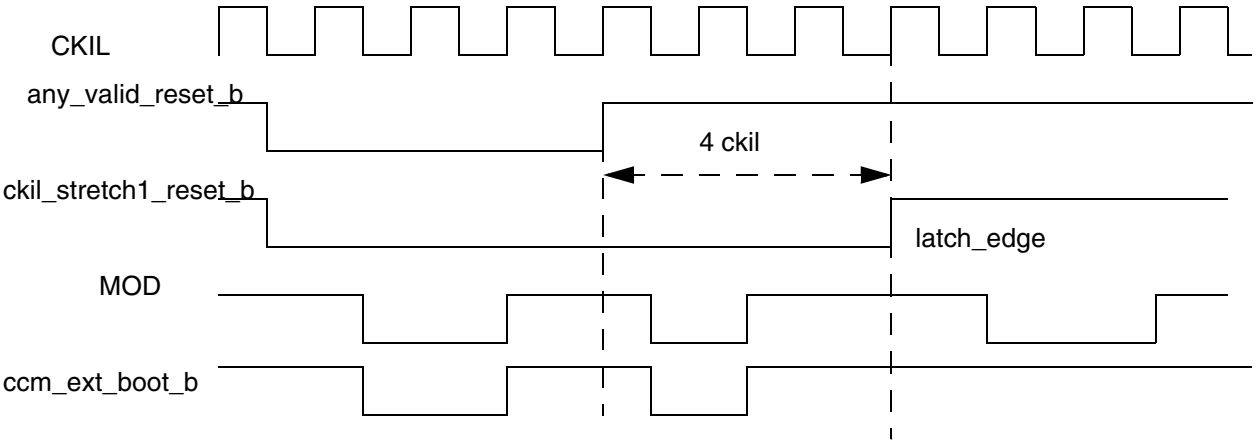


Figure 5-14. MOD PIN LOGIC

NOTE: The watchdog combines all the immediate resets mentioned above into one signal (wdog_rst_b) and provides this to the CCM

When in scan mode, the ccm generated reset is bypassed. RESET_IN drives the system reset.

CCM provides a separate reset ccm_sog_reset_b for the IP modules. When in functional mode or when the modules in the sea of gates are scanned, the ccm_sog_reset_b is same as the system reset(ccm_reset_b). However, when other modules in the chip other than the sea of gates are scanned, the modules in the sea of gates are held in reset.

The reset logic in CCM is scannable. Due to this scan logic for reset, an automatic system reset (reset stretching sequence occurs) is generated whenever scan mode is exited.

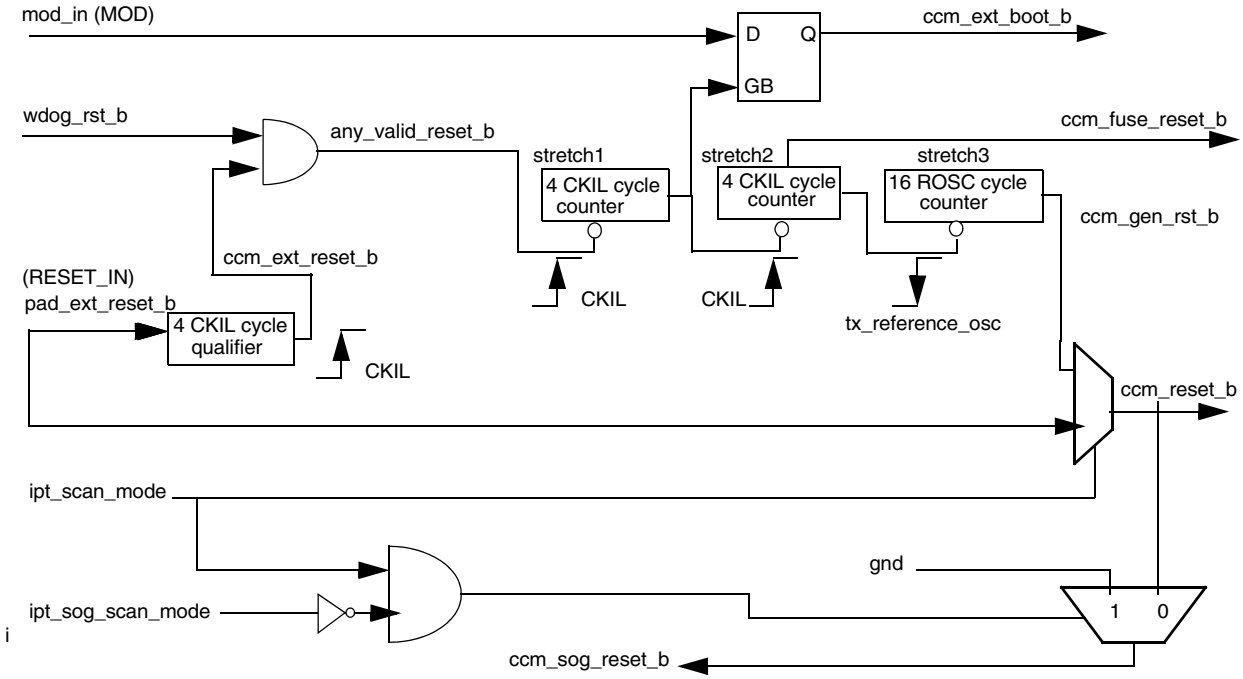


Figure 5-15. RESET Logic

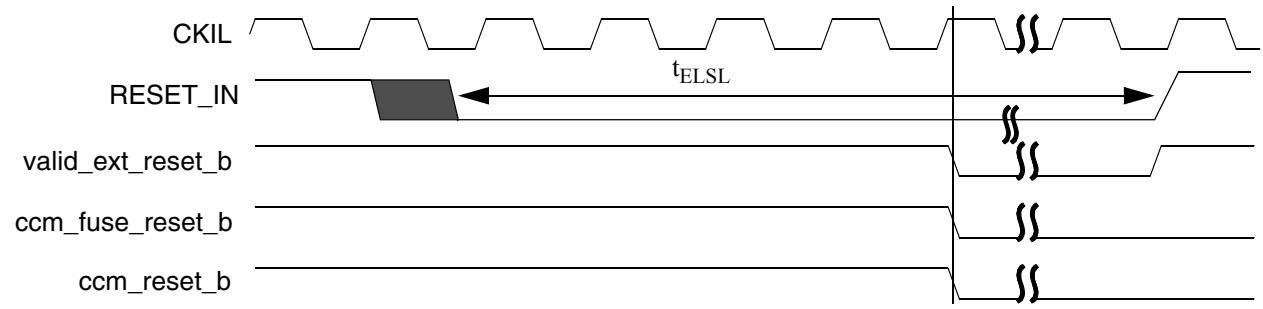


Figure 5-16.

Table 5-16. Reset Assertion Timings

	Symbol	MIN	MAX	Unit
RESET_IN assertion to system reset assertion	t_{ELSL}	122* (4 CKIL)	152.5 (5 CKIL)	us

NOTE: Assertion of an immediate reset (wdog_reset_b) will cause the system and fuse resets to assert immediately.
*the counter starts from the next CKIL rising edge after reset is asserted.

Table 5-17. Reset Deassertion Timings

	Symbol	MIN	MAX	Unit
Deassertion of all valid resets to fuse reset deassertion I	t_{VHFH}	183 (7 CKIL)	213.5 (8 CKIL)	us
Fuse reset deassertion to system reset deassertion II	t_{FHSH}	31.67 (1 CKIL + 15 ROSC*)	31.75 (1 CKIL + 16 ROSC*)	us
Deassertion of all valid resets to system reset deassertion III	t_{VHSH}	214.67 (8 CKIL + 15 ROSC*)	244.75 (9 CKIL + 16 ROSC*)	us

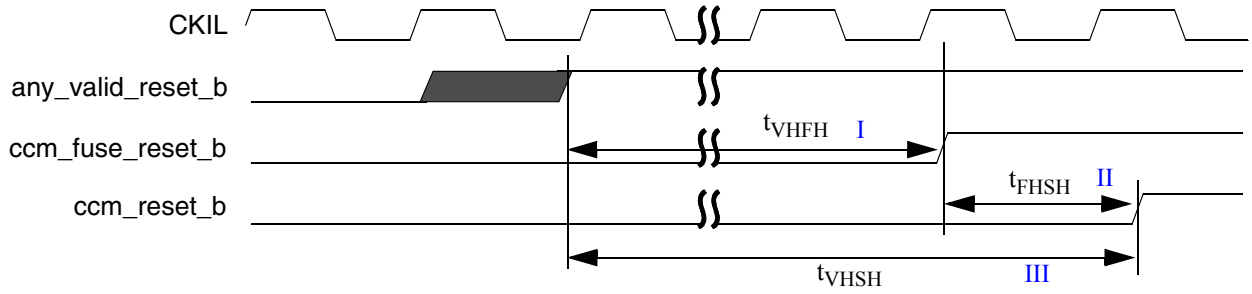


Figure 5-17.

*see the timing diagram below for timing relationship between CKIL and ROSC (tx_reference_osc) for the ROSC stretching sequence.

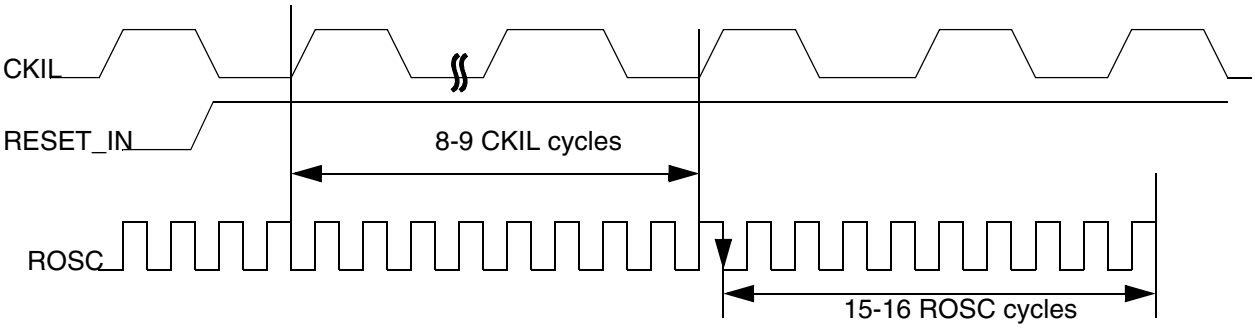


Figure 5-18.

5.9.1 Synchronized System Reset Deassertion

Deassertion of any ccm generated reset signal is synchronized with the falling edge of ROSC (pat_refx2, 26 MHz clock), and will occur only during the low phase of pat_ref (13 MHz). During reset, mcu_clk will default back to the inverse of pat_ref, where it will remain until programmed differently.

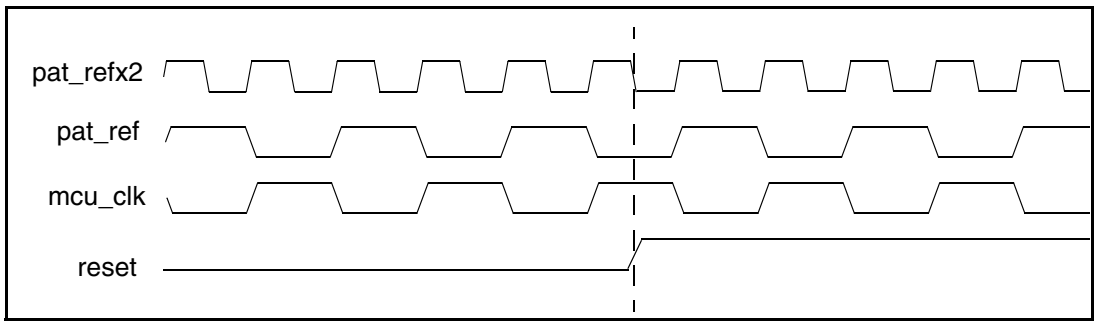


Figure 5-19. Synchronization of Reset Deassertion

5.10 Synchronized PAT_REF generation

Peripherals which used the JIG interface earlier and now reside on the IP bus require the 13 MHz pat_ref clock to be synchronous to the mcu clock. In order to achieve a synchronous pat_ref with minimum jitter, the pat_ref clock has to be sampled with a clock that is atleast four times its frequency. MCU clock however can be programmed to various frequencies from 52MHz down to 13 MHz. The mcu clock divider is designed to provide a 2xdivided output (2x mcu_clk) in addition to the divided output(mcu_clk).This 2x divided clock output is used to synchronize the pat_ref. Thus when mcu_clk is set to frequencies between 52MHz and 13 MHz with the mcu divider set to atleast a divide-by-2, the 2xdivided clock still has a frequency higher than the pat_ref inorder to generate a synchronous pat_ref. Synchronization is achieved when the source of the mcu clock is either the ref_dpll or the usb_dpll, mcu clock divider is set to a divide ratio greater than 1 and when not in mcu bypass mode and when the mcs mux selects the clock divider output as the source of the mcu_clock. If the above conditions are not satisfied, the synchronizer is bypassed. In short when mcs[1:0] is 01, mcu_byp is 0, RDD[3:0] is not 8 and DCS[1:0] is not 00 or 11, the synchronizer is in use.

Note: If the pat_ref clock is an odd multiple of 2xmcu_clk, since the sampling occurs for odd number of cycles, pat_ref can have a jitter of upto 1/2 cycle.

The clock_cleaner output and the ccm_codec_clk will not be synchronized.

5.11 Bus Interface

The clock controller connects to the IP bus, and all registers are byte-accessible.

5.12 CKIL Clock Monitor (32.768 kHz Nominal)

The CKIL clock monitor is used to detect the disappearance or a failure of the CKIL clock in a given period of time. When the CKIL clock edge is not detected for a period of 60 us (approximately 2 CKIL cycles), the CKIL monitor will assert a NO_CKIL_CLK (high) flag. The NO_CKIL_CLK flag status is written to bit 6 of the CCM CMON register. Once set, NO_CKIL_CLK is cleared by software by writing a zero to bit 6 of the CCM CMON register or when the CKIL monitor is disabled.

The CKIL clock monitor will work for any typical input clock frequency of 32.768 kHz. The CKIL monitor is enabled by setting CKIL_MON_EN = 1 (bit 7 of CCM CMON register). Software will typically enable the monitor for several seconds.

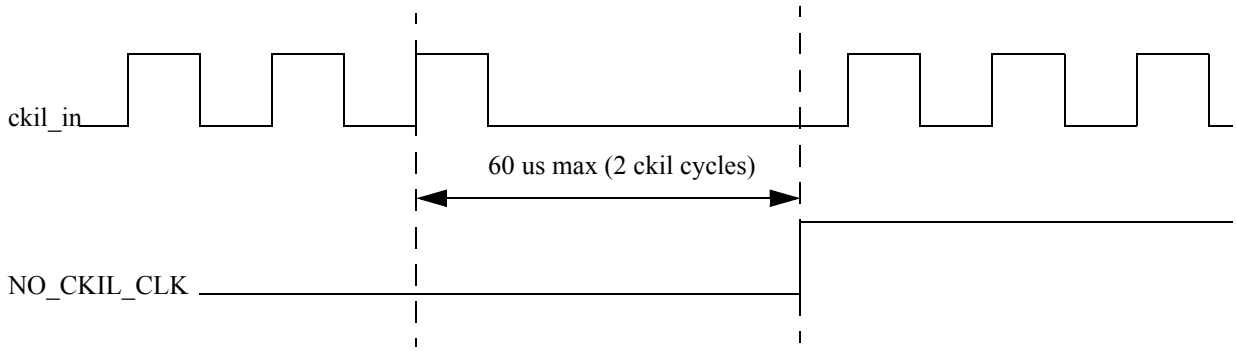


Figure 5-20. CKIL Clock Monitor Timing Diagram

5.13 CCM Desense Circuit

NOTE:

The enhanced functionality of the CCM Desense Circuit is included in Neptune LTE.

Radiation of clock harmonics causes receiver desensitization. Controlled phase modulation of the clocks coming out of the REFPLL will reduce this desensitization. The desense circuit when enabled provides a MFN value that toggles between two programmed MFN_PLUS (MFN value + delta) and MFN_MINUS (MFN value - delta) for a programmed toggle frequency. This will help achieve the phase modulation of the REFPLL clocks.

When enabled by programming the toggle enable bit, the CCM provides the REFPLL with a MFN value that switches between MFN_PLUS and MFN_MINUS for n number of correct pat_ref cycles. MFN_PLUS and MFN_MINUS are programmed by writing to REF_PLL_MFN_PLUS and REF_PLL_MFN_MINUS registers. Bits [15:0] of the TOG_CNT register determines the toggle rate n. The REFPLL receives a MFN value equal to MFN_PLUS for the first n/2 cycles and MFN_MINUS value for the remaining n/2 cycles. When the toggle enable bit is de-asserted, the circuit completes the MFN_PLUS, MFN_MINUS cycle before exiting the desense mode. Only even values for the counter are allowed. If the counter is programmed to an odd value, the counter value will default to a value one less from the odd value.

The desense mode can be activated either by writing to bit 16 of the MFN_TOG_CNT register or by the L1TIMER (l1t_tout6).

When in desense mode, CCM generates an automatic load request (ccm_update_ref_load_req) to the REFPLL whenever the MFN value changes. This signal stays asserted until the load acknowledge (ref_dpll_load_ack) signal from the REFPLL is received. The load_req asserts for a minimum of 3 pat_ref cycles. Hence for a counter value of 6, where the MFN value toggles between MFN_PLUS and MFN_MINUS every 3 pat_ref cycles, the load_req may not reflect the updated MFN value and therefore, there is a possibility that the new MFN values are not loaded. Hence it is recommended that the counter be programmed for value greater than 8.

It is required that the toggle counter values not change after the toggle enable is asserted. If toggle counter value changes, the functioning of the desense circuit may be unpredictable.

When any of the 3 MFN values i.e... MFN_PLUS, MFN_MINUS or MFN values change after the desense mode is enabled, they are buffered till the desense mode is disabled.

The desense status register, DESENSE_STAT, shows the status of the toggle_enable and the MFN value sent to the REFPLL.

Chapter 6
Chip Configuration and Memory Maps (MEMMAP)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	23 April 2002	Scott King	DSP and MCU Memory Map changes. Added new MCU peripherals. Moved PIG peripherals to IP bus.Added 2nd DMA
0.2	6 Mayl 2002	Scott King	DSP memory size clarifications. MCU RAM location moved. Minor changes to MCU peripherals,
0.3	17 May 2002	Scott King	Updates from 0.2 spec review: Fixed some LEM addresses Added DSP_DG_MD register Added AMARB Port3 Priority Reg
0.4	17 May 2002	Scott King	Modifed USB FSEN configuration Added UART2 configuration
0.5	19 July 2002	Mark Babcock Shannon Osgood	Updated per DDTS# DSPhl15004.
0.6	19 July 2002	Mark Babcock Shannon Osgood	Update per DDTS# DSPhl14898
0.7	22 July 2002	Mark Babcock Shannon Osgood	Updated Table 6-15 to include CKOH reference per PCS. Updated per DDTS# DSPhl14898.
0.8	5 August 2002	Mark Babcock Shannon Osgood	Updated per DDTS# DSPhl15087/15098/15088. Updated section 6.3.1.
0.9	7 Nov 2002	Mark Babcock	List Patch Page Register per DSPhl15119. List VODOC MIDI Registers per DSPhl15346. List EOTD counter register per DSPhl15350. List CCM Desense registers per DSPhl15120, DSPhl15635, DSPhl15369, DSPhl15316.
1.0	12 Nov 2002	Scott King, Shannon Osgood	Updated per DDTS# DSPhl15685/DSPhl15684. Added debug signal configuration section.

Revision History Table

Revision	Date	Author	Changes
1.1	3 Dec 2002	Mark Babcock	Update MCU Memory Map Tables to include Neptune LTE and Neptune ULS information. Update per DSPhl15801.
1.2	05/07/03		Updated for LTE specification release.

6.1 Introduction

This chapter describes the memory maps for the ARM7 TDMI-S MCU, 56600 DSP, memory configuration, chip configuration controls and registers used on Neptune. The Table 6-1 summarizes the DSP memory configurations. Table 6-2 outlines the ARM 7 memory configurations.

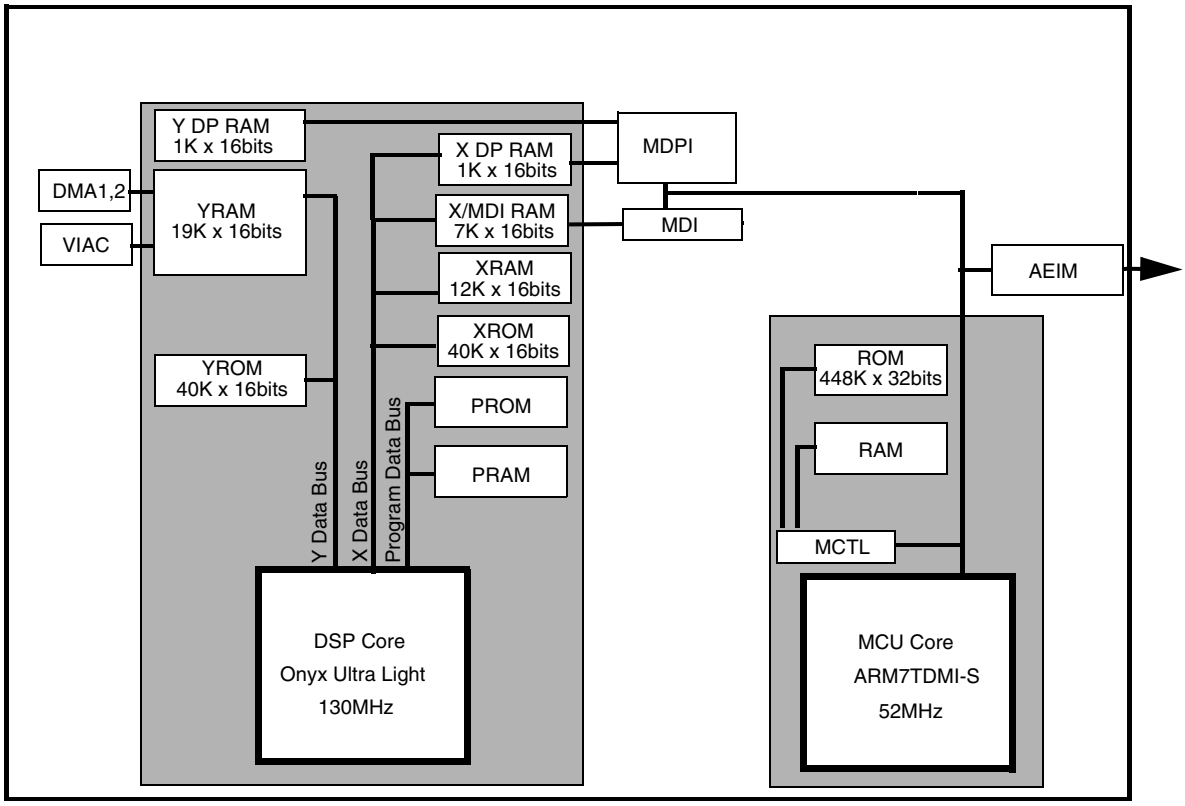


Figure 6-1. Block Diagram

Table 6-1. DSP Physical Memory Configurations

DSP Memory	LTE Configuration
Program ROM	127K[1] x 24 Bits
Program RAM	63.75K x 24 Bits
X-Data ROM	40K x 16 Bits
X-Total Data RAM	20K x 16 Bits
Y-Data ROM	40K x 16 Bits
Y-Total Data RAM	20K x 16 Bits

1. “K” means 1024.

Table 6-2. MCU Memory Configurations

MCU Memory	LTE Configuration
RAM	64K[1] x 32 Bits
ROM	448K x 32 Bits

1. “K” means 1024.

6.2 DSP Memory Space

The DSP memory space is partitoned into three (3) separate logical address spaces, Program, X-Data, and Y-Data memory space. The X and Y memory space is broken down into two separate spaces, one for internal memory (RAM and ROM) and peripheral I/O space. The Program space includes paging logic, that allows addressing greater than 64K words of Program memory. For more information on program paging refer to Section 20.3, “DSP Memory Paging.”

6.2.1 DSP Peripheral I/O Space

The DSP X-I/O and Y-I/O spaces for peripheral module access via the PMB bus are common across all Neptune devices. The following table provides the common DSP memory map for all Neptune derived devices. Table 6-3 lists the DSP X and Y I/O memory map. For a more detailed description of each register refer to the appropriate chapter or the DSP56600 Users Manual.

Table 6-3. DSP X - I/O and Y - I/O Memory Map

DSP Peripheral	X - I/O Address	Register Description	DSP Peripheral	Y - I/O Address	Register Description
PIC	X:$FFFF	Interrupt Priority Register (IPRC)	DSP	Y:$FFFF	DSP - Reserved
PIC	X:$FFFE	Interrupt Priority Register (IPRP)	DSP	Y:$FFFE	DSP - Reserved
DCLKG	X:$FFFD	Reserved	DSP	Y:$FFFD	DSP - Reserved
DCLKG	X:$FFFC	PLL Control Register (PCTL1)	DSP	Y:$FFFC	DSP - Reserved
OnCE	X:$FFFB	OnCE GDB Register (OGDB)	DSP	Y:$FFFB	DSP - Reserved
BIU	X:$FFFA	Bus Control Register (BCR)	DSP	Y:$FFFA	DSP - Reserved
BIU	X:$FFF9	ID Register (IDR)	DSP	Y:$FFF9	DSP - Reserved
PATCH	X:$FFF8	Patch Register 0	DSP	Y:$FFF8	DSP - Reserved
PATCH	X:$FFF7	Patch Register 1	DSP	Y:$FFF7	DSP - Reserved
PATCH	X:$FFF6	Patch Register 2	DSP	Y:$FFF6	DSP - Reserved
PATCH	X:$FFF5	Patch Register 3	DSP	Y:$FFF5	DSP - Reserved
BPMR	X:$FFF4	BPMRG Register (16-bit)	DSP	Y:$FFF4	DSP - Reserved
BPMR	X:$FFF3	BPMRL Register (16-bit)	DSP	Y:$FFF3	DSP - Reserved
BPMR	X:$FFF2	BPMRH Register (16-bit)	DSP	Y:$FFF2	DSP - Reserved
DSP	X:$FFF1	DSP Reserved	DSP	Y:$FFF1	DSP - Reserved
DSP	X:$FFF0	DSP Reserved	DSP	Y:$FFF0	DSP - Reserved

Table 6-3. DSP X - I/O and Y - I/O Memory Map

DSP Peripheral	X - I/O Address	Register Description	DSP Peripheral	Y - I/O Address	Register Description
DSP	X:$FFEF	DSP - Reserved	DEBUG	Y:$FFEF	XDATA Mask Register
DSP	X:$FFEE	DSP - Reserved	DEBUG	Y:$FFEE	XData Compare Register
DSP	X:$FFED	DSP - Reserved	DEBUG	Y:$FFED	XAddress High Range Register
DSP	X:$FFEC	DSP - Reserved	DEBUG	Y:$FFEC	XAddress Low Range Register
DSP	X:$FFEB	DSP - Reserved	DEBUG	Y:$FFEB	YDATA Mask Register
DSP	X:$FFEA	DSP - Reserved	DEBUG	Y:$FFEA	YData Compare Register
DSP	X:$FFE9	DSP - Reserved	DEBUG	Y:$FFE9	YAddress High Range Register
DSP	X:$FFE8	DSP - Reserved	DEBUG	Y:$FFE8	YAddress Low Range Register
DSP	X:$FFE7	DSP - Reserved	DEBUG	Y:$FFE7	Program Address Compare Reg
DSP	X:$FFE6	VOCOD - RXACQ COUNT	DEBUG	Y:$FFE6	Debug Control/Status Register
DSP	X:$FFE5	VOCOD - GR WORD MIDI	DEBUG	Y:$FFE5	Debug Page Configuration Register
DSP	X:$FFE4	VOCOD - VDIR MIDI	DEBUG	Y:$FFE4	Debug - Patch1 Register
VOCOD	X:$FFE3	VOCOD- VCTX	DEBUG	Y:$FFE3	Debug Patch0 Register
VOCOD	X:$FFE2	VOCOD- VCRX	DEBUG	Y:$FFE2	Debug - Reserved
VOCOD	X:$FFE1	VOCOD- VCSR	DEBUG	Y:$FFE1	Debug - Reserved
VOCOD	X:$FFE0	VOCOD- VCCR	DEBUG	Y:$FFE0	Debug - Reserved
DTIMER	X:$FFDF	DTimer Timer 0 CSR Register	GPIO	Y:$FFDF	GPIO - Reserved
DTIMER	X:$FFDE	DTimer Timer 0 Load Register	GPIO	Y:$FFDE	GPIO - Reserved
DTIMER	X:$FFDD	DTimer Timer 0 Compare Register	GPIO	Y:$FFDD	GPIO - Reserved
DTIMER	X:$FFDC	DTimer Timer 0 Count Register	GPIO	Y:$FFDC	GPIO - Reserved
DTIMER	X:$FFDB	DTimer Timer 1 CSR Register	GPIO	Y:$FFDB	GPIO - Reserved
DTIMER	X:$FFDA	DTimer Timer 1 Load Register	GPIO	Y:$FFDA	GPIO - Reserved
DTIMER	X:$FFD9	DTimer Timer 1 Compare Register	GPIO	Y:$FFD9	GPIO - Reserved
DTIMER	X:$FFD8	DTimer Timer 1 Count Register	GPIO	Y:$FFD8	GPIO - Reserved
DTIMER	X:$FFD7	DTimer Timer 2 CSR Register	GPIO	Y:$FFD7	GPIO - DSPBGREF
DTIMER	X:$FFD6	DTimer Timer 2 Load Register	GPIO	Y:$FFD6	GPIO - Reserved
DTIMER	X:$FFD5	DTimer Timer 2 Compare Register	GPIO	Y:$FFD5	GPIO Reserved
DTIMER	X:$FFD4	DTimer Timer 2 Count Register	GPIO	Y:$FFD4	DSP Port E Data Register
DTIMER	X:$FFD3	DTimer Prescaler Load Register	GPIO	Y:$FFD3	DSP Port D Data Register
DTIMER	X:$FFD2	DTimer Prescaler Count Register	GPIO	Y:$FFD2	DSP Port C Data Register
DTIMER	X:$FFD1	DTimer Reserved	GPIO	Y:$FFD1	DSP Port B Data Register
DTIMER	X:$FFD0	DTimer Reserved	GPIO	Y:$FFD0	DSP Port A Data Register
SAP	X:$FFCF	Serial Audio Port Control Register (PCRA)	USB	Y:$FFCF	USB - Reserved

Table 6-3. DSP X - I/O and Y - I/O Memory Map

DSP Peripheral	X - I/O Address	Register Description	DSP Peripheral	Y - I/O Address	Register Description
SAP	X:$FFCE	Serial Audio Port GPIO Direction Register (PRRA)	USB	Y:$FFCE	USB - Reserved
SAP	X:$FFCD	Serial Audio Port GPIO Data Register (PDRA)	USB	Y:$FFCD	USB - Reserved
SAP	X:$FFCC	Serial Audio Port TX Data Register (TXA)	USB	Y:$FFCC	USB - Reserved
SAP	X:$FFCB	Serial Audio Port Time Slot Register (TSRA)	USB	Y:$FFCB	USB - USB Endpoint 11 (Interrupt Transmit) Control Register
SAP	X:$FFCA	Serial Audio Port RX Data Register (RXA)	USB	Y:$FFCA	USB - DSP Buffer Register for Array Test Mode
SAP	X:$FFC9	Serial Audio Port Status Register (SSISRA)	USB	Y:$FFC9	USB Buffer Register
SAP	X:$FFC8	Serial Audio Port Control Register C (CRCA)	USB	Y:$FFC8	USB Buffer Pointer Register
SAP	X:$FFC7	Serial Audio Port Control Register B(CRBA)	USB	Y:$FFC7	Endpoint 9 (Interrupt TX) Control Register
SAP	X:$FFC6	Serial Audio Port Control Register A(CRAA)	USB	Y:$FFC6	Endpoint 8 (Interrupt RX) Control Register
SAP	X:$FFC5	Serial Audio Port Timer Count Load Register (TCLR)	USB	Y:$FFC5	Endpoint 7 (Interrupt RX) Control Register
SAP	X:$FFC4	Serial Audio Port Timer Count Register (TCRA)	USB	Y:$FFC4	Endpoint 6 (Isochronous TX) Control Register
SAP	X:$FFC3	Serial Audio Port BRM Constant A Register (BCARA)	USB	Y:$FFC3	Endpoint 5 (Isochronous RX) Control Register
SAP	X:$FFC2	Serial Audio Port BRM Constant B Register (BCBRA)	USB	Y:$FFC2	Start of Frame Register
SAP	X:$FFC1	SAP Reserved	USB	Y:$FFC1	USB Interrupt Register
SAP	X:$FFC0	SAP Reserved	USB	Y:$FFC0	USB Interrupt Enable Register
VIAC	X:$FFBF	VIAC Reserved	LEM	Y:$FFBF	LEM - Reserved
VIAC	X:$FFBE	VIAC DMA Output Channel - Current Address (VDOCA)	LEM	Y:$FFBE	LEM - Reserved
VIAC	X:$FFBD	VIAC DMA Output Channel - Base Address (VDOBA)	LEM	Y:$FFBD	LEM - Reserved
VIAC	X:$FFBC	VIAC DMA Input Channel - Current Address (VDICA)	LEM	Y:$FFBC	LEM - Reserved
VIAC	X:$FFBB	VIAC DMA Input Channel - Base Address (VDIBA)	LEM	Y:$FFBB	LEM - Reserved
VIAC	X:$FFBA	VIAC Path Metric Access Register/FIFO B (VPMAR B)	LEM	Y:$FFBA	LEM - Reserved
VIAC	X:$FFB9	VIAC Path Metric Access Register/FIFO A (VPMAR A)	LEM	Y:$FFB9	LEM - Reserved
VIAC	X:$FFB8	VIAC WED Address Register (VWADRR)	LEM	Y:$FFB8	LEM - Reserved

Table 6-3. DSP X - I/O and Y - I/O Memory Map

DSP Peripheral	X - I/O Address	Register Description	DSP Peripheral	Y - I/O Address	Register Description
VIAC	X:$FFB7	VIAC WED Data Register (VWEDR)	LEM	Y:$FFB7	LEM - Reserved
VIAC	X:$FFB6	VIAC Counter Register (VTCR)	LEM	Y:$FFB6	LEM - Reserved
VIAC	X:$FFB5	VIAC Mode Register (VMR)	LEM	Y:$FFB5	LEM - Reserved
VIAC	X:$FFB4	VIAC CSR Register (VCSR)	LEM	Y:$FFB4	LEM - Reserved
VIAC	X:$FFB3	VIAC Output Data Register (VODR)	LEM	Y:$FFB3	LEM - Reserved
VIAC	X:$FFB2	VIAC Polynomial Tap Register (VPTR)	LEM	Y:$FFB2	LEM - Reserved
VIAC	X:$FFB1	VIAC Branch Metric Register/FIFO (VBMR)	LEM	Y:$FFB1	LEM Data Register 15
VIAC	X:$FFB0	VIAC Input Data Register	LEM	Y:$FFB0	LEM Data Register 14
BBP	X:$FFAF	BBP Control Register (PCRB)	LEM	Y:$FFAF	LEM Data Register 13
BBP	X:$FFAE	BBPGPIO Direction Register (PRRB)	LEM	Y:$FFAE	LEM Data Register 12
BBP	X:$FFAD	BBP GPIO Data Register (PDRB)	LEM	Y:$FFAD	LEM Data Register 11
BBP	X:$FFAC	BBP TX Data Register (TXB)	LEM	Y:$FFAC	LEM Data Register 10
BBP	X:$FFAB	BBP Time Slot Register (TSRB)	LEM	Y:$FFAB	LEM Data Register 9
BBP	X:$FFAA	BBP RX Data Register (RXB)	LEM	Y:$FFAA	LEM Data Register 8
BBP	X:$FFA9	BBP Status Register (SSISRB)	LEM	Y:$FFA9	LEM Data Register 7
BBP	X:$FFA8	BBP Control Register C (CRCB)	LEM	Y:$FFA8	LEM Data Register 6
BBP	X:$FFA7	BBP Control Register B (CRBB)	LEM	Y:$FFA7	LEM Data Register 5
BBP	X:$FFA6	BBP Control Register A (CRAB)	LEM	Y:$FFA6	LEM Data Register 4
BBP	X:$FFA5	BBP TX Counter Load Register (TCRB)	LEM	Y:$FFA5	LEM Data Register 3
BBP	X:$FFA4	BBP RX Counter Load Register (RCRB)	LEM	Y:$FFA4	LEM Data Register 2
BBP	X:$FFA3	BBP Reserved	LEM	Y:$FFA3	LEM Data Register 1
BBP	X:$FFA2	BBP Reserved	LEM	Y:$FFA2	LEM Data Register 0
BBP	X:$FFA1	BBP Reserved	LEM	Y:$FFA1	LEM Status Register
BBP	X:$FFA0	BBP Reserved	LEM	Y:$FFA0	LEM Control Register
DMA1	X:$FF9F	DMA1 Control Register A	DMA2	Y:$FF9F	DMA2 Control Register A
DMA1	X:$FF9E	DMA1 Base Address Register	DMA2	Y:$FF9E	DMA2 Base Address Register
DMA1	X:$FF9D	DMA1 Address Counter Register	DMA2	Y:$FF9D	DMA2 Address Counter Register
DMA1	X:$FF9C	DMA1 Word Count Register	DMA2	Y:$FF9C	DMA2 Word Count Register
DMA1	X:$FF9B	DMA1 Buffer Size Register	DMA2	Y:$FF9B	DMA2 Buffer Size Register
DMA1	X:$FF9A	DMA1 Time Out Register	DMA2	Y:$FF9A	DMA2 Time Out Register
DMA1	X:$FF99	DMA1 Control Register B	DMA2	Y:$FF99	DMA2 Control Register B

Table 6-3. DSP X - I/O and Y - I/O Memory Map

DSP Peripheral	X - I/O Address	Register Description	DSP Peripheral	Y - I/O Address	Register Description
DMA1	X:$FF98	DMA1 - Reserved	DMA2	Y:$FF98	DMA2 - Reserved
DMA1	X:$FF97	DMA1 - Reserved	DMA2	Y:$FF97	DMA2 - Reserved
DMA1	X:$FF96	DMA1 - Reserved	DMA2	Y:$FF96	DMA2 - Reserved
DMA1	X:$FF95	DMA1 - Reserved	DMA2	Y:$FF95	DMA2 - Reserved
DMA1	X:$FF94	DMA1 - Reserved	DMA2	Y:$FF94	DMA2 - Reserved
DMA1	X:$FF93	DMA1 - Reserved	DMA2	Y:$FF93	DMA2 - Reserved
DMA1	X:$FF92	DMA1 - Reserved	DMA2	Y:$FF92	DMA2 - Reserved
DMA1	X:$FF91	DMA1 - Reserved	DMA2	Y:$FF91	DMA2 - Reserved
DMA1	X:$FF90	DMA1 - Reserved	DMA2	Y:$FF90	DMA2 - Reserved
MDI	X:$FF8F	DSP RX Register 0 (DRR0)	DSP	Y:$FF8F	DSP - Reserved
MDI	X:$FF8E	DSP RX Register 1 (DRR1)	DSP	Y:$FF8E	DSP - Reserved
MDI	X:$FF8D	DSP TX Register 0 (DTR0)	DSP	Y:$FF8D	DSP - Reserved
MDI	X:$FF8C	DSP TX Register 1 (DTR1)	DSP	Y:$FF8C	DSP - Reserved
MDI	X:$FF8B	DSP Status Register (DSR)	L1T	Y:$FF8B	L1- Timer_DSP_EG_MD
MDI	X:$FF8A	DSP Control Register (DCR)	L1T	Y:$FF8A	L1- Timer_DTX
MDI	X:$FF89	MDI Reserved	L1T	Y:$FF89	L1- Timer_QBCO
MDI	X:$FF88	MDI Reserved	L1T	Y:$FF88	L1- Timer_L1T_MABC
MDI	X:$FF87	MDI Reserved	L1T	Y:$FF87	L1- Timer_L1T_RQBC
MDI	X:$FF86	MDI Reserved	DSP	Y:$FF86	DSP - Reserved
MDI	X:$FF85	MDI Reserved	DSP	Y:$FF85	DSP - Reserved
MDI	X:$FF84	MDI Reserved	DSP	Y:$FF84	DSP - Reserved
MDI	X:$FF83	MDI Reserved	DSP	Y:$FF83	DSP - Reserved
MDI	X:$FF82	MDI Reserved	DSP	Y:$FF82	DSP - Reserved
MDI	X:$FF81	MDI Reserved	DSP	Y:$FF81	DSP - Reserved
MDI	X:$FF80	MDI Reserved	DSP	Y:$FF80	DSP - Reserved

6.2.2 DSP56600 Patch Jump Targets

The following describes the various jump targets for each of the Patch Address Registers. The user should download the correct piece of patch code to the target location in order for the patch program to execute correctly. For a more detail description, refer to the Program Patch Logic chapter in the DSP56600 Family Manual.

Table 6-4. Patch Jump Targets

Patch Register	Patch Jump Target Address
PAR0	$0000
PAR1	$0008
PAR2	$0010
PAR3	$0018

6.2.3 DSP Memory Space

6.2.3.1 DSP Program Memory Space

The DSP Program memory maps are shown in the figures below.

The DSP 56600 Core can only address 64Kx24 of program space. The current Neptune application exceeds this limit. This facilitates the need for the implementation of a program ram paging method. The address range from $0000 to $7FFF remains constant regardless of which page is selected. Addresses $8000-$EFFF are divided into 4 separately addressable ranges, $8000 to $9FFF, $A000 to $BFFF, $C000 to $DFFF, and $E000 to $FEFF ($FF00 and higher addresses are reserved in all pages). Each of these address ranges has up to 8 pages of memory to choose from by utilizing the Page Configuration Register in the DSP Debug Module. Upon reset, the PCR register is cleared, and the DSP will begin executing code beginning at the reset vector, P:$8000 in page 0. Refer to Section 20.3, “DSP Memory Paging,” for a more detailed description on paging.

NOTE:

Due to DSP pipeline constraints, a delay of at least three instruction cycles after writing the PCR is required before the paged memory can be accessed. This constraint can be satisfied by placing three NOP instructions after a write to the PCR. Refer to Chapter 20, “DSP Debug Module (PDDM),” for additional restrictions.

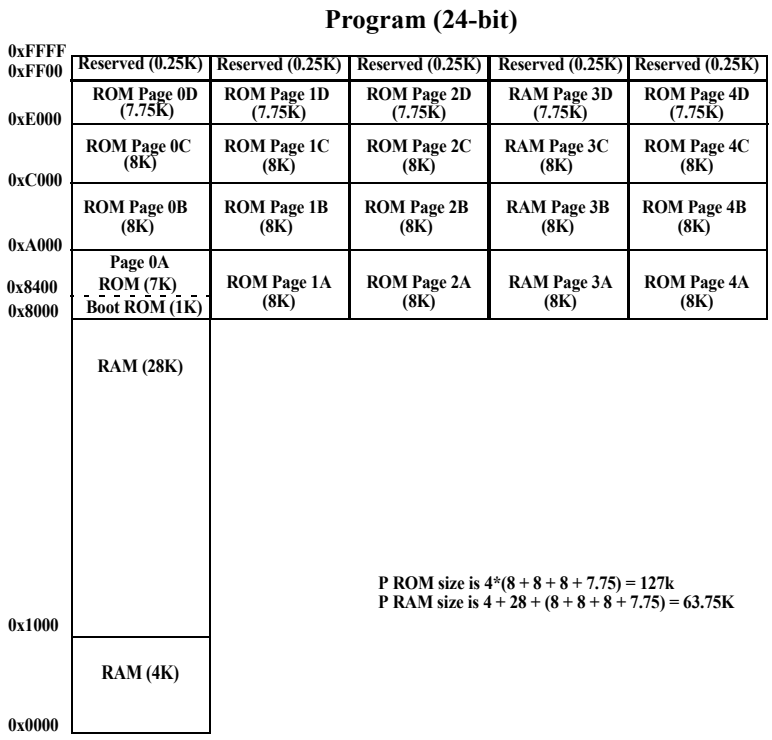


Figure 6-2. DSP 56600 P Memory Map - Neptune LTE

6.2.3.2 DSP X and Y Memory Spaces

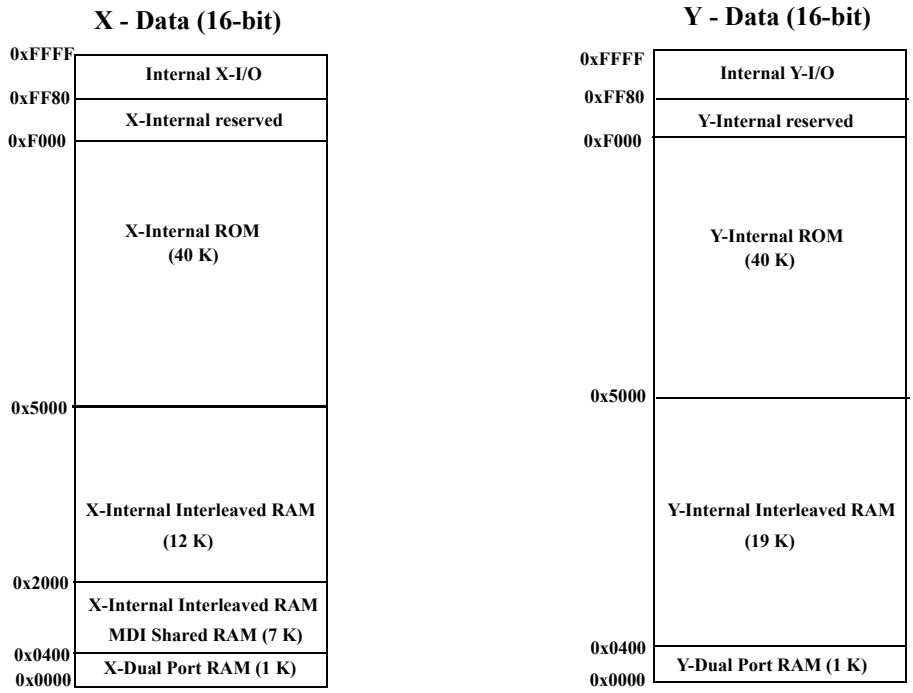
The DSP X and Y memory maps are shown below.

The DSP X and Y dual-port RAM blocks are accessible by the MCU through the MCU's MDPI peripheral. This memory provides a faster througput than the MDI shared memory, but does have some access restrictions. See Chapter 8, "DSP Memory (DSPMEM)," for more information on the restrictions.

The DSP Y interleaved memory is address interleaved over the entire Y Data RAM space except for the Dual Port section at the bottom of the Y memory map. See Section 6.2.6, "DSP56600 DMA Interface," later in this chapter for more information on how this memory in connected to the DMA and VIAC peripherals.

The DSP X interleaved memory is address interleaved over the entire X Data RAM space except for the Dual Port section at the bottom of the X memory map. 7Kx16 of the X interleaved ram is used as MDI shared memory, and is accessible by the MCU through the MDI peripheral.

For more information on the DSP memories, see Chapter 8, "DSP Memory (DSPMEM)."



MDI may access the first 7k X-Internal Interleaved RAM but cannot access X-Dual Port RAM.
All of X and Y Data RAM is accessible to DSP (0000h to 5000h).
MCU must access X and Y Dual port access via dedicated port in MCU map.

Figure 6-3. DSP 56600 X and Y Memory Map - Neptune LTE

6.2.4 Detailed Register Descriptions for DSP Core

The following figures and associated text give detailed descriptions of the various registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

The DSP Operating Mode Register (OMR) for Neptune and its derivatives is described in the following figure. This register physically is located as part of the DSP Core Program Control Unit and is addressed by specialized instructions.

OMR DSP 56600 Operating Mode Register

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	ATE		PTE	SEN	WRP	EOV	EUN	XYS		SD	PCD	EBD			MB	MA
TYPE	rw	r	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	r	r	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-5. DSP 56600 OMR Description (Continued)

Name	Description	Settings
ATE Bit 15	The Address Trace Enable (ATE) bit is used for debugging purposes where internal activity should be traceable via a logic analyzer. When this bit is set, the Address Trace mode is enabled and the external address bus reflects the internal program address bus for every program fetch. When this bit is cleared, normal operation resumes and the external address bus is activated only when the program address is in the external address space. The ATE bit is cleared by hardware reset. **Note:** This function also requires the GPIO ATE function be enabled.	0 = Address trace mode disabled 1 = Address trace mode enabled
Bit 14	Reserved - read as zero	N/A
PTE Bit 13	The Paging bits Trace Enable (PTE) control bit is used to enable or disable the generation of the paging bits in the debug module. When the PTE is set, the paging bits are generated in the DSP debug module. The paging bits are reflected on the external pins if the ATE bit in the OMR is set. The PTE bit is cleared by hardware reset.	0 = paging bits generation is disabled. 1= paging bits generation is enabled.
SEN Bit 12	The Extended Stack Enable (EN) control bit is used to enable or to disable the stack extension in the data memory. When the EN bit is set, the extension is enabled. When the EN bit is cleared, the extension is disabled. The EN bit is cleared by hardware reset.	0 = Stack extension disabled 1 = Stack extension enabled
WRP Bit 11	The Extended Stack Wrap (WR) flag bit is set when it is first recognized that a copy from the on-chip hardware stack to the stack extension memory is needed. This flag can be used during the debugging phase of the software as means of evaluating and increasing the speed of the software implemented algorithms. The WR bit is a “sticky bit” (i.e., the only way to clear this bit is by hardware reset or by an explicit MOVE operation to the OMR). The WR bit is cleared by hardware reset.	0 = Copy from the on-chip hardware stack to the stack extension memory is not needed. 1= Copy from the on-chip hardware stack to the stack extension memory is needed.

Table 6-5. DSP 56600 OMR Description (Continued)

Name	Description	Settings
EOV Bit 10	The Extended Stack Overflow (EOV) flag bit is set when a stack overflow occurs in the Stack Extended mode. The Extended Stack Overflow is generated when SP equals SZ and an additional push operation is requested while the Extended mode is enabled by the SEN bit. The EOV bit is a “sticky bit”(i.e., the only way to clear this bit is by hardware reset or by an explicit MOVE operation to the OMR). The transition of the extended stack overflow flag from 0 to 1 causes an IPL 3 Stack Error interrupt. The EOV bit is cleared by hardware reset. **Note:** There is an errata on this function.	0 = Stack overflow has not occurred 1 = Stack overflow occurred
EUN Bit 9	The Extended Stack Underflow (EUN) flag bit is set when a stack underflow occurs in the Stack Extended mode. The Extended Stack Underflow is generated when the SP equals 0 and an additional pull operation is requested while the Extended mode is enabled by the SEN bit. The EUN bit is a “sticky bit” (i.e., the only way to clear this bit is by hardware reset or by an explicit MOVE operation to the OMR). The transition of the EUN bit from 0 to 1 causes an Interrupt Priority Level (IPL) Level 3 Stack Error interrupt. The EUN bit is cleared by hardware reset.	0 = Stack underflow has not occurred 1 = Stack underflow occurred
XYS Bit 8	The XY Select (XY) control bit for the stack extension determines whether the extension is mapped onto the X memory space or onto the Y memory space. When the XY bit is set, the stack extension is mapped to the Y memory space. When the XY bit is cleared, the stack extension is mapped onto the X memory space. The XY bit is cleared by hardware reset.	0 = Stack extension mapped to X memory 1 = Stack extension mapped to Y memory
Bit 7	**Reserved**	N/A
SD Bit 6	The Stop Delay (SD) control bit enables providing a long or short delay when exiting the Stop state. The STOP instruction causes the DSP56600 core to indefinitely suspend processing in the middle of the STOP instruction. When the SD bit is set, a short delay of 16 clock cycles is inserted when exiting the Stop state before continuing the instruction cycle. When the SD bit is cleared, a long delay of 128 K clock cycles is inserted before continuing the instruction cycle. The long delay allows a clock stabilization period for the internal clock to begin oscillating and to stabilize. When a stable external clock is used, the shorter delay allows faster start-up of the DSP56600 core.	0 = Delay of 128000 clock cycles inserted 1 = Delay of 16 clock cycles inserted.

Table 6-5. DSP 56600 OMR Description (Continued)

Name	Description	Settings
PCD Bit 5	The PC Relative Disable (PCD) control bit is used when PC-relative instructions (Bcc, BRA, LRA, DO, DO FOREVER, BSR, BScc, BRSET, BRCLR, BSSET, or BSCLR) are not in use, in order to reduce the power consumption when PC-relative instructions are not needed. When the PCD bit is set, the use of any PC-relative instruction causes undetermined results. When the PCD bit is cleared, PC-relative instructions operate correctly. In addition, when the PCD bit is set and then cleared, the use of PC-relative instructions is allowed only after seven instructions are executed. (This allows the instruction pipeline to clear.) The PCD bit is cleared on hardware reset.	0 = PC relative instructions enabled 1 = PC relative instructions disabled
EBD Bit 4	The External Bus Disable (EBD) control bit is used to disable the external bus controller, in order to reduce the power consumption when external memories are not used. When the EBD bit is set, the external bus controller is disabled and external memory cannot be accessed. When the EBD bit is cleared, the external bus controller is enabled and external access to memory can be performed. The EBD bit is cleared on hardware reset.	0 = External bus controller enabled 1 = External bus controller disabled
Bits 3-2	**Reserved** — read as zero	N/A
MB Bit 1	The Chip Operating Mode bits (MB, and MA) indicate the operating mode of the DSP56600. However, the Neptune DSP has only has one memory map (given in Figure 6-2), so the MA and MB pins do not affect the operating mode for Neptune devices. On reset, the DSP will begin executing the DSP bootloader code.	
MA Bit 0	The Chip Operating Mode bits (MB, and MA) indicate the operating mode of the DSP56600. However, the Neptune DSP has only has one memory map (given in Figure 6-2), so the MA and MB pins do not affect the operating mode for Neptune devices. On reset, the DSP will begin executing the DSP bootloader code.	

The DSP Interrupt Priority Register (IPRP) for Neptune and its derivatives is described in Table 6-6.

IPRP DSP 56600 Interrupt Priority Register P **X:$FFFE**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	EVCP L1	EVCP L0	DMAP L1	DMAP L0	MDIP L1	MDIP L0	TIMP L1	TIMP L0	SDDP L1	SDDP L0	BPAP L1	BPAP L0	USBP L1	USBP L0	MDC PL1	MDC PL0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-6. DSP 56600 IPRP Description

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>EVCPL[1:0]
Bits 15-14</td><td>Encryption Priority Level
VIAC Priority Level

IPL - Interrupt Priority Level
Lowest Number Indicates Lowest Priority</td><td>
<table>
<tr><th>EVCPL [1:0]</th><th>Interrupts Enabled</th><th>Interrupt IPL</th></tr>
<tr><td>00</td><td>No</td><td>N/A</td></tr>
<tr><td>01</td><td>Yes</td><td>0</td></tr>
<tr><td>10</td><td>Yes</td><td>1</td></tr>
<tr><td>11</td><td>Yes</td><td>2</td></tr>
</table>
</td></tr>
<tr><td>DMAPL[1:0]
Bits 13-12</td><td>DMA1 and DMA2 Priority Level

IPL - Interrupt Priority Level
Lowest Number Indicates Lowest Priority</td><td>
<table>
<tr><th>DMAPL [1:0]</th><th>Interrupts Enabled</th><th>Interrupt IPL</th></tr>
<tr><td>00</td><td>No</td><td>N/A</td></tr>
<tr><td>01</td><td>Yes</td><td>0</td></tr>
<tr><td>10</td><td>Yes</td><td>1</td></tr>
<tr><td>11</td><td>Yes</td><td>2</td></tr>
</table>
</td></tr>
<tr><td>MDIPL[1:0]
Bits 11-10</td><td>MDI Priority Level

IPL - Interrupt Priority Level
Lowest Number Indicates Lowest Priority</td><td>
<table>
<tr><th>MDIPL [1:0]</th><th>Interrupts Enabled</th><th>Interrupt IPL</th></tr>
<tr><td>00</td><td>No</td><td>N/A</td></tr>
<tr><td>01</td><td>Yes</td><td>0</td></tr>
<tr><td>10</td><td>Yes</td><td>1</td></tr>
<tr><td>11</td><td>Yes</td><td>2</td></tr>
</table>
</td></tr>
</table>

Table 6-6. DSP 56600 IPRP Description

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>TIMPL[1:0]
Bits 9-8</td><td>L1Timer Priority Level
DTimer Priority Level

IPL - Interrupt Priority Level
Lowest Number Indicates Lowest Priority</td><td>
<table>
<tr><th>TIMPL [1:0]</th><th>Interrupts Enabled</th><th>Interrupt IPL</th></tr>
<tr><td>00</td><td>No</td><td>N/A</td></tr>
<tr><td>01</td><td>Yes</td><td>0</td></tr>
<tr><td>10</td><td>Yes</td><td>1</td></tr>
<tr><td>11</td><td>Yes</td><td>2</td></tr>
</table>
</td></tr>
<tr><td>SDDPL[1:0]
Bits 7-6</td><td>Serial Audio Port Priority Level
VOCOD Priority Level
DSP Debug Priority Level

IPL - Interrupt Priority Level
Lowest Number Indicates Lowest Priority</td><td>
<table>
<tr><th>SDDPL [1:0]</th><th>Interrupts Enabled</th><th>Interrupt IPL</th></tr>
<tr><td>00</td><td>No</td><td>N/A</td></tr>
<tr><td>01</td><td>Yes</td><td>0</td></tr>
<tr><td>10</td><td>Yes</td><td>1</td></tr>
<tr><td>11</td><td>Yes</td><td>2</td></tr>
</table>
</td></tr>
<tr><td>BPAPL[1:0]
Bits 5-4</td><td>Base Band Port Priority Level

IPL - Interrupt Priority Level
Lowest Number Indicates Lowest Priority</td><td>
<table>
<tr><th>BPAPL [1:0]</th><th>Interrupts Enabled</th><th>Interrupt IPL</th></tr>
<tr><td>00</td><td>No</td><td>N/A</td></tr>
<tr><td>01</td><td>Yes</td><td>0</td></tr>
<tr><td>10</td><td>Yes</td><td>1</td></tr>
<tr><td>11</td><td>Yes</td><td>2</td></tr>
</table>
</td></tr>
<tr><td>USBPL[1:0]
Bits 3-2</td><td>Universal Serial Bus Level

IPL - Interrupt Priority Level
Lowest Number Indicates Lowest Priority</td><td>
<table>
<tr><th>USBPL [1:0]</th><th>Interrupts Enabled</th><th>Interrupt IPL</th></tr>
<tr><td>00</td><td>No</td><td>N/A</td></tr>
<tr><td>01</td><td>Yes</td><td>0</td></tr>
<tr><td>10</td><td>Yes</td><td>1</td></tr>
<tr><td>11</td><td>Yes</td><td>2</td></tr>
</table>
</td></tr>
</table>

Table 6-6. DSP 56600 IPRP Description

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>MDCPL[1:0]
Bits 1-0</td><td>MDI Command Priority Level

IPL - Interrupt Priority Level
Lowest Number Indicates Lowest Priority</td><td>
<table>
<tr><th>MDCPL [1:0]</th><th>Interrupts Enabled</th><th>Interrupt IPL</th></tr>
<tr><td>00</td><td>No</td><td>N/A</td></tr>
<tr><td>01</td><td>Yes</td><td>0</td></tr>
<tr><td>10</td><td>Yes</td><td>1</td></tr>
<tr><td>11</td><td>Yes</td><td>2</td></tr>
</table>
</td></tr>
</table>

The DSP Interrupt Priority Register (IPRC) for Neptune and its derivatives is described in the following table.

IPRC DSP 56600 Interrupt Priority Register C **X:$FFFF**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
					IDTM	IDPL1	IDPL0	ICTM	ICPL1	ICPL0	IBTM	IBPL1	IBPL0	IATM	IAPL1	IAPL0
TYPE	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-7. DSP 56600 IPRC Description

<table>
<tr><th>Name</th><th>Description</th><th>Setting</th></tr>
<tr><td>Bit 15-12</td><td>RESERVED</td><td></td></tr>
<tr><td>IDTM
Bit 11</td><td>IRQD Trigger Mode</td><td>0 - Level
1 - Edge</td></tr>
<tr><td>IDPL[1:0]
Bits 10-9</td><td>IRQD Priority Level

IPL - Interrupt Priority Level
Lowest Number Indicates Lowest Priority</td><td>
<table>
<tr><th>IDPL [1:0]</th><th>Interrupts Enabled</th><th>Interrupt IPL</th></tr>
<tr><td>00</td><td>No</td><td>N/A</td></tr>
<tr><td>01</td><td>Yes</td><td>0</td></tr>
<tr><td>10</td><td>Yes</td><td>1</td></tr>
<tr><td>11</td><td>Yes</td><td>2</td></tr>
</table>
</td></tr>
<tr><td>ICTM
Bit 8</td><td>IRQC Trigger Mode</td><td>0 - Level
1 - Edge</td></tr>
<tr><td>ICPL[1:0]
Bits 7-6</td><td>IRQC Priority Level

IPL - Interrupt Priority Level
Lowest Number Indicates Lowest Priority</td><td>
<table>
<tr><th>ICPL [1:0]</th><th>Interrupts Enabled</th><th>Interrupt IPL</th></tr>
<tr><td>00</td><td>No</td><td>N/A</td></tr>
<tr><td>01</td><td>Yes</td><td>0</td></tr>
<tr><td>10</td><td>Yes</td><td>1</td></tr>
<tr><td>11</td><td>Yes</td><td>2</td></tr>
</table>
</td></tr>
<tr><td>IBTM
Bit 5</td><td>IRQB Trigger Mode</td><td>0 - Level
1 - Edge</td></tr>
</table>

Table 6-7. DSP 56600 IPRC Description

<table>
<tr><th>Name</th><th>Description</th><th>Setting</th></tr>
<tr><td>IBPL[1:0]
Bits 4-3</td><td>IRQB Priority Level

IPL - Interrupt Priority Level
Lowest Number Indicates Lowest Priority</td><td>
<table>
<tr><th>IBPL [1:0]</th><th>Interrupts Enabled</th><th>Interrupt IPL</th></tr>
<tr><td>00</td><td>No</td><td>N/A</td></tr>
<tr><td>01</td><td>Yes</td><td>0</td></tr>
<tr><td>10</td><td>Yes</td><td>1</td></tr>
<tr><td>11</td><td>Yes</td><td>2</td></tr>
</table>
</td></tr>
<tr><td>IATM
Bit 2</td><td>IRQA Trigger Mode</td><td>0 - Level
1 - Edge</td></tr>
<tr><td>IAPL[1:0]
Bits 1-0</td><td>IRQA Priority Level

IPL - Interrupt Priority Level
Lowest Number Indicates Lowest Priority</td><td>
<table>
<tr><th>IAPL [1:0]</th><th>Interrupts Enabled</th><th>Interrupt IPL</th></tr>
<tr><td>00</td><td>No</td><td>N/A</td></tr>
<tr><td>01</td><td>Yes</td><td>0</td></tr>
<tr><td>10</td><td>Yes</td><td>1</td></tr>
<tr><td>11</td><td>Yes</td><td>2</td></tr>
</table>
</td></tr>
</table>

The Device Identification Register (IDR) is a 16-bit read-only register that is factory programmed to identify the different DSP56600 core-based family members. This register specifies the product revision, product vendor/technology, and the silicon revision number.

IDR DSP 56600 Device Identification Register **Addr X:$FFF9**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PR[4:0]					PVT[2:0]			SR[7:0]							
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	Device Dependent															

Table 6-8. IDR Description

Name	Description	Settings
PR[4:0] Bits 15-11	**Product Revision -** Specifies the Neptune product type.	**PR[4:0]** / **Neptune Product** 0XXXX – Reserved 10000 – Neptune LT 10001 – Neptune LTS 10010 – Neptune LTE 10011 – Neptune ULS All Other – Reserved
PVT[2:0] Bits 10-8	Product Vendor / Technology - Specifies the Neptune vendor and silicon technology	**PVT[2:0]** / **Neptune Vendor / Technology** 010 – SPS Hip7 011 – SPS Hip8 All Other – Reserved
SR[7:0] Bits 7-0	**SIlicon Revision** - Increments with each silicon change.	0000_0000 - Initial Revision 0000_0001 - Pass 2

The JTAG Identification Register is a 32-bit read-only via JTAG, factory programmed used to distinguish the component on the board. For more detailed description refer to the DSP56600 Family Manual.

JIDR DSP 56600 JTAG Identification Register

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	Version Number				Customer Part Number											
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	1	1	1	device dependent					

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Customer Part Number				Manufacturer Identification											
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	device dependent				0	0	0	0	0	0	0	1	1	1	0	1

Table 6-9. JIDR Description

Name	Description	Settings
Version Number Bits 31-28	Specifies the Neptune version	N/A
Customer Part Number Bits 27-12	Specifies the Design center (Bits 27:22) and assigned device number (Bits 21:12)	**Bits [27:22] / Design Center** 000111 — AWDC 001000 — India Design Center Other — Reserved **Bits [21:12] / Neptune Device** 11 0000 0000 — Neptune LT 11 0000 0001 — Neptune LTS 11 0000 0010 — Neptune LTE 11 0000 0011 — Neptune ULS
Manufacturer Identification Bits 11-1	Specifies the manufacturer	Motorola = 00000001110
Bit 0	Reserved	Always reads 1

6.2.5 DSP Interrupt Controller

The DSP Interrupt Controller is part of the DSP56600 Core. Modifications include adding priority level bits for the new DSP peripherals. The DSP interrupt sources for the Neptune IC are shown in the table below, along with their corresponding VBA (Vector Base Address) and interrupt type (Note: Priority 3 is Highest).

Table 6-10. DSP Interrupt Sources

Interrupt Starting Address	Priority	Interrupt Source	Edge/Level
VBA:$00	3	Reserved	Level
VBA:$02	3	Stack Error	Level
VBA:$04	3	Illegal Instruction	Level
VBA:$06	3	Debug Request Interrupt	Level
VBA:$08	3	Trap	Level
VBA:$0A	3	Reserved	Level
VBA:$0C	3	Reserved	Level
VBA:$0E	3	Reserved	Level
VBA:$10	0-2	IRQA - GPIO or MDI or L1Timer	Edge/Level thru DSP Interrupt Controller
VBA:$12	0-2	IRQB - GPIO	Edge/Level thru DSP Interrupt Controller
VBA:$14	0-2	IRQC - MDI	Edge/Level thru DSP Interrupt Controller
VBA:$16	0-2	IRQD - L1 Timer	Edge/Level thru DSP Interrupt Controller
VBA:$18	0-2	Reserved	Level
VBA:$1A	0-2	Reserved	Level
VBA:$1C	0-2	VIAC Processing Complete	Level
VBA:$1E	0-2	VIAC Error	Level
VBA:$20	0-2	USB CVR0 - Endpoint 5	Level
VBA:$22	0-2	USB CVR1 - Endpoint 6	Level
VBA:$24	0-2	USB CVR2 - Endpoint 7	Level
VBA:$26	0-2	USB CVR3 - Endpoint 8	Level
VBA:$28	0-2	USB CVR4 - Endpoint 9	Level
VBA:$2A	0-2	USB CVR5 - Start Of Frame	Level
VBA:$2C	0-2	USB CVR6 - Suspend/Resume	Level
VBA:$2E	0-2	USB CVR7 - Reset	Level
VBA:$30	0-2	LEM Encryption Complete	Level
VBA:$32	0-2	USB - Endpoint 11	Level
VBA:$34	0-2	VOCOD Receive & Transmit Data	Level
VBA:$36	0-2	VOCOD Receive Data with Exception	Level

Table 6-10. DSP Interrupt Sources

Interrupt Starting Address	Priority	Interrupt Source	Edge/Level
VBA:$38	0-2	VOCOD Transmit Data with Exception	Level
VBA:$3A	0-2	RESERVED	Level
VBA:$3C	0-2	RESERVED	Level
VBA:$3E	0-2	RESERVED	Level
VBA:$40	0-2	SAP Receive Data	Level
VBA:$42	0-2	SAP Receive Data with Exception	Level
VBA:$44	0-2	SAP Receive last slot	Level
VBA:$46	0-2	SAP Transmit Data	Level
VBA:$48	0-2	SAP Transmit Data with Exception	Level
VBA:$4A	0-2	SAP Transmit last slot	Level
VBA:$4C	0-2	SAP Timer counter rollover	Level
VBA:$4E	0-2	Reserved	Level
VBA:$50	0-2	BBP Receive Data	Level
VBA:$52	0-2	BBP Receive Data with Exception	Level
VBA:$54	0-2	BBP Receive last slot	Level
VBA:$56	0-2	BBP Receive Frame Counter	Level
VBA:$58	0-2	BBP transmit Data	Level
VBA:$5A	0-2	BBP Transmit Data with Exception	Level
VBA:$5C	0-2	BBP Transmit last slot	Level
VBA:$5E	0-2	BBP Transmit Frame counter	Level
VBA:$60	0-2/3	MDI MCU default command / MCU NMI	Level
VBA:$62	0-2	MDI Receive 0	Level
VBA:$64	0-2	MDI Receive 1	Level
VBA:$66	0-2	MDI Transmit 0	Level
VBA:$68	0-2	MDI Transmit 1	Level
VBA:$6A	0-2	Reserved	
VBA:$6C	0-2	Reserved	
VBA:$6E	0-2	Reserved	Level
VBA:$70	0-2	L1 Timer RX	Edge
VBA:$72	0-2	L1 Timer TX	Edge
VBA:$74	0-2	L1 Timer - FSI	Edge
VBA:$76	0-2	L1 Timer - CVR1	Level
VBA:$78	0-2	L1 Timer - CVR2	Level
VBA:$7A	0-2	L1 Timer - CVR3	Level
VBA:$7C	0-2	L1 Timer - CVR4	Level

Table 6-10. DSP Interrupt Sources

Interrupt Starting Address	Priority	Interrupt Source	Edge/Level
VBA:$7E	0-2	L1 Timer - CVR5	Level
VBA:$80	0-2	L1 Timer - CVR6	Level
VBA:$82	0-2	L1 Timer - CVR7	Level
VBA:$84	0-2	L1 Timer - CVR8	Level
VBA:$86	0-2	L1 Timer - CVR9	Level
VBA:$88	0-2	L1 Timer - CVR10	Level
VBA:$8A	0-2	L1 Timer - CVR11	Level
VBA:$8C	0-2	L1 Timer - CVR12	Level
VBA:$8E	0-2	L1 Timer - CVR13	Level
VBA:$90	0-2	L1 Timer - CVR14	Level
VBA:$92	0-2	L1 Timer - CVR15	Level
VBA:$94	0-2	Reserved	Level
VBA:$96	0-2	Reserved	Level
VBA:$98	0-2	DSP Debug Flag	Level
VBA:$9A-A3	0-2	Reserved	Level
VBA:$A4	0-2	DTimer 0 Compare	Level
VBA:$A6	0-2	DTimer 0 Overflow	Level
VBA:$A8	0-2	DTimer 1 Compare	Level
VBA:$AA	0-2	DTimer 1 Overflow	Level
VBA:$AC	0-2	DTimer 2 Compare	Level
VBA:$AE	0-2	DTimer 2 Overflow	Level
VBA:$B0-B7	0-2	Reserved	Level
VBA:$B8	3	DMA1 Time Out (NMI)	Level
VBA:$BA	0-2	DMA1 Terminal Count	Level
VBA:$BC	0-2	DMA1 Word Count	Level
VBA:$BE-C7	0-2	Reserved	Level
VBA:$C8	3	DMA2 Time Out (NMI)	Level
VBA:$CA	0-2	DMA2 Terminal Count	Level
VBA:$CC	0-2	DMA2 Word Count	Level
VBA:$CE-FF	0-2	Reserved	Level

The IRQA interrupt source is the only source that can wake the DSP out of the low power STOP mode. In order to increase the number of interrupt sources that can wake the DSP out of STOP mode, the IRQB, IRQC, and IRQD interrupt sources have been connected together as shown in Figure on page 6-70. When an IRQB, IRQC, or IRQD interrupt is asserted, IRQA will be asserted as well, waking the DSP from STOP mode. IRQB is asserted by the GPIO module, IRQC by the MDI, and IRQD by the Layer 1 Timer.

If used, the IRQB, IRQC, and IRQD interrupts should be individually enabled in the Interrupt Priority Register C (IPRC). For proper operation, the IRQA interrupt should remain disabled in IPRC. When an IRQB, IRQC, or IRQD interrupt occurs, the IRQA signal will be asserted and wake the DSP from STOP mode, even though IRQA is not enabled. Then the appropriate IRQB, IRQC, or IRQD interrupt service routine will be executed. The IRQC and IRQD interrupts must be cleared in the interrupt service routine by setting the appropriate bits in the MDI module.

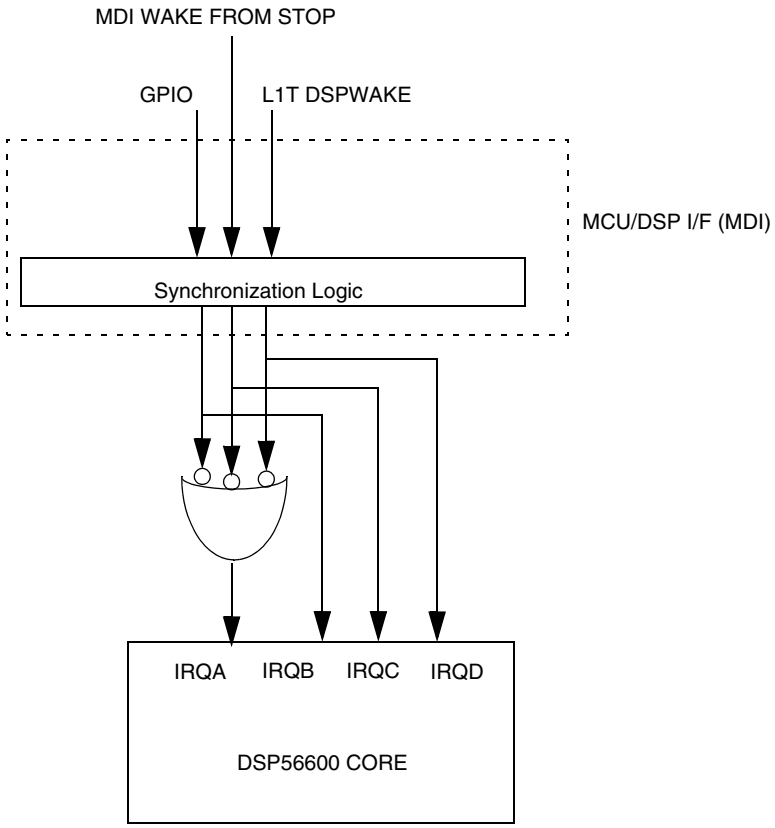


Figure 6-4. RQA Interrupt Connection Within Neptune

When more than one interrupt of the same priority occurs, a second fixed-priority hardware module exists in hardware to determine which interrupt source is serviced. The hardware is split between the DSP interrupt controller, the Special Interrupt Handler, DSIH, and each peripheral interrupt generation logic. The functionality of this split hardware is defined by Table 6-11 and Table 6-12. The second level priority order of the Neptune interrupt sources is shown in the table below.

Table 6-11. Second Level Priority Ordering for DSP Interrupt Sources

Priority	Interrupt Source
Level 3 (Non Maskable Interrupts)	
Highest	Hardware RESET
	Stack Error
	Illegal Instruction
	Debug Request Interrupt

Table 6-11. Second Level Priority Ordering for DSP Interrupt Sources (Continued)

Priority	Interrupt Source
	Trap
Lowest	MDI MCU NMI or DMA1/DMA2 Time out
Levels 0, 1, 2 (Maskable Interrupts)	
Highest	IRQA - GPIO or MDI or L1Timer
	IRQB - GPIO
	IRQC - MDI
	IRQD - L1 Timer
	MDI MCU Default Command
	USB CVR7 - Reset
All USB interrupts except Reset are not prioritized against each other. All other USB interrupts except Reset will be fed to the DSP based upon the time at which they were generated.	USB CVR0 - Endpoint 5 USB CVR1 - Endpoint 6 USB CVR2 - Endpoint 7 USB CVR3 - Endpoint 8 USB CVR4 - Endpoint 9 USB CVR5 - Start Of Frame USB CVR6 - Suspend/Resume USB Endpoint 11
	BBP Receive Data with Exception
	BBP Receive Data
	BBP Receive last slot
	BBP Receive Frame Counter
	BBP transmit Data with Exception
	BBP Transmit last slot
	BBP Transmit Data
	BBP Transmit Frame counter
	SAP Receive Data with Exception
	SAP Receive Data
	SAP Receive last slot
	SAP Transmit Data with Exception
	SAP Transmit last slot
	SAP Transmit Data
	SAP Timer counter rollover
	VOCOD RX Exception
	VOCOD TX Exception
	VOCOD Receive & Transmit Ready
	DSP Debug Flag
	L1 Timer RX
	L1 Timer TX
	L1 Timer FSI

Table 6-11. Second Level Priority Ordering for DSP Interrupt Sources (Continued)

Priority	Interrupt Source
	L1 Timer CVR1
	L1 Timer CVR2
	L1 Timer CVR3
	L1 Timer CVR4
	L1 Timer CVR5
	L1 Timer CVR6
	L1 Timer CVR7
	L1 Timer CVR8
	L1 Timer CVR9
	L1 Timer CVR10
	L1 Timer CVR11
	L1 Timer CVR12
	L1 Timer CVR13
	L1 Timer CVR14
	L1 Timer CVR15
	DTimer 0 Overflow
	DTimer 0 Compare
	DTimer 1 Overflow
	DTimer 1 Compare
	DTimer 2 Overflow
	DTimer 2 Compare
	MDI Receive 0
	MDI Receive 1
	MDI Transmit 0
	MDI Transmit 1
	DMA1 Terminal Count
	DMA1 Word Count
	DMA2 Terminal Count
	DMA2 Word Count
	VIAC Error
	VIAC Processing Complete
Lowest	LEM Encryption Complete

The DSP56600 Core is limited to eight peripheral interrupt sources. This requires that some of the Neptune sources to be OR'ed together at the input to the DSP56600. The following table shows the eight interrupt inputs to the DSP56600, and their respective sources.

NOTE:

This function is performed by the DSP Special interrupt handler module (DSIH).

Table 6-12. DSP Interrupt Sources to the DSP56600 Core

DSP56600 Peripheral Interrupt	Interrupt Sources
Interrupt 0	MDI Command
Interrupt 1	USB
Interrupt 2	BBP
Interrupt 3	SAP,DSP Debug, VOCOD
Interrupt 4	L1 Timer, DTimer
Interrupt 5	MDI
Interrupt 6	DMA1,DMA2
Interrupt 7	VIAC, LEM

6.2.6 DSP56600 DMA Interface

This section describes the DMA1 and DMA2 module connection to the DSP Memory subsystem, Base Band Port (BBP), and the Serial Audio Port (SAP).

6.2.6.1 DMA to Memory Subsystem Interface

The two DMA modules and the VIAC module interface to dedicated Y memory RAM through the Data RAM Arbiter (DRA) module. This Y memory space is interleaved to reduce the probability of memory contention between the DMA modules and the DSP56600 Core. Refer to DSP memory map Figure 6-3. The memory locations are can be placed anywhere in the interleaved Y RAM memory area or can be assigned to these address for backward compatibility:

- DMA1: $0800 - $0FFF
- DMA2: $1000 - $177F
- VIAC: $1800 - $1FFF

6.2.6.2 DMA (s) to Peripheral Subsystem Interface

The following figure shows the DMA connections to the DSP memory and peripheral modules. Arbitration to the the SAP and BBP peripherals from DMA1 and DMA2 is handled by the DMA Arbiter (DMA_ARB) module.

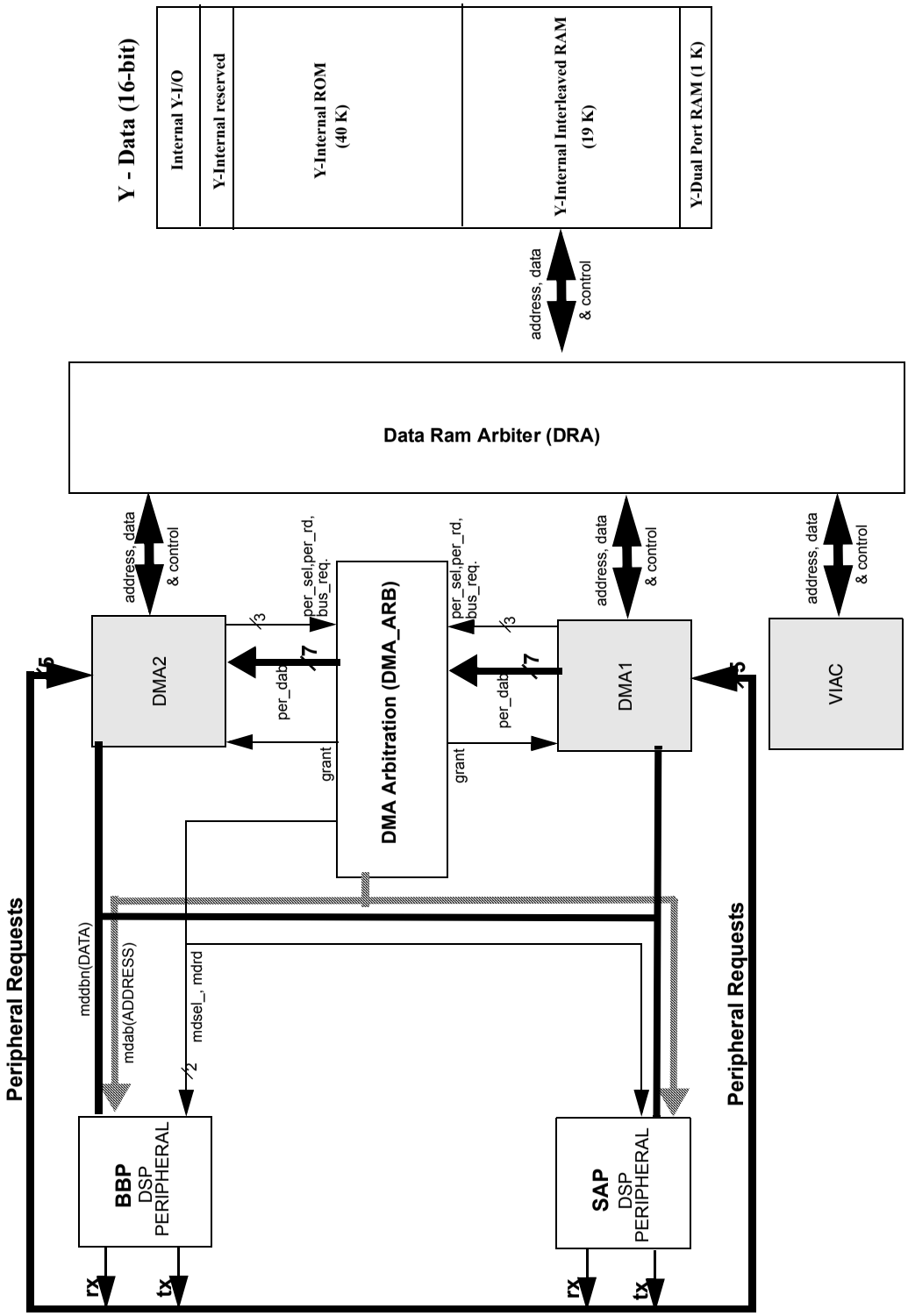


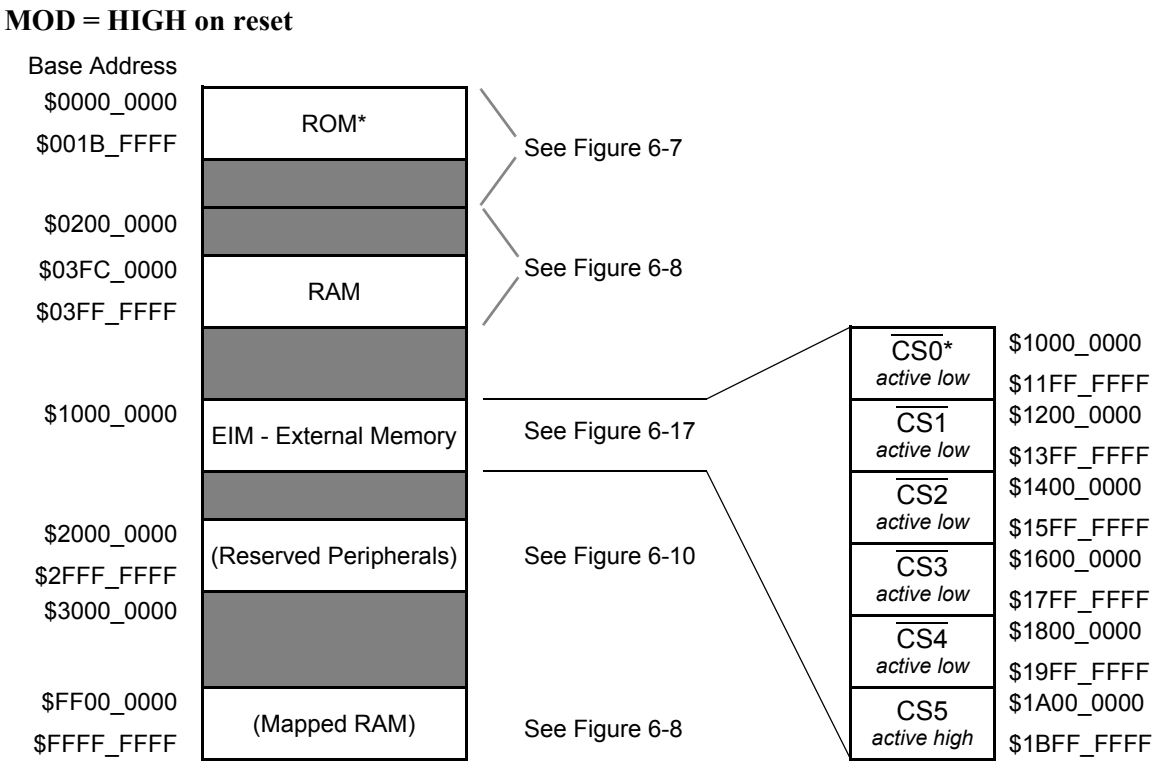
Figure 6-5. DSP DMA Configuration

6.3 MCU Memory Space

The MCU Memory Map is organized according to the following figure.

NOTE:

The Memory Map configuration is based on the state of the MOD pin at Reset. The Memory Map diagram Figure 6-6 and the Memory Map listing (Table 6-13) show the default configuration when the MOD pin is detected to be HIGH on chip reset. When the MOD pin is detected to be LOW on chip reset, the first 32 MBytes of the MCU memory space ($0000 0000 - 01FF FFFF) changes position in the memory map with the space defined for CS0 ($1000 0000 - 11FF FFFF).



*ROM location will be determined by the MOD pin on reset. (Figure shows MOD = HIGH)

Shaded areas are reserved.

Figure 6-6. Neptune LTE MCU Memory Map (MOD - High)

6.3.1 On-Chip MCU ROM

32 megabytes (4x1024Kx32) of memory map is allocated for internal MCU ROM, although it is not fully populated with physical memory. Location of the boot ROM is dependent upon the state of the MOD pin at reset. The state of the MOD pin at reset is used by the MCU to determine where to fetch the first memory value which is loaded onto the PC for the starting program address. The memory access is from either the internal ROM first address ($0000_0000) or the external memory connected to CS0 ($1000_0000). If the

MOD pin is logic-low at reset, then the internal MCU ROM will be disabled and CS0 will be asserted for the first MCU cycle following reset negation. If MOD is logic-high at reset, then the internal ROM will be enabled, and the MCU will fetch the first word from internal ROM. Read access to areas which are not implemented will result in an abort exception.

ARM7 platform is configured so that accesses to MCU ROM are allowed in all ARM operating modes.

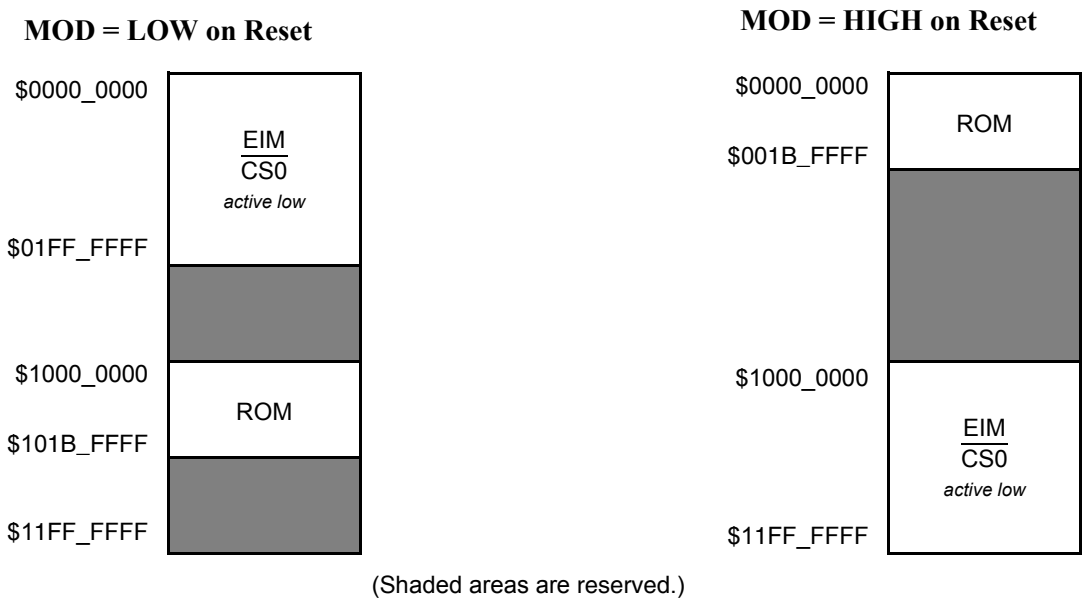


Figure 6-7. Neptune LTE ROM Boot configuration

6.3.2 On-Chip MCU RAM

Thirty-two (32) megabytes of the memory map are allocated for the MCU RAM ($0200_0000 - $03FF_FFFF) although it is not fully populated with physical memory. 64K x 32 Bits of physical memory exist from $03FC_0000 - $03FF_FFFF for Neptune LTE.

Accesses to the upper 4KB of RAM will cause an abort in user mode. This will be true for the aliased RAM area as well as the “physical” RAM area. This has been added to provide a security feature for the watchpoint registers. (See AWPT in Chapter 43 for description of watchpoint operation.)

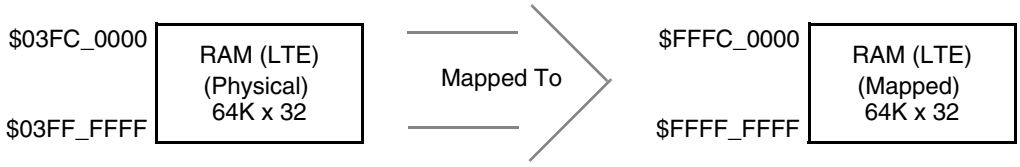
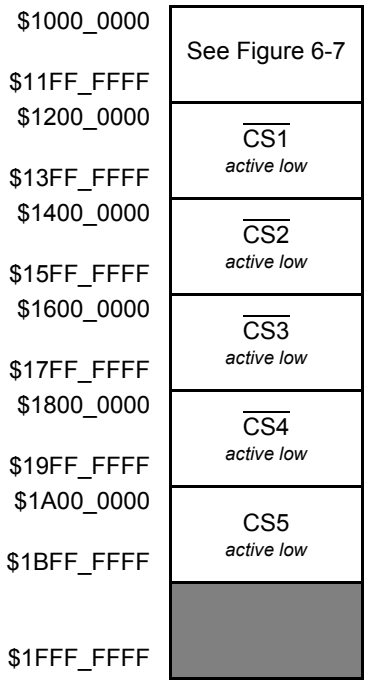


Figure 6-8. Neptune LTE MCU RAM Memory Spaces

6.3.3 MCU External Memory

Two hundred and fifty-six (256) megabytes of memory map is allocated for external chip accesses and starts at address $1000_0000. There are six (6) external chip selects that are allocated 32 megabytes each. All chip selects except CS5 are active low. Refer to the EIM section for more specific details.



NOTE:

The location of $\overline{CS0}$ in the memory map depends on the status of the MOD pin at reset. Please refer to Section 6.3.1, “On-Chip MCU ROM.” for a detailed explanation.

$\overline{CS4}$ and CS5 are not available in 225-pin packages.

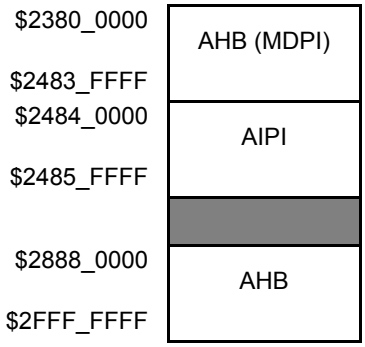
Figure 6-9. Netpune LTE EIM (External) Memory Space

6.3.4 MCU Peripherals

256 Megabytes of memory map is allocated for internal MCU peripherals (2000_0000-3000_0000). Each AIPI peripheral is allocated 4K bytes each, with the remainder reserved for future expansion. By using the aliased area provided in the figure below, this will allow for external trace analysis to decode a unique address.

Within the 4K bytes of AIPI peripheral space, any number of registers or peripheral memory may be defined, and software should explicitly address them (no assumptions on multiple mappings). Read access to an unmapped space within the 4K of allocated of peripheral address space may return a transaction abort. Write accesses of this type (unmapped space) may also return a transaction abort, and **may** alter the contents of a peripheral location within the selected 4K range.

Two main busses are located in this space. AIPI bus, and the AHB bus. Any attempt to access within the reserved portion of the peripheral memory space will result in an abort exception.



NOTE:

AIPI Peripherals are allocated 4K of address space. Access to unimplemented areas within each address space may not result in a an access abort.

Shaded areas indicate reserved (unimplemented) areas of memory.

Figure 6-10. MCU Peripheral Memory Space

6.3.4.1 MDPI

MDPI provides MCU access to the DSP dual-port RAM. See Chapter 8, “DSP Memory (DSPMEM),” and Chapter 65, “MCU to Dual-Port-Ram Peripheral Interface (MDPI),” for more information.

6.3.4.2 AIPI

In order to access peripherals located in the AIPI bus the port size should be programmed to reflect the proper access. This may be set by writing a value of 0xFCBF_F304 to the PSR0 register and a value of 0x0720_0CFF to the PSR1. All AIPI peripherals have 2 cycle read and 3 cycle write timing.

6.3.4.3 AHB

All Peripherals located on the AHB, have read/write access times of 1 cycle/access.

6.3.4.4 MDI

See the MDI specification chapter for detailed access timing and restrictions.

6.3.4.5 AEIM

The AEIM's CS0_PROM_CFG register (which configures CS0) has a reset value which is dependent upon the EBW pin. This register will select between 8, 16 and 32 bit accesses to external memory. AEIM is located on the AHB, therefore supports up to single cycle access.

NOTE:

Neptune LTE do not support EBW selection using the EBW pin. Reset boot configuration of 16-bit external memory is automatically configured. CS0 bus width configuration can still be changed in the CS0_PRIM_CFG by software.

6.3.5 MCU Memory Map

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$0000_0XXX		Internal ROM
	$0000_0000	mcu_rom_base_address	ROM
	$001B_FFFF	mcu_rom_lte_end_address	End of ROM (Neptune LTE) 448K x 32 Bits
	$0200_0XXX		Internal RAM
	$03FC_0000	mcu_ram_lte_base_address	Beginning of RAM (Neptune LTE) 64K x 32 Bits
	$03FF_FFFF	mcu_ram_end_address	End of RAM
CS0	$1000_0000	cs0_base_address	Chip Select 0 through
CS0	$11FF_FFFF		Chip Select 0 - End
CS1	$1200_0000	cs1_base_address	Chip Select 1through
CS1	$13FF_FFFF		Chip Select 1 - End
CS2	$1400_0000	cs2_base_address	Chip Select 2 through
CS2	$15FF_FFFF		Chip Select 2 - End
CS3	$1600_0000	cs3_base_address	Chip Select 3 through
CS3	$17FF_FFFF		Chip Select 3 - End
CS4	$1800_0000	cs4_base_address	Chip Select 4
CS4	$19FF_FFFF		Chip Select 4 - End
CS5	$1A00_0000	cs5_base_address	Chip Select 5
CS5	$1BFF_FFFF		Chip Select 5 - End
	$1C00_0000		Reserved through
	$1FFF_FFFF		Reserved - End
MDPI	**$2380_XXXX**		**MCU - Dual Port RAM Interface (MDPI)**
	$2380_0000		MDPI - Shared X RAM through
	$2380_07FF		MDPI - Shared X RAM

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2380_0800		MDPI - Reserved through
	$2380_FFFF		MDPI - Reserved
	$2382_0000		MDPI - Shared Y RAM through
	$2382_07FF		MDPI - Shared Y RAM
	$2382_0800		MDPI - Reserved through
	$2382_FFFF		MDPI - Reserved
	$240X_XXXX		**AIPI Bus**
AIPI	**$2484_0000**	**aipi_registers_base_address**	**AIPI Module**
	$2484_0000	aipi_psr0	Peripheral Size Register 0
	$2484_0004	aipi_psr1	Peripheral Size Register 1
	$2484_0008	aipi_par	Peripheral Access Register
GPIO	**$2484_1000**	**gpio_registers_base_address**	**General Purpose I/O Module**
	$2484_1000	gpio_pacr0	GPIO - PAConfReg0
	$2484_1004	gpio_pacr1	GPIO - PAConfReg1
	$2484_1008	gpio_paddr	GPIO - PADDirReg
	$2484_100C	gpio_paimr	GPIO - PAIMaskReg
	$2484_1010	gpio_paicrr	GPIO - PAICtrlReg
	$2484_1014	gpio_paair	GPIO - PAAltInReg
	$2484_1018	gpio_pbcr0	GPIO - PBConfReg0
	$2484_101C	gpio_pbcr1	GPIO - PBConfreg1
	$2484_1020	gpio_pbddr	GPIO - PBDirReg
	$2484_1024	gpio_pbimr	GPIO - PBIMaskReg
	$2484_1028	gpio_pbicr	GPIO - PBIntCtrlReg
	$2484_102C	gpio_pbair	GPIO - PBAltInReg
	$2484_1030	gpio_pccr0	GPIO - PCConfReg0
	$2484_1034	gpio_pccr1	GPIO - PCConfReg1
	$2484_1038	gpio_pcddr	GPIO - PCDirReg

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2484_103C	gpio_pcimr	GPIO - PCIMask
	$2484_1040	gpio_pcicr	GPIO - PCIntCtrlReg
	$2484_1044	gpio_pcair	GPIO - PCAltInReg
	$2484_1048	gpio_pdcr0	GPIO - PDConfReg0
	$2484_104C	gpio_pdcr1	GPIO - PDConfReg1
	$2484_1050	gpio_pdddr	GPIO - PDDirReg
	$2484_1054	gpio_pdimr	GPIO - PDIMaskReg
	$2484_1058	gpio_pdicr	GPIO - PDIntCtrlReg
	$2484_105C	gpio_pdair	GPIO - PDAltInReg
	$2484_1060	gpio_pecr0	GPIO - PEConfReg0
	$2484_1064	gpio_pecr1	GPIO - PEConfReg1
	$2484_1068	gpio_peddr	GPIO - PEDirReg
	$2484_106C	gpio_peimr	GPIO - PEIMaskReg
	$2484_1070	gpio_peicr	GPIO - PEIntCtrlreg
	$2484_1074	gpio_peair	GPIO - PEAltIntReg
	$2484_1078	gpio_isr	GPIO - IntSelReg
	$2484_107C	gpio_intsr	GPIO - IntStatReg
	$2484_1080	gpio_padr	GPIO - MCUA DataReg
	$2484_1084	gpio_pbdr	GPIO - MCUB DataReg
	$2484_1088	gpio_pcdr	GPIO - MCUC DataReg
	$2484_108C	gpio_pddr	GPIO - MCUD DataReg
	$2484_1090	gpio_pedr	GPIO - MCUE DataReg
	$2484_1094	gpio_bgref	GPIO - MCUBGREF
	$2484_1098		GPIO - Reserved Through
	$2484_1FFF		GPIO - Reserved
RTC	**$2484_3000**	**rtc_registers_base_address**	**Real Time Clock**
	$2484_3000	rtc_scr	RTC - Control Status Register (RTCCSR)
	$2484_3004	rtc_t	RTC - Time of Day Register (RTCT)

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2484_3008	rtc_a	RTC - Alarm (RTCA)
	$2484_300C	rtc_pcram0	RTC - PC RamRegister (RTCA)
		rtc_pcram[1:15]	
	$2484_3010		RTC- Reserved Through
	$2484_3FFF		RTC- Reserved
TCM	**$2484_4000**	**tcm_registers_base_address**	**Test Control Module**
	$2484_4000	tcm_tcr	TCM -(TCR)
	$2484_4004		TCM - Reserved
	$2484_4008	tcm_mtcr	TCM - (MTCR)
	$2484_400C	tcm_mbistr	TCM - (MBISTR)
	$2484_4010	tcm_ana_cntl	TCM - (TCM_ANA_CNTL)
	$2484_4014	tcm_ana_data1	TCM - Analog test DATA1
	$2484_4018	tcm_ana_data2	TCM - Analog test DATA2
	$2484_401C	tcm_mcuinbist	TCM - MCU Incoming Bist Register (MCUInBist)
	$2484_4020		TCM - Reserved through
	$2484_4FFF		TCM - Reserved
CCM	**$2484_5000**	**ccm_registers_base_address**	**Clock Control Module**
	$2484_5000	ccm_udpl_cr	CCM - USB DPLL Control Register (CTL)
	$2484_5004	ccm_udpl_op	CCM - USB DPLL Operation Register (OP)
	$2484_5008	ccm_udpl_mfn	CCM - USB DPLL Multiply Factor Numerator (MFN)
	$2484_500C	ccm_udpl_mfd	CCM - USB DPLL Multiply Factor Denominator (MFD)
	$2484_5010	ccm_rdpl_cr	CCM - REF Control Register (CTL)
	$2484_5014	ccm_rdpl_op	CCM - REF Operation Register (OP)
	$2484_5018	ccm_rdpl_mfn	CCM - Multiply Factor Numerator (MFN)
	$2484_501C	ccm_rdpl_mfd	CCM - Multiply Factor Denominator (MFD)
	$2484_5020	ccm_dfr	CCM - Division Factor Register (DFR)
	$2484_5024	ccm_csr	CCM - Clock Selection Register (CSR)

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2484_5028	ccm_cmon	CCM - Clock Monitor Register (CMON)
	$2484_502C	ccm_lpmdr	CCM - Low-Power Mode Register (LPMDR)
	$2484_5030	ref_pll_mfn_minus	CCM - MFN Minus Value Register (MFN_MINUS)
	$2484_5034	ref_pll_mfn_plus	CCM - MFN Plus Value Register (MFN_PLUS)
	$2484_5038	mfn_tog_cnt	CCM - MFN Toggle Count Register (TOG_CNT)
	$2484_503C	desense_stat	CCM - Desense Status Register (DESENSE_STAT)
	$2484_5040		Clock Monitor - Reserved through
	$2484_5FFF		Clock Monitor - Reserved
A2DIGL	**$2484_6XXX**	**a2digl_registers_base_address**	**Analog 2 Digital Control Module**
	$2484_6000	a2digl_pwrunflt	A2DIGL -PWRUNFLT
	$2484_6004	a2digl_pwrdac	A2DIGL -PWRDAC
	$2484_6008	a2digl_errorsig	A2DIGL -ERRORSIG
	$2484_600C	a2digl_muxctl	A2DIGL-MUXCTL
	$2484_6010	a2digl_bplus	A2DIGL-GPADC -BPLUS
	$2484_6014	a2digl_temp	A2DIGL-GPADC -TEMP
	$2484_6018	a2digl_ad1	A2DIGL-GPADC -AD1
	$2484_601C	a2digl_ref_reg	A2DIGL- Reference Regulator
	$2484_6020	a2digl_tx_cpdiv2	A2DIGL-GPADC -TX_CP_Div2
	$2484_6024	a2digl_rx_cpdiv2	A2DIGL-GPADC -RX_CP_Div2
	$2484_6028	a2digl_ctrl	A2DIGL-GPADC -CTRL
	$2484_602C	a2digl_whigh	A2DIGL-GPADC-WHIGH
	$2484_6030	a2digl_wlow	A2DIGL-GPADC -WLOW
	$2484_6034	a2digl_mtr	A2DIGL-GPADC -MTR
	$2484_6038	a2digl_tcode	A2DIGL-TCODE
	$2484_603C	a2digl_ttest	A2DIGL-TTEST
	$2484_6040	a2digl_osc26m	A2DIGL-OSC26M
	$2484_6044		A2DIGL - Reserved through

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2484_6FFF		A2DIGL - Reserved
AMARB	**$2484_7000**	**amarb_registers_base_address**	**External Alternate Bus Master**
	$2484_7000	amarb_ctl	AMARB- Control Register
	$2484_7004	amarb_interval	AMARB- Interval
	$2484_7008	amarb_soft_priority	AMARB- Soft Priority
	$2484_700C	amarb_port0_priority	AMARB- Port 0 Priority
	$2484_7010	amarb_port1_priority	AMARB- Port 1 Priority
	$2484_7014	amarb_port2_priority	AMARB- Port 2 Priority
	$2484_7018	amarb_port3_priority	AMARB- Port 3 Priority
	$2484_701C		AMARB- Reserved Through
	$2484_7FFF		AMARB- Reserved
EGPT	**$2484_8000**	egpt_registers_base_address	**Enhanced General Purpose Timers and PWM**
	$2484_8000	egpt_egptcr	EGPT - Control Register (EGPTCR)
	$2484_8002	egpt_egptmr	EGPT - Mode Register (EGPTMR)
	$2484_8004	egpt_egptsr	EGPT - Status Register (EGPTSR)
	$2484_8006	egpt_egptir	EGPT - Interrupt Enable Register (EGPTIR)
	$2484_8008	egpt_tocr1	EGPT - Timer 1 Output Compare Register (TOCR1)
	$2484_800A	egpt_tocr3	EGPT - Timer 3 Output Compare Register (TOCR3)
	$2484_800C	egpt_tocr4	EGPT - Timer 4 Output Compare Register (TOCR4)
	$2484_800E	egpt_ticr1	EGPT - Timer 1 Input Capture Register (TICR1)
	$2484_8010	egpt_ticr2	EGPT - Timer 2 Input Capture Register (TICR2)
	$2484_8012	egpt_pwocr	EGPT - (PWM) Output Compare Register (PWOCR)
	$2484_8014	egpt_tcr	EGPT - (PWM) Timer Count Register (TCR)
	$2484_8016	egpt_pwcr	EGPT - (PWM) Count Register (PWCR)

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2484_8018	egpt_pwcnr	EGPT - (PWM) Counter Register (PWCNR)
	$2484_801A		EGPT - Reserved
	$2484_801C	egpt_itadrh	EGPT- (EPIT) Alternate Data Register High (ITADH)
	$2484_801E	egpt_itadrl	EGPT - (EPIT) Alternate Data Register Low (ITADL)
	$2484_8020	egpt_itdrh	EGPT - (EPIT) High Data Register (ITDRH)
	$2484_8022	egpt_itdrl	EGPT - (EPIT) Low Data Register (ITDRL)
	$2484_8024	egpt_itcrh	EGPT - (EPIT) High Compare Register (AITCRH)
	$2484_8026	egpt_itcrl	EGPT - (EPIT) Low Compare Register (AITCRL)
	$2484_8028		EGPT - Reserved through
	$2484_87FF		EGPT - Reserved
WDOG	**$2484_9000**	wdog_registers_base_address	**WatchDog Timer Module**
	$2484_9000	wdog_wcr	WDOG - Watch Dog Control Register (WCR)
	$2484_9002	wdog_wsr	WDOG - Watch Dog Service Register (WSR)
	$2484_9004	wdog_wrsr	WDOG - Watch Dog Reset Status Register (WRSR)
	$2484_9006		WDOG - Reserved through
	$2484_9FFF		WDOG - Reserved
RTR	**$2484_A000**	rtr_registers_base_address	**Real Time Reference Module**
	$2484_A000	rtr_msb	RTR - MSB Register - RTRH[31:0]
	$2484_A004	rtr_lsb	RTR - LSB Register - RTRL[31:0]
	$2484_A008	rtr_rtrc	RTR - Control Register - RTRC[31:0]
	$2484_A00A		RTR - Reserved through
	$2484_AFFF		RTR - Reserved
DSM	**$2484_B000**	dsm_registers_base_address	**Deep Sleep Module**
	$2484_B000	dsm_count32	DSM - COUNT32 Register
	$2484_B004	dsm_refcount	DSM - REFCOUNT Register
	$2484_B008	dsm_meastime	DSM - MEASTIME Register
	$2484_B00C	dsm_sleeptime	DSM - SLEEPTIME Register

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2484_B010	dsm_restart_time	DSM - RESTART_TIME Register
	$2484_B014	dsm_waketime	DSM - WAKETIME Register
	$2484_B018	dsm_warmtime	DSM - WARMTIME Register
	$2484_B01C	dsm_locktime	DSM - LOCKTIME Register
	$2484_B020	dsm_control0	DSM - CONTROL 0 Register
	$2484_B024	dsm_control1	DSM - CONTROL 1 Register
	$2484_B028	dsm_ctren	DSM - CTREN Register
	$2484_B02C	dsm_status	DSM - STATUS Register
	$2484_B030	dsm_state	DSM - STATE Register
	$2484_B034	dsm_int_stat	DSM - INT_STATUS Register
	$2484_B038	dsm_mask	DSM - MASK Register
	$2484_B03C	dsm_cnt32_cap	DSM - COUNT32_CAP Register
	$2484_B040	dsm_warmper	DSM - WARMPER Register
	$2484_B044	dsm_lockper	DSM - LOCKPER Register
	$2484_B048	dsm_poscount	DSM - POSCOUNT Register
	$2484_B04C	dsm_mgper	DSM - MGPER Register
	$2484_B050		DSM - Reserved through
	$2484_BFFF		DSM - Reserved
INT	**$2484_C000**	int_registers_base_address	**MCU External Interrupt Module (INT)**
	$2484_C000	int_eppar	INT- Edge Port Pin Assignment Register (EPPAR)
	$2484_C002	int_epddr	INT- Edge Port Data Direction Register (EPDDR)
	$2484_C004	int_epdr	INT- Edge Port Data Register (EPDR)
	$2484_C006	int_epfr	INT- Edge Port Flag Register (EPFR)
	$2484_C007		INT - Reserved through
	$2484_CFFF		INT - Reserved
UART1	**$2484_D000**	uart1_base_address	**Universal Asynchronous RX TX Module (UART1)**
	$2484_D000	uart1_urx_start	UART1 RX Register_1 (URX) - Start

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2484_D03E	uart1_urx_end	UART1 RX Register_1 (URX) - End
	$2484_D040	uart1_utx_start	UART1 TX Register_1 (UTX) - Start
	$2484_D07E	uart1_utx_end	UART1 TX Register_1 (UTX) - Start
	$2484_D080	uart1_ucr1	UART1 Control Register 1 - UCR1[15:0]
	$2484_D082	uart1_ucr2	UART1 Control Register 2 - UCR2[15:0]
	$2484_D084	uart1_ucr3	UART1 Control Register 3 - UCR3[15:0
	$2484_D086	uart1_ucr4	UART1 Control Register 4- UCR4[15:0]
	$2484_D088	uart1_ufcr	UART1 FIFO Control Register - UFCR[15:0]
	$2484_D08A	uart1_usr1	UART1 Status Register 1 - USR1[15:0]
	$2484_D08C	uart1_usr2	UART1 Status Register 2 - USR2[15:0]
	$2484_D08E	uart1_uesc	UART1 Escape Character Register - UESC[15:0]
	$2484_D090	uart1_utim	UART1 Escape Timer Register - UTIM
	$2484_D092	uart1_ubir	UART1 BRM INC Register - UBIR
	$2484_D094	uart1_ubmr	UART1 BRM MOD Register - UBMR
	$2484_D096	uart1_ubrc	UART1 Baud Rate Count Register - UBRC
	$2484_D098	uart1_bipr1	UART1 BRM INC Preset Register 1 - BIPR1
	$2484_D09A	uart1_bmpr1	UART1 BRM MOD Preset Register 1 - BMPR1
	$2484_D09C	uart1_bipr2	UART1 BRM INC Preset Register 2 - BIPR2
	$2484_D09E	uart1_bmpr2	UART1 BRM MOD Preset Register 2 - BMPR2
	$2484_D0A0	uart1_bipr3	UART1 BRM INC Preset Register 3 - BIPR3
	$2484_D0A2	uart1_bmpr3	UART1 BRM MOD Preset Register 3 - BMPR3
	$2484_D0A4	uart1_bipr4	UART1 BRM INC Preset Register 4 - BIPR4
	$2484_D0A6	uart1_bmpr4	UART1 BRM MOD Preset Register 4 - BMPR4
	$2484_D0A8	uart1_uts	UART1 Test Register - UTS
	$2484_D0AA		UART1 Reserved through
	$2484_DFFF		UART1 Reserved
KPP	**$2484_E000**	kpp_registers_base_address	**Key Pad GPIO**
	$2484_E000	kpp_kpcr	Keypad - Control Register (KPCR)

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2484_E002	kpp_kpsr	Keypad - Status Register (KPSR)
	$2484_E004	kpp_kddr	Keypad - Direction Register (KDDR)
	$2484_E006	kpp_kpdr	Keypad - Data Register (KPDR)
	$2484_E008		Keypad - Reserved through
	$2484_EFFF		Keypad - Reserved
SIM	**$2484_F000**	sim1_registers_base_address	SIM Interface Module
	$2484_F000	sim1_port1_cntl	SIM- Port 1 Control Register
	$2484_F002	sim1_setup	SIM- Setup Register 1
	$2484_F004	sim1_port1_detect	SIM - Port 1 Detect Register (PORT1_DETECT_1)
	$2484_F006		SIM - Reserved
	$2484_F008		SIM - Reserved
	$2484_F00A		SIM - Reserved
	$2484_F00C	sim1_port1_xmt_buff	SIM - Port 1 TX Buffer_1
	$2484_F00E	sim1_port1_rcv_buff	SIM - Port 1 RX Buffer_1
	$2484_F010	sim1_port0_cntl	SIM - Port 0 Control Register (PORT0_CNTL)_1
	$2484_F012	sim1_cntl	SIM - Control Register_1
	$2484_F014	sim1_rcv_threshold	SIM - RX Threshold (RCV_THRESHOLD_1)
	$2484_F016	sim1_enable	SIM - Enable Register_1
	$2484_F018	sim1_xmt_status	SIM - TX Status Register (XMT_STATUS_1)
	$2484_F01A	sim1_rcv_status	SIM - RX Status Register (RCV_STATUS_1)
	$2484_F01C	sim1_int_mask	SIM - Interrupt Mask Register (INT_MASK_1)
	$2484_F01E	sim1_port0_xmt_buf	SIM - Port 0 TX Buffer (PORT0_XMT_BUF_1)
	$2484_F020	sim1_port0_rcv_buf	SIM - Port 0 RX Buffer (PORT0_RCV_BUF_1)
	$2484_F022	sim1_port0_detect	SIM - Port 0 Detect Register (PORT0_DETECT_1)

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2484_F024	sim1_data_format	SIM - Data Format Register (DATA_FORMAT_1)
	$2484_F026	sim1_xmt_threshold	SIM - TX Threshold Register (XMT_TRESHOLD_1)
	$2484_F028	sim1_guard_cntl	SIM - Guard Control Register (GUARD_CNTL_1)
	$2484_F02A	sim1_od_config	SIM - OD Configuration Register (OD_CONFIG_1)
	$2484_F02C	sim1_reset_cntl	SIM - Reset Control Register (RESET_CNTL_1)
	$2484_F02E	sim1_chr_wtimer_reg	SIM - Character Wait Timer Register (CHAR_WAIT)
	$2484_F030	sim1_gen_purpose_creg	SIM - General Purpose Counter Register (GPCNT)
	$2484_F032	sim1_progr_div_reg	SIM - Programable Divisor Register (DIVISOR)
	$2484_F034	sim1_block_wait_msb	SIM - Block Wait Time Counter [31:16] (BLOCK WAIT MSB)
	$2484_F036	sim1_block_wait_lsb	SIM - Block Wait Time Counter [15:0] (BLOCK WAIT LSB)
	$2484_F038	sim1_char_wait_acc	SIM - Charachter Wait Time Accurate (CHAR WAIT ACC)
	$2484_F03A	sim1_block_guard	SIM - Block Guard Time Counter (BLOCK GUARD)
	$2484_F03C		SIM - Reserved through
	$2484_FFFF		SIM - Reserved
IIM	**$2485_0000**	**iim_registers_base_address**	**IC Identification Module**
	$2485_0000	iim_uid0	IIM - UID Register 0 (UID0)
	$2485_0002	iim_uid1	IIM - UID Register 1 (UID1)
	$2485_0004	iim_uid2	IIM - UID Register 2 (UID2)
	$2485_0006	iim_uid3	IIM - UID Register 3 (UID3)
	$2485_0008	iim_uid4	IIM - UID Register 4 (UID4)
	$2485_000A	iim_uid5	IIM - UID Register 5 (UID5)
	$2485_000C	iim_uid6	IIM - UID Register 6 (UID6)
	$2485_000E	iim_uid7	IIM - UID Register 7 (UID7)

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2485_0010	iim_rev	IIM - Hardware Revision Register (REV)
	$2485_0012		IIM - Reserved through
	$2485_0FFF		IIM - Reserved
MQSPI	**$2485_1000**	**mqspi_registers _base_address**	**Multi-Queue Serial Port Interface (MQSPI)**
	$2485_1000	mqspi_mcon	MQSPI Control Register
	$2485_1002	mqspi_qcfg_l	MQSPI - Control Queue Configuration Register, Low word
	$2485_1004	mqspi_qcfg_h	MQSPI - Control Queue Configuration Register, High word
	$2485_1006	mqspi_mtrig	MQSPI - Trigger Register
	$2485_1008	mqspi_stpr_l	MQSPI - Trigger Priority Register, Low word
	$2485_100A	mqspi_stpr_h	MQSPI - Trigger Priority Register, High word
	$2485_100C	mqspi_sdef	MQSPI - SPI Default Setup Register
	$2485_100E	mqspi_sflg	MQSPI - SPI Flag Register
	$2485_1010	mqspi_spim	MQSPI - SPI Interrupt Mask Register
	$2485_1012	mqspi_stff_l	MQSPI - SPI Transfer Finished Flags, Low word
	$2485_1014	mqspi_stff_h	MQSPI - SPI Transfer Finished Flags, High word
	$2485_1016	mqspi_stfe_l	MQSPI - SPI Transfer Finished Enables, Low word
	$2485_1018	mqspi_stfe_h	MQSPI - SPI Transfer Finished Enables, High word
	$2485_101A	mqspi_qre_l	MQSPI - Queue Read Enables, Low word
	$2485_101C	mqspi_qre_h	MQSPI - Queue Read Enables, High word
	$2485_101E	mqspi_sdi_md_l	MQSPI - Serial Display Interface, Low
	$2485_1020	mqspi_sdi_md_h	MQSPI - Serial Display Interface, High
	$2485_1022	mqspi_cdps1	MQSPI - SPI1 Control Data Pointer Status Register
	$2485_1024	mqspi_cdms1	MQSPI - SPI1 Control Data Mode Status Register
	$2485_1026	mqspi_cdps2	MQSPI - SPI2 Control Data Pointer Status Register
	$2485_1028	mqspi_cdms2	MQSPI - SPI2 Control Data Mode Status Register
	$2485_102A	mqspi_cscfg0a	MQSPI - Chip Select 0 Configuration Register A
	$2485_102C	mqspi_cscfg0b	MQSPI - Chip Select 0 Configuration Register B

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2485_102E	mqspi_cscfg1a	MQSPI - Chip Select 1 Configuration Register A
	$2485_1030	mqspi_cscfg1b	MQSPI - Chip Select 1 Configuration Register B
	$2485_1032	mqspi_cscfg2a	MQSPI - Chip Select 2 Configuration Register A
	$2485_1034	mqspi_cscfg2b	MQSPI - Chip Select 2 Configuration Register B
	$2485_1036	mqspi_cscfg3a	MQSPI - Chip Select 3 Configuration Register A
	$2485_1038	mqspi_cscfg3b	MQSPI - Chip Select 3 Configuration Register B
	$2485_103A	mqspi_cscfg4a	MQSPI - Chip Select 4 Configuration Register A
	$2485_103C	mqspi_cscfg4b	MQSPI - Chip Select 4 Configuration Register B
	$2485_103E	mqspi_cscfg5a	MQSPI - Chip Select 5 Configuration Register A
	$2485_1040	mqspi_cscfg5b	MQSPI - Chip Select 5 Configuration Register B
	$2485_1042	mqspi_cscfg6a	MQSPI - Chip Select 6Configuration Register A
	$2485_1044	mqspi_cscfg6b	MQSPI - Chip Select 6 Configuration Register B
	$2485_1046	mqspi_cscfg7a	MQSPI - Chip Select 7 Configuration Register A
	$2485_1048	mqspi_cscfg7b	MQSPI - Chip Select 7Configuration Register B
	$2485_104A	mqspi_cscfg8a	MQSPI - Chip Select 8 Configuration Register A
	$2485_104C	mqspi_cscfg8b	MQSPI - Chip Select 8 Configuration Register B
	$2485_104E	mqspi_cscfg9a	MQSPI - Chip Select 9 Configuration Register A
	$2485_1050	mqspi_cscfg9b	MQSPI - Chip Select 9 Configuration Register B
	$2485_1052	mqspi_qst1a	MQSPI - SPI1 Queue Status Register 1
	$2485_1054	mqspi_qst1b	MQSPI - SPI1 Queue Status Register 2
	$2485_1056		MQSPI - reserved
	$2485_1058		MQSPI - reserved
	$2485_105A	mqspi_sst	MQSPI - SPI Status Register
	$2485_105C	mqspi_shpfq1a	MQSPI - SPI1 High Priority FIFO Queue 1, LSW
	$2485_105E	mqspi_shpfq1b	MQSPI - SPI1 High Priority FIFO Queue 1, midW
	$2485_1060	mqspi_shpfq1c	MQSPI - SPI1 High Priority FIFO Queue 1, MSW
	$2485_1062	mqspi_slpfq1a	MQSPI - SPI1 Low Priority FIFO Queue 1, LSW
	$2485_1064	mqspi_slpfq1b	MQSPI - SPI1 Low Priority FIFO Queue 1, midW

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2485_1066	mqspi_slpfq1c	MQSPI - SPI1 Low Priority FIFO Queue 1, MSW
	$2485_1068		MQSPI - RESEREVD
	$2485_106A		MQSPI - RESEREVD
	$2485_106C		MQSPI - RESEREVD
	$2485_106E		MQSPI - RESEREVD
	$2485_1070		MQSPI - RESEREVD
	$2485_1072		MQSPI - RESERVED
	$2485_1074	mqspi_bcdp0s1	MQSPI - SPI1 Trigger 0 Burst CMD Data PTR Status Reg
	$2485_1076	mqspi_bcdm0s1	MQSPI - SPI1 Trigger 0 Burst CMD Data Mode Status Reg
	$2485_1078	mqspi_bcdp1s1	MQSPI - SPI1 Trigger 1 Burst CMD Data PTR Status Reg
	$2485_107A	mqspi_bcdm1s1	MQSPI - SPI1 Trigger 1 Burst CMD Data Mode Status Reg
	$2485_107C	mqspi_bcdp2s1	MQSPI - SPI1 Trigger 2 Burst CMD Data PTR Status Reg
	$2485_107E	mqspi_bcdm2s1	MQSPI - SPI1 Trigger 2 Burst CMD Data Mode Status Reg
	$2485_1080	mqspi_bcdp3s1	MQSPI - SPI1 Trigger 3 Burst CMD Data PTR Status Reg
	$2485_1082	mqspi_bcdm3s1	MQSPI - SPI1 Trigger 3 Burst CMD Data Mode Status Reg
	$2485_1084	mqspi_bcdp0s2	MQSPI - SPI2 Trigger 0 Burst CMD Data PTR Status Reg
	$2485_1086	mqspi_bcdm0s2	MQSPI - SPI2 Trigger 0 Burst CMD Data Mode Status Reg
	$2485_1088	mqspi_bcdp1s2	MQSPI -
	$2485_108A	mqspi_bcdm1s2	MQSPI -
	$2485_108C	mqspi_bcdp2s2	MQSPI -
	$2485_108E	mqspi_bcdm2s2	MQSPI -
	$2485_1090	mqspi_bcdp3s2	MQSPI -
	$2485_1092	mqspi_bcdm3s2	MQSPI -

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2485_1094	mqspi_bmas	MQSPI - Burst Mode Active Status Register
	$2485_1096		MQSPI - Reserved through
	$2485_13FF		MQSPI - Reserved
	$2485_1400	mqspi_cdr_start	MQSPI - Control Data RAM - Start
	$2485_147E	mqspi_cdr_end	MQSPI - Control Data RAM - End
	$2485_1480	mqspi_dsr_start	MQSPI - Data Storage RAM - Start
	$2485_167E	mqspi_dsr_end	MQSPI - Data Storage RAM - End
	$2485_1680		MQSPI - Reserved through
	$2485_1FFF		MQSPI - Reserved
USB	**$2485_2000**	**usb_registers_base_address**	**Universal Serial Bus Module**
	$2485_2000	usb_fad	USB - Function Address Register
	$2485_2002	usb_ier	USB - Interrupt Enable Register
	$2485_2004	usb_int	USB - CPU Interrupt Register
	$2485_2006	usb_e0rxcr	USB - Endpoint 0 RX Control Register
	$2485_2008	usb_e0txcr	USB- Endpoint 0 TX Control Register
	$2485_200A	usb_e1cr	USB - Endpoint 1 Control Register
	$2485_200C	usb_e2cr	USB - Endpoint 2 Control Register
	$2485_200E	usb_e3cr	USB - Endpoint 3 Control Register
	$2485_2010	usb_e4cr	USB - Endpoint 4 Control Register
	$2485_2012	usb_e10cr	USB - Endpoint 10 Control Register
	$2485_2014	usb_cpucr	USB - CPU Control Register
	$2485_2016		USB - Reserved through
	$2485_207F		USB - Reserved
	$2485_2080	usb_rxbuff0_start	USB - Endpoint 0 Buffer - RX (Start)
	$2485_2087	usb_rxbuff0_end	USB - Endpoint 0 Buffer - RX (End)
	$2485_2088	usb_txbuff0_start	USB - Endpoint 0 Buffer - TX (Start)
	$2485_208F	usb_txbuff0_end	USB - Endpoint 0 Buffer - TX (End)
	$2485_2090[1]	usb_rxbuff1_start	USB - Endpoint 1 - RX Buffer (Start)

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2485_209F[1]	usb_rxbuff1_end	USB - Endpoint 1 - RX Buffer (End)
	$2485_20A0[1]	usb_txbuff2_start	USB - Endpoint 2 TX - Buffer (Start)
	$2485_20AF[1]	usb_txbuff2_end	USB - Endpoint 2 - TX Buffer (End)
	$2485_20B0[1]	usb_rxbuff3_start	USB - Endpoint 3 - RX Buffer (Start)
	$2485_20BF[1]	usb_rxbuff3_end	USB - Endpoint 3 - RX Buffer (End)
	$2485_20C0[1]	usb_txbuff4_start	USB - Endpoint 4 - RX Buffer (Start)
	$2485_20DF[1]	usb_txbuff4_end	USB - Endpoint 4 - RX Buffer (End)
	$2485_20E0[1]	usb_txbuff10_start	USB - Endpoint 10 - TX Buffer (Start)
	$2485_20E7[1]	usb_txbuff10_end	USB - Endpoint 10 - TX Buffer (End)
	$2485_2100		USB - Reserved through
	$2485_2FFF		USB - Reserved
L1TIMER	**$2485_3XXX**	**l1t_registers_base_address**	**Layer 1 Timer Module**
	$2485_3000	l1t_ftect0_ram_start	L1T - Default Frame Table Event Codes RAM Start
	$2485_307E	l1t_ftect0_ram_end	L1T - Default Frame Table Event Codes RAM End
	$2485_3080	l1t_maect0_ram_start	L1T - Default Macro Table A Event Codes RAM Start
	$2485_30BE	l1t_maect0_ram_end	L1T - Default Macro Table A Event Codes RAM End
	$2485_30C0	l1t_mbect0_ram_start	L1T - Default Macro Table B Event Codes RAM Start
	$2485_30FE	l1t_mbect0_ram_end	L1T - Default Macro Table B Event Codes RAM End
	$2485_3100	l1t_mcect0_ram_start	L1T -Default Macro Table C Event Codes RAM Start
	$2485_313E	l1t_mcect0_ram_end	L1T - Default Macro Table C Event Codes RAM End
	$2485_3140	l1t_ptfd_start	L1T - Default Parameter Table Frame Delays Start
	$2485_317E	l1t_ptfd_end	L1T - Default Parameter Table Frame Delays End

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2485_3180		L1T - Reserved Space Start
	$2485_319E		L1T - Reserved Space End
	$2485_3200	l1t_ftect1_ram_start	L1T - Default Frame Table Events Quarter Bit Counts RAM Start
	$2485_327E	l1t_ftect1_ram_end	L1T - Default Frame Table Events Quarter Bit Counts RAM End
	$2485_3280	l1t_maect1_ram_start	L1T - Default Macro Table A Event s Quarter Bit Counts RAM Start
	$2485_32BE	l1t_maect1_ram_end	L1T - Default Macro Table A Event s Quarter Bit Counts RAM End
	$2485_32C0	l1t_mbect1_ram_start	L1T - Default Macro Table B Event s Quarter Bit Counts RAM Start
	$2485_32FE	l1t_mbect1_ram_end	L1T - Default Macro Table B Event s Quarter Bit Counts RAM End
	$2485_3300	l1t_mcect1_ram_start	L1T -Default Macro Table C Event s Quarter Bit Counts RAM Start
	$2485_333E	l1t_mcect1_ram_end	L1T - Default Macro Table C Event s Quarter Bit Counts RAM End
	$2485_3340	l1t_ptfd1_start	L1T - Default Parameter Table Quarter Bit Count Delays and Loop Counts RAM Start
	$2485_337E	l1t_ptfd1_end	L1T - Default Parameter Table Quarter Bit Count Delays and Loop Counts RAM End
	$2485_3380	l1t_mstr	L1T- Master Enable Register (MSTR)
	$2485_3382	l1t_gtc0	L1T - Global Time Control Register 0 - GTC0[15:0]
	$2485_3384	l1t_gtc1	L1T - Global Time Control Register 1 - GTC1[15:0]
	$2485_3386	l1t_gts0	L1T - Global Time Status Register 0 - GTS0[15:0]
	$2485_3388	l1t_gts1	L1T - Global Time Status Register 1 - GTS1[15:0]
	$2485_338A	l1t_igm	L1T - Interrupt Generator Mask Register - IGM[15:0]
	$2485_338C	l1t_gpwt	L1T - Global Programmable Wakeup Timer - GPWT[15:0]
	$2485_338E	l1t_es0	L1T - Error Source 0 Register - ES0[15:0]
	$2485_3390	l1t_em0	L1T - Error Mask 0 Register EM0[15:0]
	$2485_3392	l1t_es1	L1T - Error Source 1 Register - ES1[15:0]

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2485_3394	l1t_em1	L1T - Error Mask 1 Register EM1[15:0]
	$2485_3396	l1t_es2	L1T - Error Source 2 Register - ES2[15:0]
	$2485_3398	l1t_em2	L1T - Error Mask 2 Register EM2[15:0]
	$2485_339A	l1t_mcuis	L1T - MCU Interrupt Status Register - MCUIS[15:0]
	$2485_339C	l1t_mcuim	L1T - MCU Interrupt Mask Register - MCUIM[15:0]
	$2485_339E	l1t_dspim	L1T - DSP Interrupt Mask Register - DSPIM[15:0]
	$2485_33A0	l1t_tb_pr	L1T - Timing Block (TB) - Prescaler Register
	$2485_33A2	l1t_tb_pmr	L1T - TB - 1/4 bit Cnt Prescaler Modulus Register
	$2485_33A4	l1t_tb_rc	L1T - TB - Reference 1/4 bit Count
	$2485_33A6	l1t_tb_rcm	L1T - TB - Reference 1/4 bit Count Modulus
	$2485_33A8	l1t_tb_mc	L1T - TB - Matching 1/4bit Count
	$2485_33AA	l1t_tb_cfc	L1T - TB - Channel Frame Counter
	$2485_33AC	l1t_tb_rfc	L1T - TB - Reference Frame Counter
	$2485_33AE	l1t_tb_rfnms	L1T - TB - Reference Abs. Frame Number MS word
	$2485_33B0	l1t_tb_rfnls	L1T - TB - Reference Abs. Frame Number LS word
	$2485_33B2	l1t_tb_rfnmms	L1T - TB - Reference Abs. Frame Number Modulus, MS word
	$2485_33B4	l1t_tb_rfnmls	L1T - TB - Reference Abs. Frame Number Modulus, LS word
	$2485_33B6	l1t_tb_cfnms	L1T - TB - Channel Frame Number, MS word
	$2485_33B8	l1t_tb_cfnls	L1T - TB - Channel Frame Number, LS word
	$2485_33BA	l1t_tb_bcft	L1T - TB - 1/4 Bit Count Frame Tick
	$2485_33BC	l1t_tb_pitc	L1T - TB - Periodic Interrupt Time Count
	$2485_33BE	l1t_tb_pitm	L1T - TB - Periodic Interrupt Time Modulus
	$2485_33C0	l1t_tb_bco	L1T - TB - 1/4 Bit Count Offset
	$2485_33C2	l1t_qbco_utf	L1T - QBCO Update Time - Frame
	$2485_33C4	l1t_qbco_utbc	L1T - QBCO Update Time - 1/4 Bit Count
	$2485_33C6	l1t_qbco_ucr0	L1T - QBCO Update Control Register 0
	$2485_33C8	l1t_qbco_ucr1	L1T - QBCO Update Control Register 1

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2485_33CA	l1t_eg_f0tcr	L1T - Event Gen. Frame 0 Table Configuration Register
	$2485_33CC	l1t_eg_f1tcr	L1T - Event Gen. Frame 1 Table Configuration Register
	$2485_33CE	l1t_eg_nftcr	L1T - Event Gen. Next Frame Table Configuration Register
	$2485_33D0	l1t_eg_macr	L1T - Event Gen. Macro A Configuration Register
	$2485_33D2	l1t_eg_mbcr	L1T - Event Gen. Macro B Configuration Register
	$2485_33D4	l1t_eg_mccr	L1T - Event Gen. Macro C Configuration Register
	$2485_33D6	l1t_eg_ptcr	L1T - Event Gen.Parameter Table Configuration Register
	$2485_33D8	l1t_eg_ftptr	L1T - Event Gen.Frame Table PTR Register
	$2485_33DA	l1t_eg_ma_ptr0	L1T - Event Gen. Macro A Table PTR Register 0
	$2485_33DC	l1t_eg_ma_ptr1	L1T - Event Gen. Macro A Table PTR Register 1
	$2485_33DE	l1t_eg_vdr0_fd	L1T - Event Gen. Variable Delay Register 0 - Frame Delay
	$2485_33E0	l1t_eg_vdr0_qbc	L1T - Event Gen. Variable Delay Register 0 - QBC Delay
	$2485_33E2	l1t_eg_vdr1_fd	L1T - Event Gen. Variable Delay Register 1 - Frame Delay
	$2485_33E4	l1t_eg_vdr1_qbc	L1T - Event Gen. Variable Delay Register 1 - QBC Delay
	$2485_33E6	l1t_eg_vdr2_fd	L1T - Event Gen. Variable Delay Register 2 - Frame Delay
	$2485_33E8	l1t_eg_vdr2_qbc	L1T - Event Gen. Variable Delay Register 2 - QBC Delay
	$2485_33EA	l1t_eg_vdr3_fd	L1T - Event Gen. Variable Delay Register 3 - Frame Delay
	$2485_33EC	l1t_eg_vdr3_qbc	L1T - Event Gen. Variable Delay Register 3 - QBC Delay
	$2485_33EE	l1t_eg_vlc_r0	L1T - Event Gen. Variable Loop Count Register 0
	$2485_33F0	l1t_eg_vlc_r1	L1T - Event Gen. Variable Loop Count Register 1
	$2485_33F2	l1t_eg_vlc_r2	L1T - Event Gen. Variable Loop Count Register 2
	$2485_33F4	l1t_eg_vlc_r3	L1T - Event Gen. Variable Loop Count Register 3

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2485_33F6	l1t_pc_ucr	L1T - Pin Control Unit Configuration Register
	$2485_33F8	l1t_pc_udr	L1T - Pin Control Unit Default Register
	$2485_33FA	l1t_pc_usr	L1T - Pin Control Unit Status Register
	$2485_33FC	l1t_pc_utscr	L1T - Pin Control Unit Tri-State Configuration Register
	$2485_33FE	l1t_pc_updimr	L1T - Pin Control Unit Pin Driver Internal Mask Register
	$2485_3400	l1t_eg_mdr	L1T - Event Gen. Macro Disable Register
	$2485_3402		L1T - Reserved Through
	$2485_3FFF		L1T - Reserved
DMAC	**$2485_4000**	**dmac_registers_base_address**	**Display Memory Access Controller (DMAC)**
	$2485_4000	dmac_addrh	DMAC ADDRESS_H [31:16]
	$2485_4002	dmac_addrl	DMAC ADDRESS_H [15:0]
	$2485_4004	dmac_dbsizh	DMAC DATABUFSIZ_H[31:16]
	$2485_4006	dmac_dbsizl	DMAC DATABUFSIZ_L [15:0]
	$2485_4008	dmac_comba_seadrh	DMAC COMBASEADR_H[31:16]
	$2485_400A	dmac_comba_seadrl	DMAC COMBASEADR_L[15:0]
	$2485_400C	dmac_cmdbuf_szh	DMAC- Command Buffer Size [31:16]
	$2485_400E	dmac_cmdbuf_szl	DMAC- Command Buffer Size [15:0]
	$2485_4010	dmac_cmdstr_sz	DMAC Command String Size
	$2485_4012	dmac_fifo_cfg	DMAC FIFO Configuration Register
	$2485_4014	dmac_lcd_cfg	DMAC LCD CONFIG
	$2485_4016	dmac_lcdtrf_cfg	DMAC LCD TRANSFER CONFIG
	$2485_4018	dmac_ctlstat	DMAC CTLSTAT
	$2485_401A	dmac_clkcfg	DMAC CLKCFG
	$2485_401C	dmac_lcd_wrdata	LCDWRDATA

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
OWIRE	**$2485_6XXX**	owire_base_address	**One Wire Interface Module**
	$2485_6000	owire_control	Owire - Control/Status Register
	$2485_6002	owire_clock_divider	Owire - Clock Divider Register
	$2485_6004	owire_reset	Owire - Reset Register
	$2485_6008		Owire - Reserved through
	$2485_6FFF		Owire - Reserved
UART2	**$2485_7XXX**	uart2_base_address	**Universal Asynchronous RX TX Unit 2 (UART2)**
	$2485_7000	uart2_urx_start	UART2 RX Register_2 (URX) - Start
	$2485_703E	uart2_urx_end	UART2 RX Register_2 (URX) - End
	$2485_7040	uart2_utx_start	UART2 TX Register_2 (UTX) - Start
	$2485_707E	uart2_utx_end	UART2 TX Register_2 (UTX) - End
	$2485_7080	uart2_ucr1	UART2 Control Register 1 - UCR1[15:0]
	$2485_7082	uart2_ucr2	UART2 Control Register 2 - UCR2[15:0]
	$2485_7084	uart2_ucr3	UART2 Control Register 3 - UCR3[15:0]
	$2485_7086	uart2_ucr4	UART2 Control Register 4- UCR4[15:0]
	$2485_7088	uart2_ufcr	UART2 FIFO Control Register - UFCR[15:0]
	$2485_708A	uart2_usr1	UART2 Status Register 1 - USR1[15:0]
	$2485_708C	uart2_usr2	UART2 Status Register 2 - USR2[15:0]
	$2485_708E	uart2_uesc	UART2 Escape Character Register - UESC[15:0]
	$2485_7090	uart2_utim	UART2 Escape Timer Register - UTIM
	$2485_7092	uart2_ubir	UART2 BRM INC Register - UBIR
	$2485_7094	uart2_ubmr	UART2 BRM MOD Register - UBMR
	$2485_7096	uart2_ubrc	UART2 Baud Rate Count Register - UBRC
	$2485_7098	uart2_bipr1	UART2 BRM INC Preset Register 1 - BIPR1
	$2485_709A	uart2_bmpr1	UART2 BRM MOD Preset Register 1 - BMPR1
	$2485_709C	uart2_bipr2	UART2 BRM INC Preset Register 2 - BIPR2
	$2485_709E	uart2_bmpr2	UART2 BRM MOD Preset Register 2 - BMPR2

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2485_70A0	uart2_bipr3	UART2 BRM INC Preset Register 3 - BIPR3
	$2485_70A2	uart2_bmpr3	UART2 BRM MOD Preset Register 3 - BMPR3
	$2485_70A4	uart2_bipr4	UART2 BRM INC Preset Register 4 - BIPR4
	$2485_70A6	uart2_bmpr4	UART2 BRM MOD Preset Register 4 - BMPR4
	$2485_70A8	uart2_uts	UART2 Test Register - UTS
	$2485_70AA		UART2 Reserved through
	$2485_7FFF		UART2 Reserved
HAC	**$2485_8XXX**	hac_base_address	**Hash Acceleration Module**
	$2485_8000	hac_ctl_stat	HAC -CTL_STAT
	$2485_8004	hac_start_addr	HAC - START_ADDR[31:0]
	$2485_8008	hac_block_count	HAC - BLOCK_COUNT[22:0]
	$2485_800C	hac_hsh4	HAC - HSH[159:128]
	$2485_8010	hac_hsh3	HAC - HSH[127:96]
	$2485_8014	hac_hsh2	HAC - HSH[95:64]
	$2485_8018	hac_hsh1	HAC - HSH[63:32]
	$2485_801C	hac_hsh0	HAC - HSH[31:0]
	$2485_8020		HAC - Reserved through
	$2485_8FFF		HAC - Reserved
GEM	**$2485_9000**	**gem_registers_base_address**	**GPRS Encryption Module**
	$2485_9000	gem_ctl	GPRS - Control Register (CTL)
	$2485_9004	gem_stat	GPRS - Key Status Register 1 (STAT)
	$2485_9008	gem_kc1	GPRS - Key Constant Register 1 (Kc1)
	$2485_900C	gem_kc0	GPRS - Key Constant Register 0 (Kc0)
	$2485_9010	gem_input	GPRS - Input Register (INPUT)
	$2485_9014	gem_frame_start	GPRS - Frame Start Register (FRAME START)
	$2485_9018	gem_header_length	GPRS - Header Length Register (HEADER LENGTH)

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2485_901C	gem_frame_length	GPRS - Frame Length Register (FRAME LENGTH)
	$2485_9020	gem_access_delay	GPRS - Access Delay Register (ACCESS DELAY)
	$2485_9024	gem_ctl2	GPRS - Control Register 2 (CTL2)
	$2485_9028	gem_write_frame_start	GPRS - Write Frame Start Register (WRITE FRAME START)
	$2485_902C		GPRS - Reserved through
	$2485_9FFF		GPRS - Reserved
MDI	**$2485_B000**	**mdi_registers_base_address**	**MCU/DSP Interface Module (MDI)**
	$2485_B000		MDI - Reserved through
	$2485_BFF0		MDI - Reserved
	$2485_BFF2	mdi_mcvr	MDI - Command Vector Register MCVR[15:0]
	$2485_BFF4	mdi_mcr	MDI - Control Register MCR[15:0]
	$2485_BFF6	mdi_msr	MDI - Status Register MSR[15:0]
	$2485_BFF8	mdi_mtr1	MDI - TX Register 1 MTR1[15:0]
	$2485_BFFA	mdi_mtr0	MDI - TX Register 0 MTR0[15:0]
	$2485_BFFC	mdi_mrr1	MDI - RX Register 1 MRR1[15:0]
	$2485_BFFE	mdi_mrr0	MDI - RX Register 0 MRR0[15:0]
MDI	**$2485_C000**		**MDI - Shared Memory Space**
	$2485_C800	mdi_memory_base_address	MDI Shared Memory Start
	$2485_FFFF	mdi_memory_end	MDI Shared Memory End
AHB	**$2800_0000**		**AHB- Arm High Performance Bus**
AEIM	**$2888_0000**	**eim_registers_base_address**	**MCU External Interface Module**
	$2888_0000	eim_config	AEIM Configuration Register
	$2888_0004		AEIM - Reserved through
	$2888_000F		AEIM - Reserved

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2888_0010	eim_cs0_pcfg	AEIM - CS0 Primary Configuration Register
	$2888_0014	eim_cs0_scfg	AEIM - CS0 Secondary Configuration Register
	$2888_0018	eim_cs0_wsctl	EIM - CS0 Wait State Control Register
	$2888_001C	eim_cs0_bcctl	EIM - CS0 Burst Clock Control Register
	$2888_0020	eim_cs1_pcfg	EIM - CS1 Primary Configuration Register
	$2888_0024	eim_cs1_scfg	EIM - CS1 Secondary Configuration Register
	$2888_0028	eim_cs1_wsctl	EIM - CS1 Wait State Control Register
	$2888_002C	eim_cs1_bcctl	EIM - CS1 Burst Clock Control Register
	$2888_0030	eim_cs2_pcfg	EIM - CS2 Primary Configuration Register
	$2888_0034	eim_cs2_scfg	EIM - CS2 Secondary Configuration Register
	$2888_0038	eim_cs2_wsctl	EIM - CS2 Wait State Control Register
	$2888_003C	eim_cs2_bcctl	EIM - CS2 Burst Clock Control Register
	$2888_0040	eim_cs3_pcfg	EIM - CS3 Primary Configuration Register
	$2888_0044	eim_cs3_scfg	EIM - CS3 Secondary Configuration Register
	$2888_0048	eim_cs3_wsctl	EIM - CS3 Wait State Control Register
	$2888_004C	eim_cs3_bcctl	EIM - CS3 Burst Clock Control Register
	$2888_0050	eim_cs4_pcfg	EIM - CS4 Primary Configuration Register
	$2888_0054	eim_cs4_scfg	EIM - CS4 Secondary Configuration Register
	$2888_0058	eim_cs4_wsctl	EIM - CS4 Wait State Control Register
	$2888_005C	eim_cs4_bcctl	EIM - CS4 Burst Clock Control Register
	$2888_0060	eim_cs5_pcfg	EIM - CS5 Primary Configuration Register
	$2888_0064	eim_cs5_scfg	EIM - CS5 Secondary Configuration Register
	$2888_0068	eim_cs5_wsctl	EIM - CS5 Wait State Control Register
	$2888_006C	eim_cs5_bcctl	EIM - CS5 Burst Clock Control Register
	$2888_0070		EIM - Reserved through
	$2888_0FFF		EIM - Reserved through
AITC	**$2989_0000**	**itc_registers_base_address**	**MCU Interrupt Controller (AITC)**

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2989_0000	itc_intcntl	Interrupt Control Register
	$2989_0004	itc_nimask	Normal Interrupt Mask Register
	$2989_0008	itc_intennum	Interrupt Enable Number Register
	$2989_000C	itc_intdisnum	Interrupt Disable Number Register
	$2989_0010	itc_intenableh	Interrupt Enable Register High
	$2989_0014	itc_intenablel	Interrupt Enable Register Low
	$2989_0018	itc_inttypeh	Interrupt Type Register High
	$2989_001C	itc_inttypel	Normal Interrupt Pending Register Low
	$2989_0020	itc_niprior7	Normal Interrupt Priority Level Register 7
	$2989_0024	itc_niprior6	Normal Interrupt Priority Level Register 6
	$2989_0028	itc_niprior5	Normal Interrupt Priority Level Register 5
	$2989_002c	itc_niprior4	Normal Interrupt Priority Level Register 4
	$2989_0030	itc_niprior3	Normal Interrupt Priority Level Register 3
	$2989_0034	itc_niprior2	Normal Interrupt Priority Level Register 2
	$2989_0038	itc_niprior1	Normal Interrupt Priority Level Register 1
	$2989_003C	itc_niprior0	Normal Interrupt Priority Level Register 0
	$2989_0040	itc_nivecsr	Normal Interrupt Vector and Status Register
	$2989_0044	itc_fivecsr	Fast Interrupt Vector and Status Register
	$2989_0048	itc_intsrch	Interrupt Source Register High
	$2989_004C	itc_intsrcl	Interrupt Source Register Low
	$2989_0050	itc_intfrch	Interrupt Force Register High
	$2989_0054	itc_intfrcl	Interrupt Force Register
	$2989_0058	itc_nipndh	Normal Interrupt Pending Register High
	$2989_005C	itc_nipndl	Normal Interrupt Pending Register Low
	$2989_0060	itc_fipndh	Fast Interrupt Pending Register High
	$2989_0064	itc_fipndl	Fast Interrupt Pending Register Low
	$2989_0068		AITC - Reserved through
	$2989_0FFF		AITC - Reserved

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
AWPT	**$2A8A_0XXX**	**wpt_registers_base_address**	**Watch Point Module**
	$2A8A_00B4	awpt_d15	WPT - WPTD15
	$2A8A_00B8	awpt_d14	WPT - WPTD14
	$2A8A_00Bc	awpt_d13	WPT - WPTD13
	$2A8A_00C0	awpt_d12	WPT - WPTD12
	$2A8A_00C4	awpt_d11	WPT - WPTD11
	$2A8A_00C8	awpt_d10	WPT - WPTD10
	$2A8A_00CC	awpt_d9	WPT - WPTD9
	$2A8A_00D0	awpt_d8	WPT - WPTD8
	$2A8A_00D4	awpt_d7	WPT - WPTD7
	$2A8A_00D8	awpt_d6	WPT - WPTD6
	$2A8A_00DC	awpt_d5	WPT - WPTD5
	$2A8A_00E0	awpt_d4	WPT - WPTD4
	$2A8A_00E4	awpt_d3	WPT - WPTD3
	$2A8A_00E8	awpt_d2	WPT - WPTD2
	$2A8A_00EC	awpt_d1	WPT - WPTD1
	$2A8A_00F0	awpt_d0	WPT - WPTD0
	$2A8A_00F4	awpt_cntl	WPT-WPTCNTL
	$2A8A_00F8	awpt_enh	AWPT-ENABLE[63:32]
	$2A8A_00FC	awpt_enl	AWPT-ENABLE[31:0]
		awpt_a[0:63]	AWPT-ADDRn
	$2A8A_0200	awpt_brpt	AWPT- AWPTBRPT
	$2A8A_0204	awpt_basea	AWPT-BASEA
	$2A8A_0208	awpt_sr	AWPT- AWPTSR
	$2A8A_020C		AWPT - Reserved Through
	$2A8A_0FFF		AWPT - Reserved
MCTL	**$2B8B_0XXX**		**MCTL- Memory Control Interface**

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2B8B_0000	mctl_registers_base_address	MCTL- Base Address
	$2B8B_0000	mctl_mfslram	MCTL- Memory Fuse Sense
	$2B8B_0004		MCTL - Reserved
	$2B8B_0008	mctl_mftlram	MCTL- Memory Fuse Test SRAM
	$2B8B_000C		MCTL - Reserved
	$2B8B_0010	mctl_mctltm	MCTL- Memory Test Mode
	$2B8B_0014	mctl_mctphg	MCTL- Memory Test Precharge
	$2B8B_0018	mctl_wvallrm	MCTL- Write Val Setting
	$2B8B_001C		MCTL - Reserved
	$2B8B_0020	mctl_rvallrm	MCTL- Read Val Setting (sram)
	$2B8B_0024		MCTL - Reserved
	$2B8B_0028	mctl_rvalrom	MCTL- Read Val Setting (ROM)
	$2B8B_002C	mctl_rval2rom	MCTL- Read Val 2 (ROM)
	$2B8B_0030		MCTL - Reserved
	$2B8B_0034		MCTL - Reserved
	$2B8B_0038		MCTL - Reserved
	$2B8B_003C		MCTL - Reserved
	$2B8B_0040	mctl_mfspram1	MCTL- Fuse Sense 1 PRAM
	$2B8B_0044	mctl_mfspram2	MCTL- Fuse Sense 2 PRAM
	$2B8B_0048	mctl_mfspram3	MCTL- Fuse Sense 3 PRAM
	$2B8B_004C	mctl_mfspram4	MCTL- Fuse Sense 4 PRAM
	$2B8B_0050	mctl_mtsclram	MCTL- Memory Test Scan (sram)
MSU	**$2C8C_0XXX**	**msu_base_address**	**Memory Separation Unit (MSU)**
	$2C8C_0000	msu_idr	MSU - IDENTIFICATION REGISTER (MIDR)
	$2C8C_0004	msu_ser	MSU - SET ENABLE REGISTER (MSER)
	$2C8C_0008	msu_cer	MSU - CLEAR ENABLE REGISTER (MCER)
	$2C8C_000C	msu_far	MSU - FAULT ADDRESS REGISTER (MFAR)

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2C8C_0010	msu_fsr	MSU - FAULT STATUS REGISTER (MFSR)
	$2C8C_0014		MSU - Reserved through
	$2C8C_002F		MSU - Reserved
	$2C8C_0030	msu_aer	MSU - AREA ENABLE REGISTER (MAER)
	$2C8C_0034	msu_a0cr	MSU - AREA 0 CONTROL REGISTER (MA0CR)
	$2C8C_0038	msu_a1cr	MSU - AREA 1 CONTROL REGISTER (MA1CR)
	$2C8C_003C	msu_a2cr	MSU - AREA 2 CONTROL REGISTER (MA2CR)
	$2C8C_0040	msu_a3cr	MSU - AREA 3 CONTROL REGISTER (MA3CR)
	$2C8C_0044	msu_a4cr	MSU - AREA 4 CONTROL REGISTER (MA4CR)
	$2C8C_0048	msu_a5cr	MSU - AREA 5 CONTROL REGISTER (MA5CR)
	$2C8C_004C	msu_a6cr	MSU - AREA 6 CONTROL REGISTER (MA6CR)
	$2C8C_0050	msu_a7cr	MSU - AREA 7 CONTROL REGISTER (MA7CR)
	$2C8C_0054		MSU - Reserved through
	$2C8C_00AF		MSU - Reserved
SCC MON	**$2D8D_XXXX**	**scc_mon_base_address**	**SCC MON - Security Controller Module Monitor**
	$2D8D_0000	scc_mon_status	**SCC** - MONITOR STATUS
	$2D8D_0004	scc_mon_command	**SCC** - MONITOR COMMAND
	$2D8D_0008	scc_mon_seqstart	**SCC** - MONITOR SEQUENCE START
	$2D8D_000C	scc_mon_seqend	**SCC** - MONITOR SEQUENCE END
	$2D8D_0010	scc_mon_seqchk	**SCC** - MONITOR SEQUENCE CHECK
	$2D8D_0014	scc_mon_bitcnt	**SCC** - MONITOR BIT COUNT
	$2D8D_0018	scc_mon_incsize	**SCC** - MONITOR INCREMENT SIZE
	$2D8D_001C	scc_mon_bbdec	**SCC** - MONITOR BB DECODE
	$2D8D_0020	scc_mon_compsize	**SCC** - MONITOR COMPARE SIZE
	$2D8D_0024	scc_mon_ptchk	**SCC** - MONITOR PT CHECK
	$2D8D_0028	scc_mon_ctchk	**SCC** - MONITOR CT CHECK

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2D8D_002C	scc_mon_timeriv	**SCC** - MONITOR TIMER IV
	$2D8D_0030	scc_mon_timerctl	**SCC** - MONITOR TIMER CTL
	$2D8D_0034	scc_mon_ddstatus	**SCC** - MONITOR DD STATUS
	$2D8D_0038	scc_mon_timer	**SCC** - MONITOR TIMER
	$2D8D_003C		**SCC** - Reserved through
	$2D8D_0FFF		**SCC** - Reserved
SCC MEM	**$2E8E_XXXX**	**scc_mem_base_address**	**SCC MEM - Security Controller Module Memory**
	$2E8E_0000	**scc_mem_red_start**	**SCC** - RED_START (SRS)
	$2E8E_0004	**scc_mem_black_start**	**SCC** - BLACK_START (SBS)
	$2E8E_0008	**scc_mem_length**	**SCC** - LENGTH REGISTER (SLR)
	$2E8E_000C	**scc_mem_control**	**SCC** - CONTROL REGISTER (SCR)
	$2E8E_0010	**scc_mem_status**	**SCC** - STATUS REGISTER (SSR)
	$2E8E_0014	**scc_mem_error**	**SCC** - ERROR REGISTER (SER)
	$2E8E_0018	**scc_mem_int_cont**	**SCC** - INTERRUPT MASK REGISTER (SIMR)
	$2E8E_001C	**scc_mem_config**	**SCC** - CONFIGURATION REGISTER (SCON)
	$2E8E_0020	**scc_mem_init_vector_0**	**SCC** - INTERRUPT VECTOR 0 (SIV0)
	$2E8E_0024	**scc_mem_init_vector_1**	**SCC** - INTERRUPT VECTOR 1 (SIV1)
	$2E8E_0028		**SCC** - Reserved through
	$2E8E_03FF		**SCC** - Reserved
	$2E8E_0400	**scc_mem_red_mem_start**	**SCC** - RED_MEM through
	$2E8E_07FF	**scc_mem_red_mem_end**	**SCC** - RED_MEM

Table 6-13. MCU Memory Space (MOD = HIGH on reset)

Used With	MCU Address (Hex)	REGISTER EQUATES NAME	Description
	$2E8E_0800	**scc_mem_black _mem_start**	**SCC** - BLACK_MEM through
	$2E8E_0BFF	**scc_mem_black _mem_end**	**SCC** - BLACK_MEM
	$2E8E_0C00		**SCC** - Reserved through
	$2E8E_0FFF		**SCC** - Reserved
END			

1. USB Endpoint addresses are affected by the USBCPUCR MEMMAP bit. See Table 21-16 on page 21-46 in the Chapter 6, “Chip Configuration and Memory Maps (MEMMAP),” for details on MEMMAP operation.

6.3.6 MCU Peripheral Interrupts

Information on MCU peripheral interfaces and source clock(s) can be found in Table 3-1 on page 3-5.

In Table 6-14 the order number 63 represents the highest priority interrupt and the order number 0 represents the lowest priority interrupt.

Table 6-14. MCU Interrupt Sources

Order	Interrupt Source	Order	Interrupt Source
63	Unused	31	GPIO Port E
62	GPIO Port A	30	External 4
61	RTR	29	GEM
60	External 0	28	USB SUSPEND, RESUME, RESET
59	UART1 RX	27	UART1 COMMOM
58	RTCA_INT	26	DMAC_IRQ
57	USB RX	25	UART1 DTR
56	USB TX	24	UART1 RTS
55	GPIO Port B	23	Unused
54	External 1	22	External 5
53	UART1 TX	21	UART2 COMMON
52	UART2 RX	20	Unused
51	SIM RX/TX	19	UART2 DTR

Table 6-14. MCU Interrupt Sources (Continued)

Order	Interrupt Source	Order	Interrupt Source
50	UART2 TX	18	UART2 RTS
49	A2DIGL	17	MDI General 0
48	DSM	16	Unused
47	L1Timer MCU_INT0	15	EGPT
46	L1Timer Channel Int	14	EPIT
45	L1Timer RFCI	13	External 6
44	L1Timer RFI	12	SIM General
43	L1Timer MCU_INT1	11	RXSDG
42	MDI RX/TX 0 OR'd	10	Unused
41	GPIO Port C	9	KeyPad Port
40	Unused	8	Unused
39	External 2	7	MQSPI Port 2
38	MDI RX/TX 1 OR'd	6	MDI General 1
37	GPIO Port D	5	External 7
36	Unused	4	MDI Or'd
35	External 3	3	HAC
34	MQSPI Port 1	2	SCC Secure RAM
33	Unused	1	SCC Security Monitor
32	Unused	0	Unused

6.4 Mixed-Signal Modules Interconnections

This section illustrates the top-level interconnects for the mixed-signal modules. Figure 6-11 shows the connections between the RXAFE, RxCPROC, DCADAPT, RxSDG, PAC, TX/TRSYNT, and GPADC modules. All of these mixed signal modules accept control inputs from the A2DIGL block. The RXAFE sigma-delta output is input to the RxCPROC, DCADAPT, and RxSDG modules. The RxSDG module feeds back a dither amplitude value to the RXAFE, and a overflow detect flag to the AITC. The DCADAPT block sends SPI signals intended for the Algae IC to the A2DIGL block, where these signals are muxed with SPI signals from the MQSPI. The RxCPROC outputs received data in SSI (synchronous serial interface) format to the A2DIGL where these signals are muxed and routed to the BBP. The TRSYNT outputs divided-down charge pump signals to the GPADC so that these signals can be sampled; the GPADC outputs these and other A/D samples to the A2DIGL where they are accessible through the MCU peripheral bus. The TX block accepts serial data from the BBP (these signals are routed through the A2DIGL for muxing). The PAC outputs several signals to the A2DIGL which are accessible through the MCU peripheral bus. Each mixed signal module must be configured through the appropriate register prior to use.

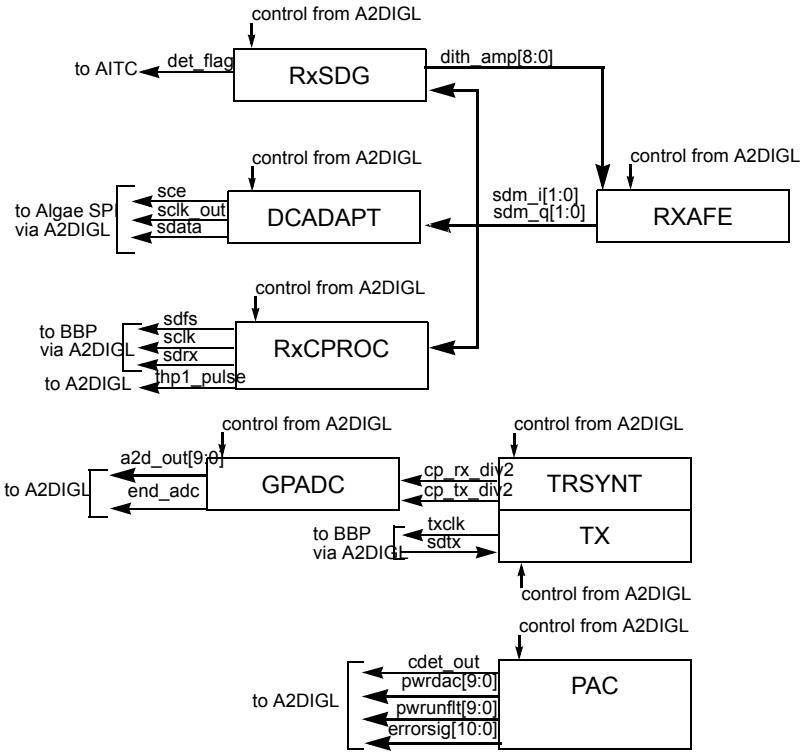


Figure 6-11. Mixed Signal Modules internal connections

The next three figures focus on the connections between the mixed-signal modules and the A2DIGL, MQSPI, L1T, and BBP. These figures illustrate the primary connections as well as two alternate connection configurations: a ‘control’ configuration which allows external control and data access to the mixed signal modules (illustrated by blue dashed lines in the figures below), and a ‘bypass’ configuration which allows the mixed signal modules to be bypassed (illustrated by red dotted lines in the figures below). In addition to the figures text below, see also Section 36.5.1.1, “Support for bypass and control modes,” on page 36-24 in the A2DIGL chapter.

The ‘control’ configuration is provided to facilitate test of the mixed signal modules without needing to program the MCU and DSP cores. This configuration is characterized as follows:

- The mixed-signal modules can be programmed through an external SPI interface
- The timing control signals which normally are provided by the L1T will be provided as inputs from external pins
- The SSI signals normally routed internally to the BBP/glue logic will be brought out to pins

Similarly, to facilitate development in case some of the mixed signal modules are not functional in the 1st silicon, the ‘bypass’ configuration allows for off-chip interface to an external IC such as the Magic-LV. This bypass mode is characterized as follows:

- Internally routed MQSPI signals will be brought out to pins
- Internally routed L1T signals will brought out to pins
- The BBP/glue logic signals will be brought out to pins

Figure 6-12 shows the connections between the MQSPI and A2DIGL modules; this interface provides most of the programming control for the mixed signal modules. The SPI2 interface (QSCKB, MOSIB, MISOB) is used to communicate with the A2DIGL and Algae; SPI_CS4 is the A2DIGL chip select;

SPI_CS5 is the Algae chip select. The Algae SPI signals are also muxed inside the A2DIGL with signals from DCADAPT. See Chapter 36, “Mixed-Signal Control Interface (A2DIGL),” for more information on this mux.

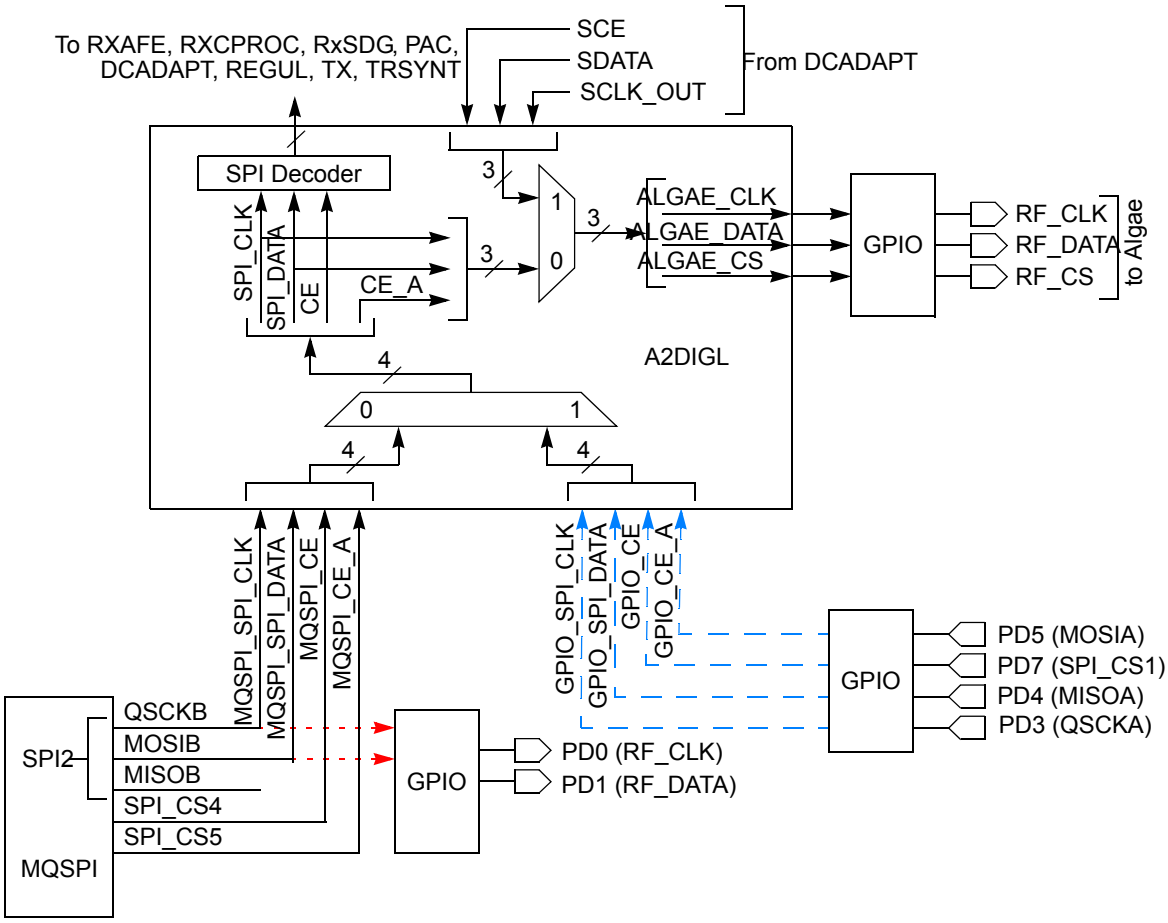


Figure 6-12. MQSPI/A2DIGL Internal Connections

The next figure, Figure 6-13, shows the connections between the L1T and A2DIGL modules; this interface provides timing control for the mixed signal modules. The L1T TOUT0 is used for the RX_ACQ signal (for receive timeslots), and the L1T TOUT1 is used for the DMCS signal (for transmit timeslots).

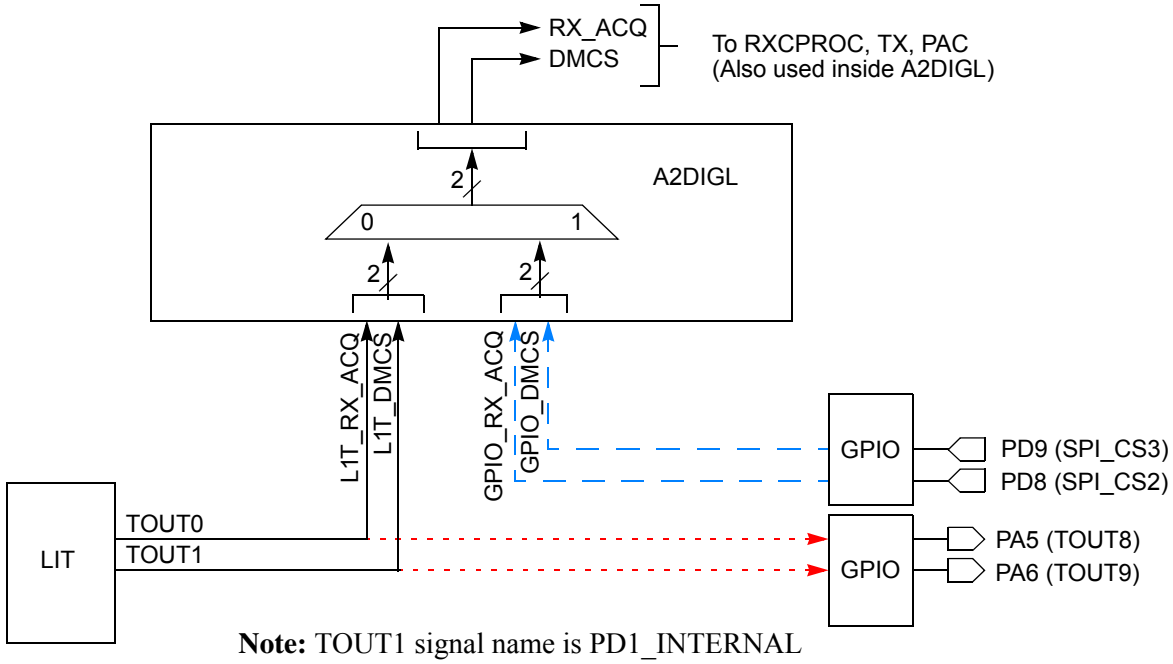


Figure 6-13. L1T/A2DIGL Internal Connections

The last figure, Figure 6-14, shows the connections between the RxCPROC, TX, BBP, and A2DIGL modules; this is the synchronous serial (SSI) interface for receive and transmit data.

NOTE:BBP Restrictions

For proper operation with the TX and RXCPROC, the BBP should be configured for asynchronous operation with external clocks and frame syncs. In this case, SC0B will be used as the receive clock, and SC1B will be used as the receive frame sync.

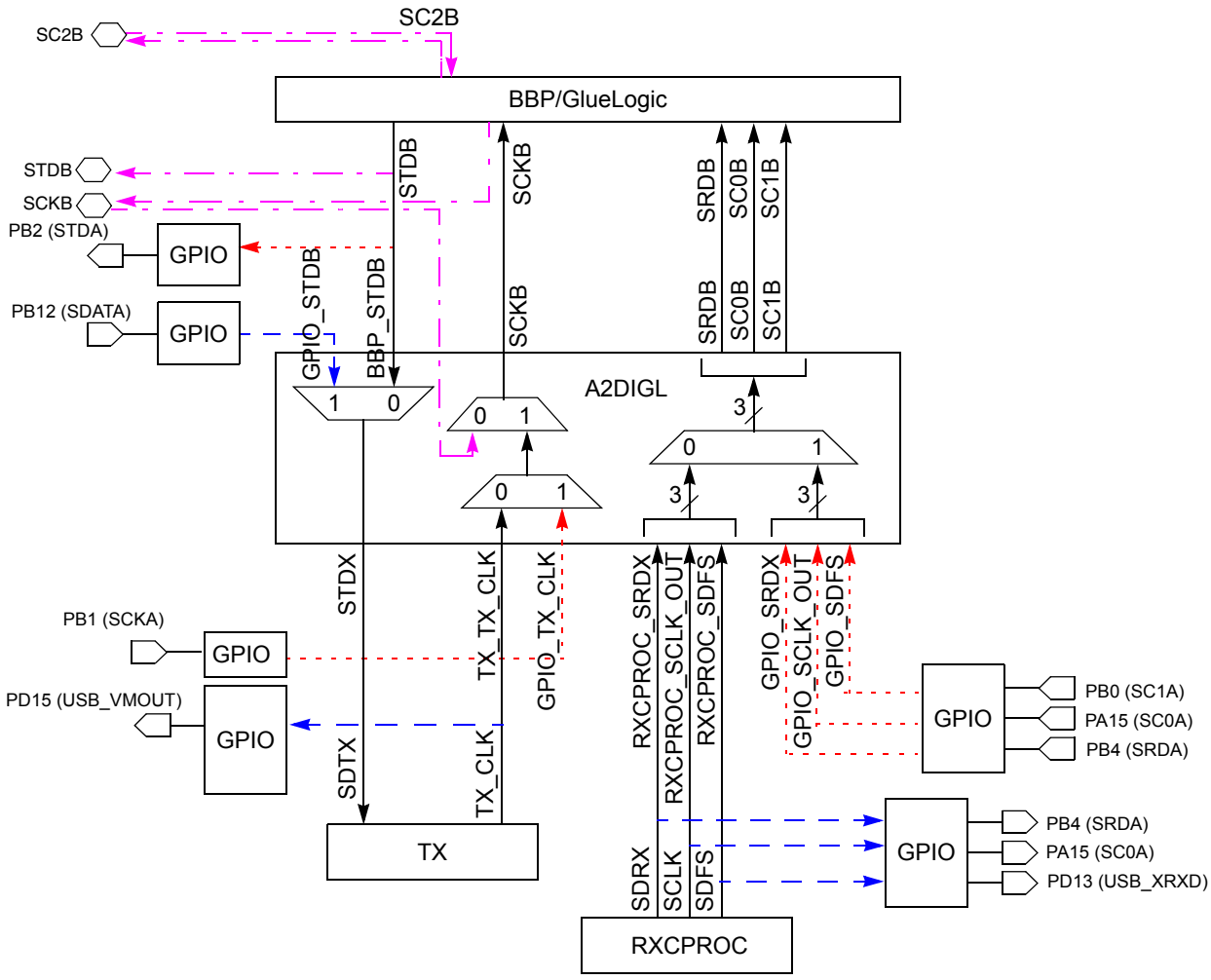


Figure 6-14. TX/RxCPROC/A2DIGL/BBP internal connections

6.5 MAGIC Serial Communication Protocol Enable on BBP pins

The reason for the addition of this serial protocol is that the existing BBP does not support “gated-clock” mode which is required by the TX module. The connection between the Neptune’s BBP and the TX module (or an external MAGIC IC) is outlined in Figure 6-15. Register control bits are provided on Neptune for control of this functionality; the register is physically implemented in the A2DIGL Module. The glue logic and muxing is implemented in the BBP module.

In the BBP receive path the RxCPROC module pin (or the pins from an external MAGIC) can be connected directly to the BBP receive pins. In the transmit path the INTERNAL CLOCK (Figure 6-17) is a burst of 8 PAT_REF/2 clocks generated inside the Neptune and is used to supply the clock edges which are needed by the BBP state machine to wake-up from its reset state. The “BBP BIT CLOCK” is internally generated and is identical to the TX_CLK form TX_CLK first assertion, when DMCSG[1] is high, until

FRAME SYNC negation. Before this period the “BBP BIT CLOCK” is the fed with 8 edges of the INTERNAL CLOCK and after this period the “BBP BIT CLOCK” is the fed with 16 edges of the INTERNAL CLOCK. The internal circuitry assures a glitchless switch between these two clocks.

If the MAGEN bit in A2DIGL MUXCTL register is set (see Table 6-17), the transmit glue logic will generate a bit length late frame sync pulse to the BBP block synchronously rising with the rising edge of the first TX_CLK and remaining high for one TX_CLK period. It will again go high for one TX_CLK period on the 16th rising edge of the TX_CLK, and every 16 cycles subsequently until DMCSG is de-asserted. The SC2B pin is muxed with the internal glue logic frame sync. If the MAGEN bit is set, the glue logic generates the frame sync. If the MAGEN bit is cleared, the standard RSI communication protocol is enabled for the SC2B pin. Refer to Table 14-8 on page 14-19 for more information on the BBP protocol.

NOTE:

The MAGEN bit should be programmed to 0 for BBP’s sckb signal to be read as a GPIO input through the BBP’s PDRB register.

This glue logic allows the BBP to be used with the MAGIC interface by configuring for 16 bit transmit words, normal mode, and word length late frame sync. Figure shows the BBP glue logic and I/O muxing for the BBP module.

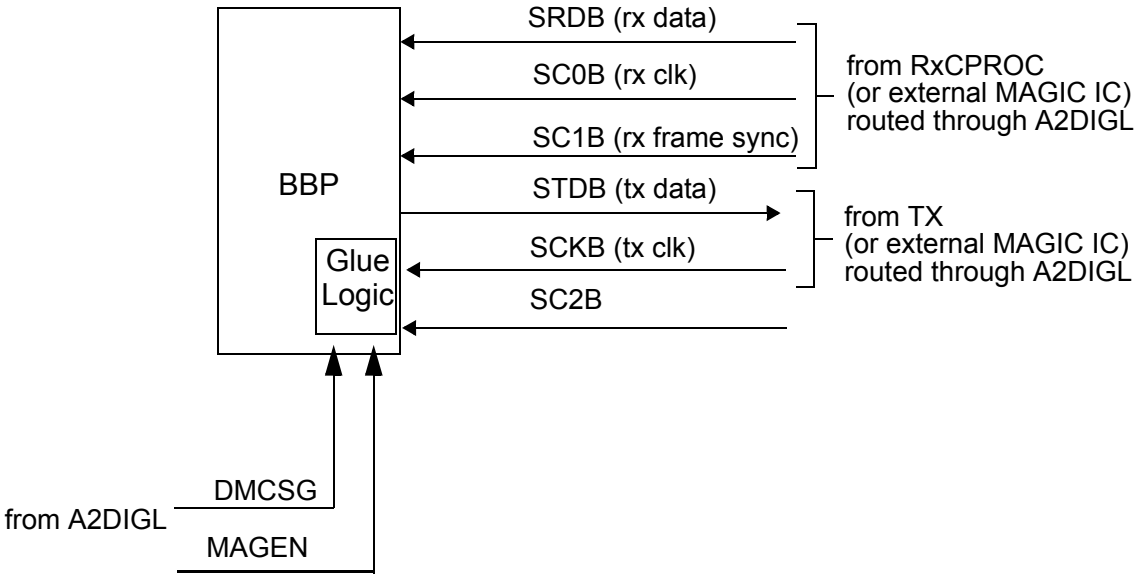


Figure 6-15. Connections for BBP Glue Logic

Typical waveforms for receive and transmit of the BBP are shown in Figure 6-16 and Figure 6-17.

1. DMCSG is a derived signal in the A2DIGL module. It is based on the L1T TOUT1 (DMCS) signal. SeeChapter 36, “Mixed-Signal Control Interface (A2DIGL),” for more information on DMCSG.

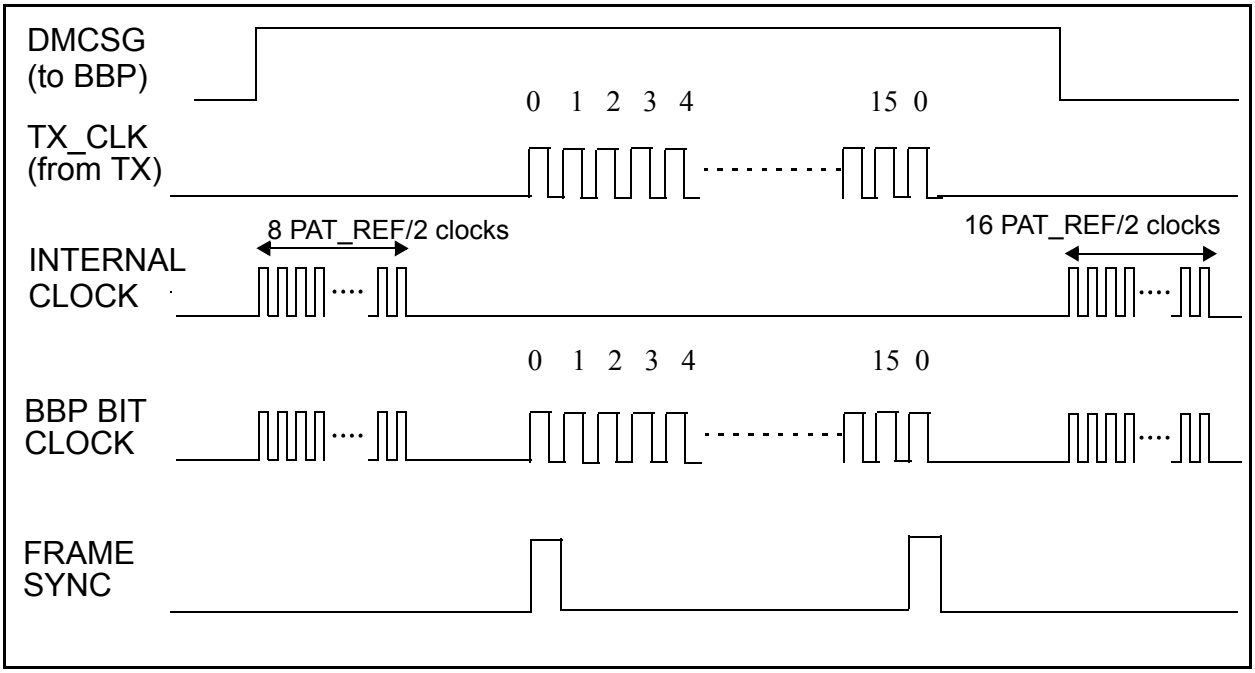


Figure 6-16. Typical Waveforms For BBP Transmit - TX Receive

At the BBP receive path the RX CLK will supply the needed clock edges for the BBP state machine to wake-up from its reset state, before the first frame sync, therefore no special logic is needed.

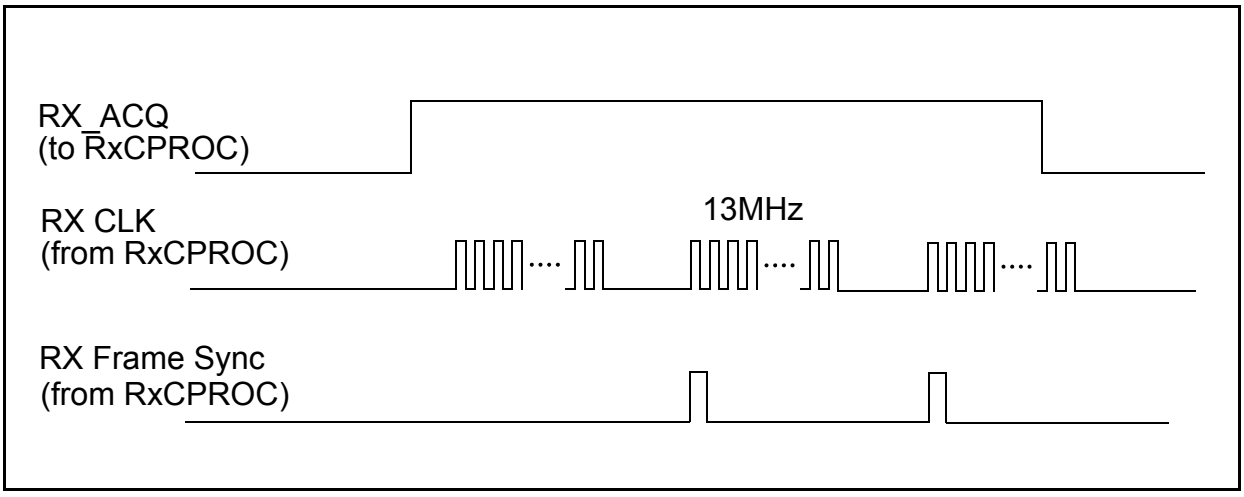


Figure 6-17. Typical Waveforms For BBP Receive - RxCPROC Transmit

6.6 DSP Debug Configuration

Refer to the DSP Debug and GPIO sections for a more detailed description of the debug configuration. If DSP Trace is not enabled via the DSP 56600 OMR register, the GPIO outputs are tristated to save power.

6.7 Debug Event Assertion Timing

In order to put the DSP core into DEBUG state, the $\overline{\text{DSP\_DE}}$ pin or an internal debug event request must be asserted for at least 1.5 DSP clocks cycles (a shorter request will be ignored).

In order to put the MCU core into DEBUG state, the $\overline{\text{MCU\_DE}}$ pin or an internal debug request should be asserted for at least 6 MCU clock cycles.

6.8 IO Pad Drive Programming

Software programmable I/O pad drive strength is available on some of the Neptune pads. The programmable pads are grouped into eleven groups, described in Table 6-15. Each of these groups can be controlled by one of the General Purpose Output (GPO) bits in the EIM Configuration Register, Table 6-17.

At reset, the Neptune internal MCU Boot code reconfigures these pins as 8mA Drive by writing to the GPO bits of the EIM module. The programming model of the GPO bits in the EIM Configuration Register for the Neptune device is shown in Table 6-17. For a complete description of the EIM Configuration Register, refer to Table 48-9, AEIM Configuration Register Description in the EIM chapter.

In addition to the eleven pad groups that may be programmed to 4mA or 8mA Drive using GPO[0:9] and GPO[15], some of the pad groups may be programmed to a 16mA drive using GPO[14:10] and GPO[16] in the EIM Configuration Register. Table 6-15 lists the pads that are associated with each group that may be programmed to 16mA drive. Table 6-17 describes the programming model for the 16mA drive strength groups.

Table 6-15. Pad Drive Groups

Group	Pads in Drive Group	Pads in 16mA Drive Group	Control Bits	Default at Reset	Notes
1	Address[23:0]	Address[23:0], $\overline{\text{LBA}}$	GPO0:GPO10	4mA	
2	Data[31:0]	Data[31:0]	GPO1:GPO11	4mA	
3	$\overline{\text{CS}}$[4:0] CS5	$\overline{\text{CS}}$[4:0] CS5	GPO2:GPO12	4mA	
4	$\overline{\text{EB}}$[1:0] R/$\overline{\text{W}}$ $\overline{\text{OE}}$	$\overline{\text{EB}}$[1:0] R/$\overline{\text{W}}$ $\overline{\text{OE}}$	GPO3:GPO13	4mA	
5	QSCKA MOSIA		GPO4	4mA	Power Control IC SPI
6	CKOH CKO		GPO5	4 mA	
7	$\overline{\text{LBA}}$ BURSTCLK	BURSTCLK	GPO6:GPO14	4mA	
8	TOUT[12:7,5] RX_EN_OUT GSM*_DCS	TOUT[12:7,5] RX_EN_OUT GSM*_DCS	GPO7:GPO16	4 mA	
9	RF_CLK RF_DATA		GPO8	4mA	IF IC SPI
10	SAP: SCKA, STDA, SC2A BBP: SCKB, STDB, SC2B		GPO9	4 mA	
11	SIZ0 SIZ1 QSTAT0 QSTAT1 QSTAT2 QSTAT3 QSTAT4 MLB_TSCA MLB_TSCD		GPO15	4mA	Real Time Trace

NOTE:

Defaults are 4mA Drive, but the Neptune internal MCU Boot code reconfigures these pins as 8mA Drive by writing to the General Purpose Outputs of the EIM module.

NOTE:

The $\overline{\text{LBA}}$ pad is controlled by Group 7 for 4mA or 8mA drive strength. However, it is controlled by Group 1 for 16mA drive strength. Therefore it's Drive Strength can be a bit confusing. The following table attempts to clarify.

Table 6-16. LBA Drive Strength

Group 1	Group 7	$\overline{\text{LBA}}$ Drive Strength
4mA or 8mA	4mA or 8mA	4mA or 8mA
4mA or 8mA	16mA	4mA or 8mA
16mA	4mA or 8mA	16mA
16mA	16mA	16mA

EIM CONFIG — EIM Configuration Register — Addr $2888_0000

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
											GPO [17:12]					
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	GPO[11:0]												BCM	HDB	SHEN1	SHEN0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-17. GPO Bit Descriptions

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>Reserved
Bit 31:24</td><td>AEIM Configuration Register Reserved—
These bits are reserved for future expansion. They are read as zero and should be written with zero for upward compatibility.
N/A</td><td></td></tr>
<tr><td>GPIO18-19
Bit 22,23</td><td>Not Connected</td><td>N/A</td></tr>
<tr><td>GPO17
Bit 21</td><td>General Purpose Output 15 — This bit is an output only software controlled register that can be used at the chip integration level to provide desired EIM pin muxing, etc., or can be directly connected to chip outputs for general purpose use.

These bits are forced to zero at reset, and can be read or written by the CPU. Reads will return either the reset value, or the previously written value.</td><td>0 -The general purpose register output is driven low.
1 -The general purpose register output is driven high.</td></tr>
<tr><td>GPO16:GPO7
Bit 20
& Bit 11</td><td>General Purpose Output 16 and 7 — Controls the Pad Drive Strength Group 8 drive strength.

These bits are forced to zero at reset, and can be read or written by the CPU. Reads will return either the reset value, or the previously written value.</td><td>
<table>
<tr><th>GPO16</th><th>GPO7</th><th>Function</th></tr>
<tr><td>0</td><td>0</td><td>4mA</td></tr>
<tr><td>0</td><td>1</td><td>8mA</td></tr>
<tr><td>1</td><td>0</td><td>16mA</td></tr>
<tr><td>1</td><td>1</td><td>16mA</td></tr>
</table>
</td></tr>
</table>

Table 6-17. GPO Bit Descriptions

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>GPO15
Bit 19</td><td>General Purpose Output 15 — Controls the Pad Drive Strength Group 11 drive strength.

These bits are forced to zero at reset, and can be read or written by the CPU. Reads will return either the reset value, or the previously written value.</td><td><table><tr><th>GPO15</th><th>Function</th></tr><tr><td>0</td><td>4mA</td></tr><tr><td>1</td><td>8mA</td></tr></table></td></tr>
<tr><td>GPO14:GPO6
Bit 18
& Bit 10</td><td>General Purpose Output 16 and 7 — Controls the Pad Drive Strength Group 7 drive strength.

These bits are forced to zero at reset, and can be read or written by the CPU. Reads will return either the reset value, or the previously written value.</td><td><table><tr><th>GPO14</th><th>GPO6</th><th>Function</th></tr><tr><td>0</td><td>0</td><td>4mA</td></tr><tr><td>0</td><td>1</td><td>8mA</td></tr><tr><td>1</td><td>0</td><td>16mA</td></tr><tr><td>1</td><td>1</td><td>16mA</td></tr></table></td></tr>
<tr><td>GPO13:GPO3
Bit 17
& Bit 7</td><td>General Purpose Output 13 and 3 — Controls the Pad Drive Strength Group 4 drive strength.

These bits are forced to zero at reset, and can be read or written by the CPU. Reads will return either the reset value, or the previously written value.</td><td><table><tr><th>GPO13</th><th>GPO3</th><th>Function</th></tr><tr><td>0</td><td>0</td><td>4mA</td></tr><tr><td>0</td><td>1</td><td>8mA</td></tr><tr><td>1</td><td>0</td><td>16mA</td></tr><tr><td>1</td><td>1</td><td>16mA</td></tr></table></td></tr>
<tr><td>GPO12:GPO2
Bit 16
& Bit 6</td><td>General Purpose Output 12 and 2 — Controls the Pad Drive Strength Group 3 drive strength.

These bits are forced to zero at reset, and can be read or written by the CPU. Reads will return either the reset value, or the previously written value.</td><td><table><tr><th>GPO12</th><th>GPO2</th><th>Function</th></tr><tr><td>0</td><td>0</td><td>4mA</td></tr><tr><td>0</td><td>1</td><td>8mA</td></tr><tr><td>1</td><td>0</td><td>16mA</td></tr><tr><td>1</td><td>1</td><td>16mA</td></tr></table></td></tr>
<tr><td>GPO11:GPO1
Bit 15
& Bit 5</td><td>General Purpose Output 11 and 1 — Controls the Pad Drive Strength Group 2 drive strength.

These bits are forced to zero at reset, and can be read or written by the CPU. Reads will return either the reset value, or the previously written value.</td><td><table><tr><th>GPO11</th><th>GPO1</th><th>Function</th></tr><tr><td>0</td><td>0</td><td>4mA</td></tr><tr><td>0</td><td>1</td><td>8mA</td></tr><tr><td>1</td><td>0</td><td>16mA</td></tr><tr><td>1</td><td>1</td><td>16mA</td></tr></table></td></tr>
</table>

Table 6-17. GPO Bit Descriptions

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>GPO10:GPO0
Bit 14
& Bit 4</td><td>General Purpose Output 10 and 0 — Controls the Pad Drive Strength Group 1 drive strength.

These bits are forced to zero at reset, and can be read or written by the CPU. Reads will return either the reset value, or the previously written value.</td><td><table><tr><th>GPO10</th><th>GPO0</th><th>Function</th></tr><tr><td>0</td><td>0</td><td>4mA</td></tr><tr><td>0</td><td>1</td><td>8mA</td></tr><tr><td>1</td><td>0</td><td>16mA</td></tr><tr><td>1</td><td>1</td><td>16mA</td></tr></table></td></tr>
<tr><td>GPO9
Bit 13</td><td>General Purpose Output 9 — Controls the Pad Drive Strength Group 10 drive strength.

These bits are forced to zero at reset, and can be read or written by the CPU. Reads will return either the reset value, or the previously written value.</td><td><table><tr><th>GPO9</th><th>Function</th></tr><tr><td>0</td><td>4mA</td></tr><tr><td>1</td><td>8mA</td></tr></table></td></tr>
<tr><td>GPO8
Bit 12</td><td>General Purpose Output 8 — Controls the Pad Drive Strength Group 9 drive strength.

These bits are forced to zero at reset, and can be read or written by the CPU. Reads will return either the reset value, or the previously written value.</td><td><table><tr><th>GPO8</th><th>Function</th></tr><tr><td>0</td><td>4mA</td></tr><tr><td>1</td><td>8mA</td></tr></table></td></tr>
<tr><td>GPO5
Bit 9</td><td>General Purpose Output 5 — Controls the Pad Drive Strength Group 6 drive strength.

These bits are forced to zero at reset, and can be read or written by the CPU. Reads will return either the reset value, or the previously written value.</td><td><table><tr><th>GPO5</th><th>Function</th></tr><tr><td>0</td><td>4mA</td></tr><tr><td>1</td><td>8mA</td></tr></table></td></tr>
<tr><td>GPO4
Bit 8</td><td>General Purpose Output 4 — Controls the Pad Drive Strength Group 5 drive strength.

These bits are forced to zero at reset, and can be read or written by the CPU. Reads will return either the reset value, or the previously written value.</td><td><table><tr><th>GPO4</th><th>Function</th></tr><tr><td>0</td><td>4mA</td></tr><tr><td>1</td><td>8mA</td></tr></table></td></tr>
<tr><td>BCM
Bit 3</td><td>Burst Clock Mode— This bit is used to select the burst clock mode of operation. When set, the BCLK will act as the MCU_CLK. This bit is cleared by hardware reset.
Note: Setting this bit will cause other burst modes to fail!</td><td>0 -The burst clock only runs when accessing a chip select range with the BEN bit set. When the burst clock is not running it will remain in a logic 0 state. When the burst clock is running it will be configured by the CS Burst Clock Control Register.
1 -The burst clock runs all the time. (Independent of CS accesses).</td></tr>
</table>

Table 6-17. GPO Bit Descriptions

Name	Description	Settings
HDB Bit 2	**High Data Bus—** This bit is used to determine which byte lanes of the internal data bus are driven onto the lower 16 external data bus bits when show cycles are enabled. This bit is ignored if SHEN is cleared. This bit is cleared by hardware reset.	0 -Lower internal data bus bits D15-D0 are driven externally. 1 -Upper internal data bus bits D31-D16 are driven externally.
SHEN Bit [1:0]	**Show Cycle Enable —** These two bits are used to determine what the AEIM does with the external bus during internal transfers (e.g. an access to the internal ROM, RAM or internal peripherals). When show cycles are enabled, the internal address and data bus are driven externally. When Show cycles is disabled, the external address lines will not toggle during burst accesses. Additionally, the PSTAT[3:0], SIZ[1:0], SHOW_RW_B, and STROBE outputs of the AEIM are disabled. These bits are cleared by hardware reset.	See AEIM Chapter For Details

6.9 USB Module Configuration

Two inputs to the USB module have Neptune specific configurations.

- The Full Speed Enable (FSEN) input determines whether the USB defaults to a full speed device, or a low speed device. When set to ‘1’, the USB operates as a full speed device (12MHZ). When set to a ‘0’, the USB operates as a low speed device(1.5MHZ). This input is tied high in the Neptune.
- The Big or Little Endian (BIGEND) select input determines the type of buffer accesses for the CPU. A ‘1’ indicates Big endian, and a ‘0’ indicates Little endian buffer accesses for the CPU. This line is tied high in the Neptune allowing Big endian access only. Little endian accesses are not supported in the Neptune configuration.

Refer to Section 21.3, “USB Module Pin List,” for more information on USB I/O.

6.10 UART2 Module Configuration

Since the UART2 DTR input signal is not available on any pin, the signal is tied high inside the Neptune chip. UART2 will behave as if the DTE device to which UART2 is connected is always ready.

6.11 SCC Debug Signal Configuration

The SCC debug signals are connected in the chip as follows:

- debug1 - tcm_test
- debug2 - tcm_test1
- debug3 - tcm_test2
- debug4 - AEIM show cycles
- debug5 - dbgen (arm debug enable from SJC)

Chapter 7
Synthesizable ONYXU (SONYXU)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	29/Oct/02	David Liddle	Updated for new address trace strobe timing.
0.2	04/29/03	Mark Babcock, Shannon Osgood	Updated per DDTS# DSPhl15829.
0.3	05/07/03		Updated for LTE specification release.

7.1 Introduction

This introduction describes the DSP56600S Core, a core of Motorola's family of programmable CMOS digital signal processors. The DSP56600S is a synthesizable version of DSP56600 core, capable of executing an instruction on every clock cycle.

The DSP56600S core is composed of the Data ALU, Address Generation Unit, Program Controller, Program Patch Detector, Bus Interface Unit, and an On-chip Emulator. A standard interface between the DSP56600S core and the on-chip memory and peripherals supports many memory and peripheral configurations.

Detailed information about theDSP56600 core is found in the DSP56600 16-bit Digital Signal Processor Family Manual.

7.1.1 High Performance DSP56600 Core

- 130 Million Instructions per Second (MIPS) with a 130 MHz clock.
- Fully pipelined 16 x 16 Bit Parallel Multiplier-Accumulator
- 40 Bit Parallel Barrel Shifter
- Highly Parallel Instruction Set with Unique DSP Addressing Modes
- Code Compatible with the 56300 Core
- Position Independent Code (PIC) support
- Nested Hardware Do Loops
- Fast Auto-Return Interrupts
- On-Chip support for software patching and enhancements
- On-Chip Emulator (OnCE)
- JTAG port

7.1.2 DSP Core Low Power support

- Low power Wait for Interrupt standby mode
- Ultra Low power STOP standby mode

7.2 Core Registers

See *DSP56600 Family Manual*, sections 3, 4, and 5. See Figure 5-7 of that document for a programming model diagram.

7.3 Instructions

7.3.1 Instruction Types

See *DSP56600 Family Manual*, section A.3.

7.3.2 Addressing Modes

See *DSP56600 Family Manual*, section 4.4.

7.3.3 Instruction Descriptions

See *DSP56600 Family Manual*, section A.5.

7.3.4 Opcode Map

TBD

7.4 Instruction Execution

This section must describe how the core fetches, decodes, and executes instructions. It provides details concerning changes of execution flow and instruction cycle timing. Aspects of operation that affect tracing and debugging, such as recognition of breakpoints, should also be described.

7.4.1 Normal Instruction Execution

See *DSP56600 Family Manual*, section 5.3.1.

7.4.2 Execution sequence

See *DSP56600 Family Manual*, section 7.2.

7.4.3 Changes of flow

See *DSP56600 Family Manual*, section 5.4.5.

7.4.4 Instruction Timing

See *DSP56600 Family Manual*, Appendix B.

7.5 Exception Processing

See *DSP56600 Family Manual*, section 7.

7.6 Core Interface

7.6.1 Core Interface Overview

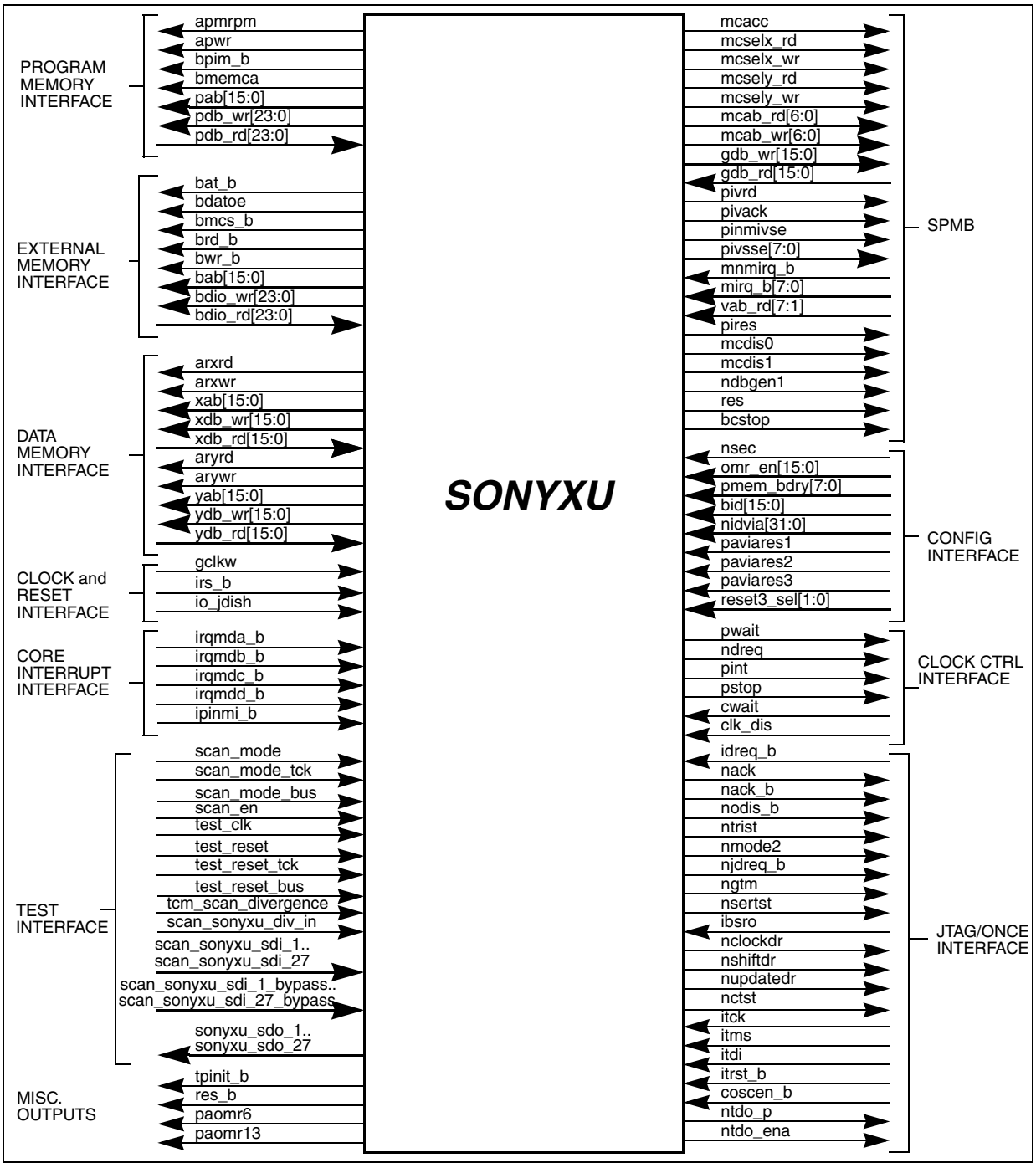


Figure 7-1. Core Interface Signals

7.7 SONYXU Module Pin List

Table 7-1 shows the signals that interface to the core. Refer to Chapter 75, “Neptune LTE Electrical Requirements (ELEC),” for electrical specifications.

Table 7-1. SONYXU Module Pin List

Pin Name	Direction	Description
Program Memory Interface		
apmrpm	O	Program Memory Read
apwr	O	Program Memory Write
bpim_b	O	Internal Program Memory Enable
bmemca	O	Program Memory Read Data Hold
pab[15:0]	O	Program Memory Address Bus
pdb_wr[23:0]	O	Program Memory Write Data Bus
pdb_rd[23:0]	I	Program Memory Read Data Bus
External Memory Interface (Port A)		
bat_b	O	Address Trace
bdatoe	O	Data Output Enable
bmcs_b	O	Chip Select
brd_b	O	Read Strobe
bwr_b	O	Write Strobe
bab[15:0]	O	Port A Address Bus
bdio_wr[23:0]	O	Port A Write Data Bus
bdio_rd[23:0]	I	Port A Read Data Bus
Data Memory Interface		
arxrd	O	X Memory Read
arxwr	O	X Memory Write
xab[15:0]	O	X Memory Address Bus
xdb_wr[15:0]	O	X Memory Write Data Bus
xdb_rd[15:0]	I	X Memory Read Data Bus
aryrd	O	Y Memory Read
arywr	O	Y Memory Write
yab[15:0]	O	Y Memory Address Bus
ydb_wr[15:0]	O	Y Memory Write Data Bus

Table 7-1. SONYXU Module Pin List (Continued)

Pin Name	Direction	Description
ydb_rd[15:0]	I	Y Memory Read Data Bus
Clock and Reset Interface		
gclkw	I	System Clock
irs_b	I	Core Reset
io_jdish	I	Core Reset due to Voltage Difference
Core Interrupt Interface		
irqmda_b	I	Interrupt Request A
irqmdb_b	I	Interrupt Request B
irqmdc_b	I	Interrupt Request C
irqmdd_b	I	Interrupt Request D
ipinmi_b	I	Non-Maskable Interrupt Request
Test Interface		
scan_mode	I	Scan Mode
scan_mode_tck	I	Scan Mode for TCK Logic
scan_mode_bus	I	Scan Mode for Bus Logic
scan_en	I	Scan Enable
test_reset	I	Test Reset
test_reset_tck	I	Test Reset for TCK Logic
test_reset_bus	I	Test Reset for Bus Logic
test_clk	I	Test Clock
tcm_scan_divergence	I	Scan Divergence Mode
scan_sonyxu_div_in	I	Scan Divergence In
scan_sonyxu_sdi_X[27:1]	I	Scan Data In
scan_sonyxu_sdi_X_bypass[27:1]	I	Scan Data In Bypass
sonyxu_sdo_X[27:1]	O	Scan Data Out
Synthesizable Peripheral Module Bus (SPMB)		
mcacc	O	I/O Space Access
mcselx_rd	O	X I/O Space Read
mcselx_wr	O	X I/O Space Write

Table 7-1. SONYXU Module Pin List (Continued)

Pin Name	Direction	Description
mcsely_rd	O	Y I/O Space Read
mcsely_wr	O	Y I/O Space Write
mcab_rd[6:0]	O	I/O Space Read Address Bus
mcab_wr[6:0]	O	I/O Space Write Address Bus
gdb_wr[15:0]	O	I/O Space Write Data Bus
gdb_rd[15:0]	I	I/O Space Read Data Bus
pivrd	O	Peripheral Interrupt Vector Read
pivack	O	Peripheral Interrupt Acknowledge
pinmivse	O	Peripheral NMI Vector Select
pivsse[7:0]	O	Peripheral Interrupt Vector Selects
mnmirq_b	I	Peripheral Non-Maskable Interrupt Request
mirq_b[7:0]	I	Peripheral Interrupt Requests
vab_rd[7:1]	I	Peripheral Interrupt Vector Bus
pires	O	Peripheral Software Reset
res	O	System Hardware Reset
mcdis0	O	Core Stall Phase 0
mcdis1	O	Core Stall Phase 1
ndbgen1	O	Core Debug Mode
bcstop	O	Core Going to Stop
Configuration Interface		
nsec	I	Disable OnCE
omr_en[15:0]	I	OMR Enable
pmem_bdry[7:0]	I	Program Space Internal/External Boundary
bid[15:0]	I	IDR Value
nidvia[31:0]	I	JTAG ID
paviares1	I	RESET1 Configuration
paviares2	I	RESET2 Configuration
paviares3	I	RESET3 Configuration
reset3_sel[1:0]	I	RESET3 Select

Table 7-1. SONYXU Module Pin List (Continued)

Pin Name	Direction	Description
JTAG/OnCE Interface		
idreq_b	I	Debug Event
nack	O	Debug Acknowledge
nack_b	O	Debug Acknowledge — Active Low
nodis_b	O	Disable Inputs
ntrist	O	Tristate Outputs
nmode2	O	System Reset due to JTAG Instruction
njdreq_b	O	JTAG DEBUG_REQUEST Instruction
ngtm	O	General Test Mode
nsertst	O	Serial Test Mode
nctst	O	Clock Test Mode
ibsro	I	JTAG Boundary Scan Register Output
nclockdr	O	JTAG Boundary Scan Register Control
nshiftdr	O	JTAG Shift-DR Control
nupdatedr	O	JTAG Update-DR Strobe
itck	I	JTAG/OnCE Test Clock
itms	I	JTAG/OnCE Test Mode Select
itdi	I	JTAG/OnCE Test Data In
itrst_b	I	JTAG/OnCE Test Reset
coscen_b	I	Power-On Reset
ntdo_p	O	JTAG/OnCE Test Data Out
ntdo_ena	O	JTAG/OnCE Test Data Output Enable
Clock Control Interface		
pwait	O	Processing Wait
pint	O	Exit Wait due to Interrupt Request
ndreq	O	Exit Wait due to Debug Request
pstop	O	Processing Stop
tpinit_b	O	PLL Enable Initial Value
cwait	I	Clock Wait

Table 7-1. SONYXU Module Pin List (Continued)

Pin Name	Direction	Description
clk_dis	I	Clock Disable for Wait
Miscellaneous Outputs		
res_b	O	System Hardware Reset — Active Low
paomr6	O	OMR Bit 6
paomr13	O	OMR Bit 13

7.8 Signal Descriptions

7.8.1 Program Memory Interface Signals

7.8.1.1 Program Memory Read (apmrpm)

This output indicates a program memory read access.

7.8.1.2 Program Memory Write (apwr)

This output indicates a program memory write access.

7.8.1.3 Internal Program Memory Enable (bpim_b)

This active-low output indicates that the program memory access is to an internal address. When bpim_b is negated, the internal memory should not respond to apmrpm or apwr.

7.8.1.4 Program Memory Read Data Hold (bmemca)

This output indicates that the core is not ready to receive the read data from the program memory. The memory should hold the data until this signal is negated.

7.8.1.5 Program Memory Address Bus (pab[15:0])

These outputs provide the address for a program memory read or write access. These signals are valid when apmrpm or apwr is asserted.

7.8.1.6 Program Memory Write Data Bus (pdb_wr[23:0])

These outputs provide the data for a program memory write access.

7.8.1.7 Program Memory Read Data Bus (pdb_rd[23:0])

These inputs provide the data for a program memory read access.

7.8.2 External Memory Interface (Port A) Signals

7.8.2.1 Address Trace (bat_b)

This output provides the AT output which indicates when the address can be sampled by an external device.

7.8.2.2 Data Output Enable (bdatoe)

This output is the output enable for the Port A data.

7.8.2.3 Chip Select (bmcs_b)

This output provides the MCS pin.

7.8.2.4 Read Strobe (brd_b)

This output provides the RD pin.

7.8.2.5 Write Strobe (bwr_b)

This output provides the WR pin.

7.8.2.6 Port A Address Bus (bab[15:0])

These outputs provide the address for an access to external program memory.

7.8.2.7 Port A Write Data Bus (bdio_wr[23:0])

These outputs provide the data for a write access to external program memory.

7.8.2.8 Port A Read Data Bus (bdio_rd[23:0])

These inputs provide the data for a read access from external program memory.

7.8.3 Data Memory Interface Signals

7.8.3.1 X Memory Read (arxrd)

This output indicates an X data memory read access.

7.8.3.2 X Memory Write (arxwr)

This output indicates an X data memory write access.

7.8.3.3 X Memory Address Bus (xab[15:0])

These outputs provide the address for an X data memory access.

7.8.3.4 X Memory Write Data Bus (xdb_wr[15:0])

These outputs provide the data for an X data memory write access.

7.8.3.5 X Memory Read Data Bus (xdb_rd[15:0])

These inputs provide the data for an X data memory read access.

7.8.3.6 Y Memory Read (aryrd)

This output indicates an Y data memory read access.

7.8.3.7 Y Memory Write (arywr)

This output indicates an Y data memory write access.

7.8.3.8 Y Memory Address Bus (yab[15:0])

These outputs provide the address for an Y data memory access.

7.8.3.9 Y Memory Write Data Bus (ydb_wr[15:0])

These outputs provide the data for an Y data memory write access.

7.8.3.10 Y Memory Read Data Bus (ydb_rd[15:0])

These inputs provide the data for an Y data memory read access.

7.8.4 Clock and Reset Interface Signals

7.8.4.1 System Clock (gclkw)

This input provides the system clock for the DSP sub-system.

7.8.4.2 Core Reset (irs_b)

This asynchronous, active-low input resets the core and the DSP sub-system. The mode pins (see Section 7.7.5 Core Interrupt Interface Signals) are sampled at the rising edge (negation) of this signal.

7.8.4.3 Core Reset due to Voltage Difference (io_jdish)

This asynchronous input resets the core and the DSP sub-system. This input is intended to be connected to circuitry that detects voltage problems. The effect of this input is identical to irs_b, except that it does not sample the mode pins.

7.8.5 Core Interrupt Interface Signals

7.8.5.1 Interrupt Request A (irqmda_b)

This asynchronous, active-low input functions provides Interrupt Request A to the core. The core can be programmed to be sensitive to either a falling edge or a low level on this pin.

While irs_b is asserted, this input functions as the active-high Mode A pin. Its value is sampled into Bit 0 of the OMR and affects the core operating mode.

7.8.5.2 Interrupt Request B (irqmdb_b)

This asynchronous, active-low input functions provides Interrupt Request B to the core. The core can be programmed to be sensitive to either a falling edge or a low level on this pin.

While irs_b is asserted, this input functions as the active-high Mode B pin. Its value is sampled into Bit 1 of the OMR and affects the core operating mode.

7.8.5.3 Interrupt Request C (irqmdc_b)

This asynchronous, active-low input functions provides Interrupt Request C to the core. The core can be programmed to be sensitive to either a falling edge or a low level on this pin.

While irs_b is asserted, this input functions as the active-high Mode C pin. Its value is sampled into Bit 2 of the OMR and affects the core operating mode.

7.8.5.4 Interrupt Request D (irqmdd_b)

This asynchronous, active-low input functions provides Interrupt Request D to the core. The core can be programmed to be sensitive to either a falling edge or a low level on this pin.

While irs_b is asserted, this input functions as the active-high Mode D pin. Its value is sampled into Bit 3 of the OMR and affects the core operating mode.

7.8.5.5 Non-Maskable Interrupt Request (ipinmi_b)

This asynchronous, active-low input functions provides a non-maskable interrupt to the core. The core is sensitive to a falling edge on this pin.

While irs_b is asserted, this input functions as the active-high PLL Init pin. Its value is provided on the tpinit_b output.

7.8.6 Test Interface Signals

7.8.6.1 Scan Mode (scan_mode)

This input indicates that the GCLKW registers of the core not in the bus logic are in Scan Mode. This input must be at logic 0 for normal operation of the core.

7.8.6.2 Scan Mode for TCK Logic (scan_mode_tck)

This input indicates that the TCK registers of the core are in Scan Mode. This input must be at logic 0 for normal operation of the core.

7.8.6.3 Scan Mode for Bus Logic (scan_mode_bus)

This indicates that the registers that are connected to the pins are in Scan Mode. This input must be at logic 0 for normal operation of the core.

7.8.6.4 Scan Enable (scan_en)

This input indicates a shift cycle. It is used only in Scan Mode.

7.8.6.5 Test Reset (test_reset)

This asynchronous input brings all GCLKW registers of the core not in the bus logic to their reset values. It should only be asserted in Scan Mode.

7.8.6.6 Test Reset for TCK Logic (test_reset_tck)

This asynchronous input brings all TCK registers of the core to their reset values. It should only be asserted in Scan Mode.

7.8.6.7 Test Reset for Bus Logic (test_reset_bus)

This asynchronous input brings all registers connected to the pins of the core to their reset values. It should only be asserted in Scan Mode.

7.8.6.8 Test Clock (test_clk)

This input provides the clock for scan testing. It is used only in Scan Mode.

7.8.6.9 Scan Divergence Mode (tcm_scan_divergence)

This input indicates that the scan chains should be combined into one chain for scan divergence or burn-in testing. It is used only in Scan Mode.

7.8.6.10 Scan Divergence In (scan_sonyxu_div_in)

This input provides the input data for the single scan chain when scan divergence is active.

7.8.6.11 Scan Data In (scan_sonyxu_sdi_1 .. scan_sonyxu_sdi_27)

These inputs provide the input data for the core's 27 scan chains. They are used only in Scan Mode when scan divergence is inactive.

7.8.6.12 Scan Data In Bypass (scan_sonyxu_sdi_1_bypass .. scan_sonyxu_sdi_27_bypass)

These inputs are muxed onto the Scan Data Out pins when Scan Mode is inactive.

7.8.6.13 Scan Data Out (sonyxu_sdo_1 .. sonyxu_sdo_27)

These outputs provide the output data from the core's 27 scan chains. They are driven only in Scan Mode. During scan divergence or burn-in testing only sonyxu_sdo_27 is used. See the Test Guide for more information.

7.8.7 Synthesizable Peripheral Module Bus (SPMB) Signals

7.8.7.1 I/O Space Access (mcacc)

This output indicates that there is activity in the I/O space. It may be used to gate the clock to the SPMB interface logic in order to reduce the power consumption when there is no activity on this bus.

7.8.7.2 X I/O Space Read (mcselx_rd)

This output indicates an X I/O space read from the address provided by mcab_rd[6:0].

7.8.7.3 X I/O Space Write (mcselx_wr)

This output indicates an X I/O space write to the address provided by mcab_wr[6:0].

7.8.7.4 Y I/O Space Read (mcsely_rd)

This output indicates a Y I/O space read from the address provided by mcab_rd[6:0].

7.8.7.4.1 Y I/O Space Write (mcsely_wr)

This output indicates a Y I/O space write to the address provided by mcab_wr[6:0].

7.8.7.5 I/O Space Read Address Bus (mcab_rd[6:0])

These outputs provide the address for X or Y I/O space read accesses.

7.8.7.6 I/O Space Write Address Bus (mcab_wr[6:0])

These outputs provide the address for X or Y I/O space write accesses.

7.8.7.7 I/O Space Write Data Bus (gdb_wr[15:0])

These outputs provide the data for X or Y I/O space write accesses.

7.8.7.8 I/O Space Read Data Bus (gdb_rd[15:0])

These inputs provide the data for X or Y I/O space write accesses.

7.8.7.9 Peripheral Interrupt Vector Read (pivrd)

This output indicates that the core is reading the value currently on vab_rd[7:1].

7.8.7.10 Peripheral Interrupt Acknowledge (pivack)

This output provides an acknowledge to a peripheral interrupt. The interrupt being acknowledged is indicated by pinmivse or pivsse[7:0].

7.8.7.11 Peripheral NMI Vector Select (pinmivse)

This output indicates that the peripheral non-maskable interrupt is currently at the highest priority to be accepted by the core. When this signal is at a logic one, the non-maskable interrupt vector should be provided on vab_rd[7:1].

7.8.7.12 Peripheral Interrupt Vector Selects (pivsse[7:0])

These outputs correspond to the eight peripheral interrupt request signals (mirq_b[7:0]). Each output indicates that the corresponding interrupt request is currently at the highest priority to be accepted by the core and its vector should be provided on vab_rd[7:1]. At most one of these outputs can be at a logic one value.

7.8.7.13 Peripheral Non-Maskable Interrupt Request (mnmirq_b)

This active-low input provides a non-maskable interrupt request from a peripheral.

7.8.7.14 Peripheral Interrupt Requests (mirq_b[7:0])

These active-low inputs provide interrupt requests from peripherals.

7.8.7.15 Peripheral Interrupt Vector Bus (vab_rd[7:1])

These inputs provide the interrupt vector address for the interrupt currently selected by one of the pinmivse or pivsse[7:0] signals.

7.8.7.16 Peripheral Software Reset (pires)

This output indicates that the RESET instruction has been executed and all peripherals should be reset. It is synchronous to gclkw.

7.8.7.17 System Hardware Reset (res)

This output indicates that a core hardware reset has occurred. The negation of this signal is synchronous to gclkw, but its assertion may be asynchronous.

7.8.7.18 Core Stall Phase 0 (mcdis0)

This output indicates that the core has a phase 0 pipeline stall. This signal affects the timing of I/O space accesses.

7.8.7.19 Core Stall Phase 1 (mcdis1)

This output indicates that the core has a phase 1 pipeline stall. This signal affects the timing of I/O space accesses. It also causes a read data delay for X and Y data memory read accesses.

7.8.7.20 Core Debug Mode (ndbgen1)

This output indicates that the core is in Debug Mode. It may be used by peripherals that need this information.

7.8.7.21 Core Going to Stop (bcstop)

This output is asserted a few clock cycles before the clocks are stopped in the Stop Processing State, allowing peripheral devices to prepare for this state.

7.8.8 Configuration Interface Signals

These signals are used to configure the core when it is used in a chip. They should all be tied to constant logic values.

7.8.8.1 Disable OnCE (nsec)

This input indicates that the OnCE logic should be disabled.

7.8.8.2 OMR Enable (omr_en[15:0])

These inputs indicate which bits of the Operating Mode Register (OMR) should be enabled. Bits that are disabled cannot be written and are always read as 0.

7.8.8.3 Program Space Internal/External Boundary (pmem_bdry[7:0])

These inputs define the boundary between the internal and external address spaces of the program memory. The internal space is from $0000 to {pmem_bdry[7:0], $ff}. The external space is from ({pmem_bdry[7:0], $ff} + 1) to $ffff.

7.8.8.4 IDR Value (bid[15:0])

These inputs provide the values of the core ID register.

7.8.8.5 JTAG ID (nidvia[31:0])

These inputs provide the value of the JTAG ID register.

7.8.8.6 RESET1 Configuration (paviares1)

This input determines the reset program address in Operating Mode 0.

$c000

$8000

7.8.8.7 RESET2 Configuration (paviares2)

This input determines the reset program address in Operating Mode 8.

$4000

$0000

7.8.8.8 RESET3 Configuration (paviares3)

This input determines the reset program address in Operating Modes other than 0 or 8.

$0200

depends on reset3_sel[1:0]

7.8.8.9 RESET3 Select (reset3_sel[1:0])

These inputs determine the reset program address in Operating Modes other than 0 or 8 if paviares3 is logic one.

$0400

$0800

$1000

$2000

7.8.9 JTAG/OnCE Interface Signals

7.8.9.1 Debug Event (idreq_b)

This active-low input provides a request to enter debug mode. This is normally input from the chip input port.

7.8.9.2 Debug Acknowledge (nack)

This output acknowledges that the core has recognized the debug request and entered the debug mode.

7.8.9.3 Debug Acknowledge — Active Low (nack_b)

This active-low output has the same funtionality as nack.

7.8.9.4 Disable Inputs (nodis_b)

This active-low output indicates that chip inputs should be disabled. It is a function of the current JTAG instruction.

7.8.9.5 Tristate Outputs (ntrist)

This output indicates that chip outputs should be tristated. It is a function of the current JTAG instruction.

7.8.9.6 System Reset due to JTAG Instruction (nmode2)

This output indicates provides a DSP sub-system reset as a result of certain JTAG instructions.

7.8.9.7 JTAG DEBUG_REQUEST Instruction (njdreq_b)

This active-low output indicates that the current JTAG instruction is DEBUG_REQUEST.

7.8.9.8 General Test Mode (ngtm)

This output indicates that the General Test Mode is active.

7.8.9.9 Serial Test Mode (nsertst)

This output indicates that the Serial Test Mode is active.

7.8.9.10 Clock Test Mode (nctst)

This output indicates that the Clock Test Mode is active.

7.8.9.11 JTAG Boundary Scan Register Output (ibsro)

This input is the output of the last stage of the Boundary Scan Register.

7.8.9.12 JTAG Boundary Scan Register Control (nclockdr)

This is the enable signal for the Boundary Scan Register.

7.8.9.13 JTAG Shift-DR Control (nshiftdr)

This output selects the shift data vs. the capture data to be sampled into the Boundary Scan Register when nclockdr is asserted.

7.8.9.14 JTAG Update-DR Strobe (nupdatedr)

This gated clock output samples data into the parallel output latches of the Boundary Scan Register.

7.8.9.15 JTAG/OnCE Test Clock (itck)

This input is the JTAG TCK input from the chip port.

7.8.9.16 JTAG/OnCE Test Mode Select (itms)

This input is the JTAG TMS input from the chip port.

7.8.9.17 JTAG/OnCE Test Data In (itdi)

This input is the JTAG TDI input from the chip port.

7.8.9.18 JTAG/OnCE Test Reset (itrst_b)

This active-low input is the JTAG TRST input from the chip port.

7.8.9.19 Power-On Reset (coscen_b)

This active-low input provides power-on reset and is functionally equivalent to the itrst_b signal.

7.8.9.20 JTAG/OnCE Test Data Out (ntdo_p)

This output provides the data to be driven on the JTAG TDO chip port.

7.8.9.21 JTAG/OnCE Test Data Output Enable (ntdo_ena)

This output provides the output enable for the JTAG TDO chip port.

7.8.10 Clock Control Interface Signals

7.8.10.1 Processing Wait (pwait)

This output indicates that a WAIT instruction has been decoded, and the clock logic should enter the Wait Processing State.

7.8.10.2 Exit Wait due to Interrupt Request (pint)

This output indicates that the clock logic should exit the Wait Processing State because an unmasked interrupt has been asserted.

7.8.10.3 Exit Wait due to Debug Request (ndreq)

This output indicates that the clock logic should exit the Wait Processing State because a debug request has been asserted.

7.8.10.4 Processing Stop (pstop)

This output indicates that a STOP instruction has been decoded, and the clock logic should enter the Stop Processing State.

7.8.10.5 PLL Enable Initial Value (tpinit_b)

This active-low output provides the initial value for the PLL Enable control bit. It is generated by sampling ipinmi_b at the negation of irs_b.

7.8.10.6 Clock Wait (cwait)

This input indicates that the clock logic is in the Wait or Stop Processing State.

7.8.10.7 Clock Disable for Wait (clk_dis)

This input is used to gate the system clock during the Wait Processing State to reduce the power consumption.

7.8.11 Miscellaneous Outputs Signals

7.8.11.1 System Hardware Reset — Active Low (res_b)

This active-low output has the same functionality as res.

7.8.11.2 OMR Bit 6 (paomr6)

This output is connected to Bit 6 of the Operating Mode Register (OMR).

7.8.11.3 OMR Bit 13 (paomr13)

This output is connected to Bit 13 of the Operating Mode Register (OMR).

7.9 DSP Address Trace Timing

The following diagram shows the timing (on a cycle-by-cycle basis) for the DSP address tracing. This diagram assumes that no external memory accesses are occuring at the same time that address tracing is active. If external memory accesses do occur, then the address tracing will be suspended until the access completes.

The address trace strobe (bat_b) signal will toggle once each time a new instruction is fetched. This signal will not toggle if a data fetch is made to program memory. In the case of a data fetch to program memory, that address will not be visible on the address tracing pins

A logic analyser connected for DSP address tracing should be configured to sample the address whenever the strobe changes from 0 to 1 OR 1 to 0.

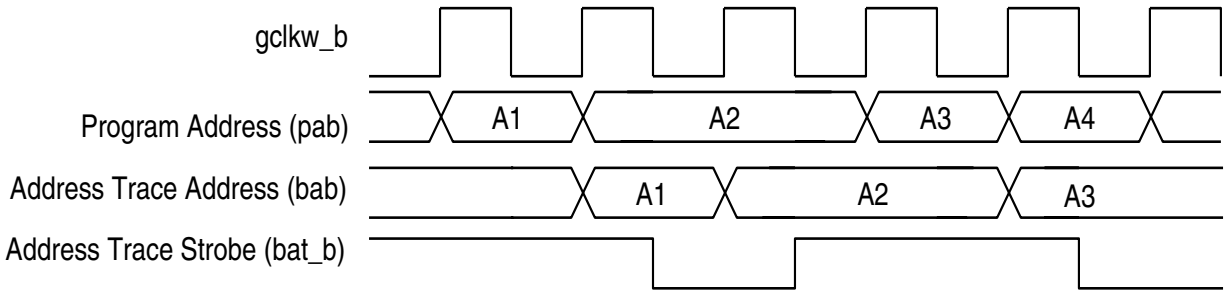


Figure 7-2. DSP Address Tracing Timing Diagram

In the following is a description of the various addresses:

- A1 - Program Instruction Fetch Address
- A2 - Program Instruction Fetch Address with a 1 cycle pipeline stall (in case of longer stalls, this would be extended accordingly)
- A3 - Program Instruction Fetch Address
- A4 - Program Data Fetch Address

It is also worth noting that due to chip level logic, this timing will change at the chip level.

Chapter 8
DSP Memory (DSPMEM)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	05/03/02	Thomas Jew, Scott King, Shannon Osgood	Updates made from SPMC 2 Block Guide Rev 0, Thomas Jew and Scott King.
0.2	05/07/03		Initial release of LTE only specification.

8.1 Overview

The DSPMEM is an embedded, scalable memory system tailored for use with the S-OnyxU DSP. The DSPMEM provides SRAM and diffusion programmable ROM for use as program and data memory store. Figure 8-1 is a block diagram that illustrates the functional partitioning of the DSPMEM.

8.1.1 Modes of Operation

The DSPMEM supports the following modes of operation:

- User Mode

  User mode is the default operational mode of the DSPMEM. In this mode, reads may be performed to the SRAM and ROM blocks and writes may be performed to the SRAM block.

- Stop Mode

  In DSP Stop mode the DSP cannot access the memories. However, if the MCU clocks are active, the MDI and MDPI MCU peripherals will still be able to access the DSP memories to which they are connected.

- Factory Test Mode

  Factory test mode is reserved for factory testing by Motorola.

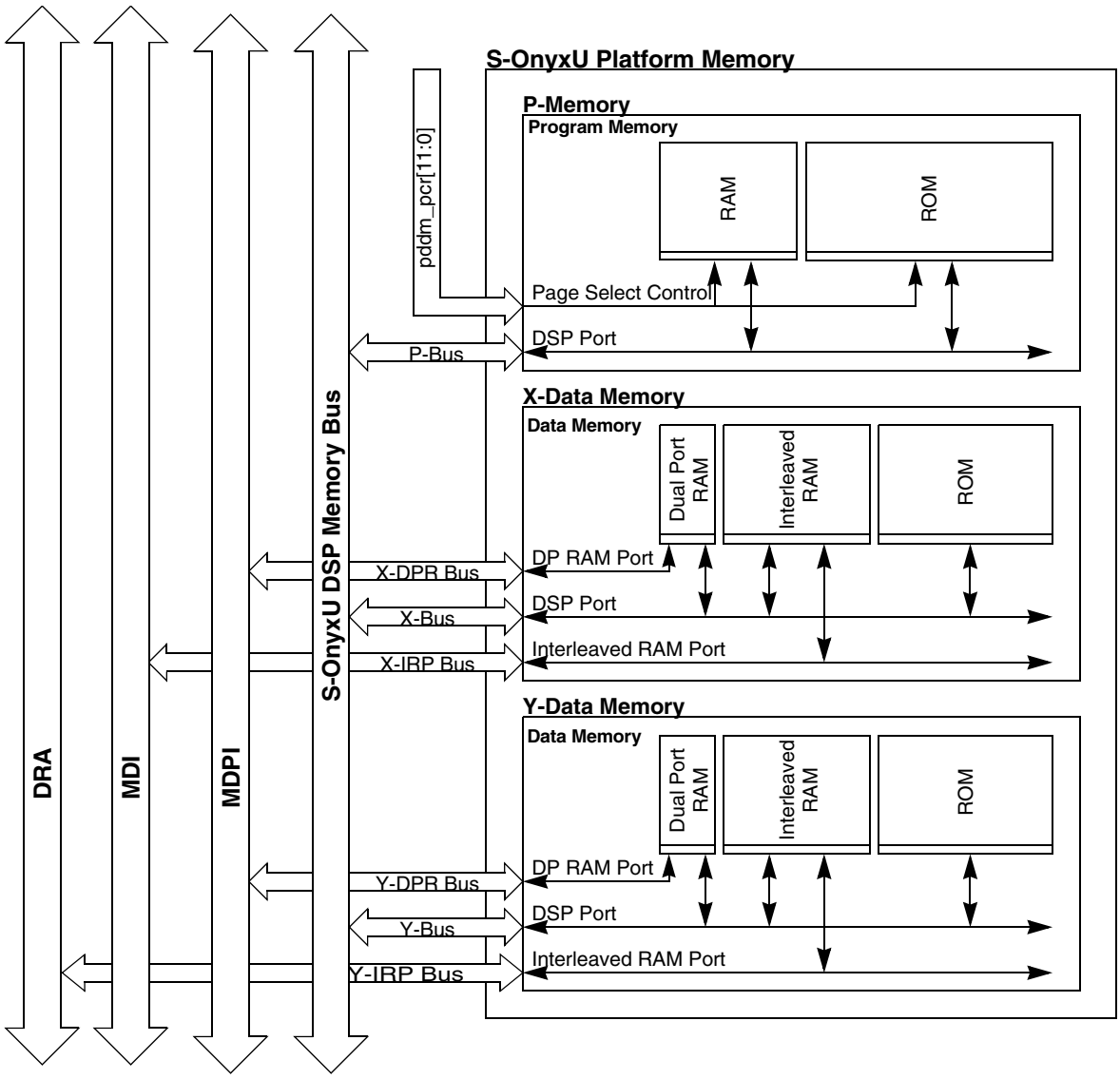


Figure 8-1. DSPMEM Block Diagram

8.2 Functional Description

The DSPMEM is an embedded memory system which can be accessed via the S-OnyxU DSP memory bus.

Read and write accesses may be performed to the SRAM partitions within the p, x, and y memories when the DSPMEM is in user mode operation.

Read accesses may be performed to the ROM partitions within the p, x, and y memories when the DSPMEM is in user mode. The DSPMEM does not respond to write accesses addressing the ROM partitions. Write accesses to ROM locations cannot alter the contents of the p, x, or y memories.

The DSPMEM does not respond to read or write accesses which address unimplemented memory locations in p, x, or y memory maps. Unimplemented memory locations are those locations in the addressable, non-reserve p, x, and y memory map which are not occupied by SRAM or ROM.

8.2.1 Program Memory Operation

The program memory is a single port, paged memory accessed via the DSP port as shown in Figure 8-2.

8.2.2 X And Y Data Memory Operation

The data memory is accessed via three separate ports as shown in Figure 8-2: the DSP Port, the Dual Port RAM Port (DP RAM Port), and the Interleaved RAM Port. The DSP Port can access all of the implemented memory, both SRAM and ROM, in the data memory. The DP RAM Port can only access the dual port RAM portion of the data memory. The Interleaved RAM Port can only access the interleaved SRAM not contained within the dual port RAM block.

8.2.2.1 Dual Port RAM Block

The dual port RAM block may be accessed via the DSP Port and the DP RAM Port. The data memory controller does not contain any hardware to arbitrate or synchronize accesses to the dual port RAM block. All management of accesses to the two different ports of the dual port RAM block and data coherency of the dual port RAM contents are managed by the application software.

The X/Y Data Memory uses a late write architecture which is illustrated in Figure 8-2.

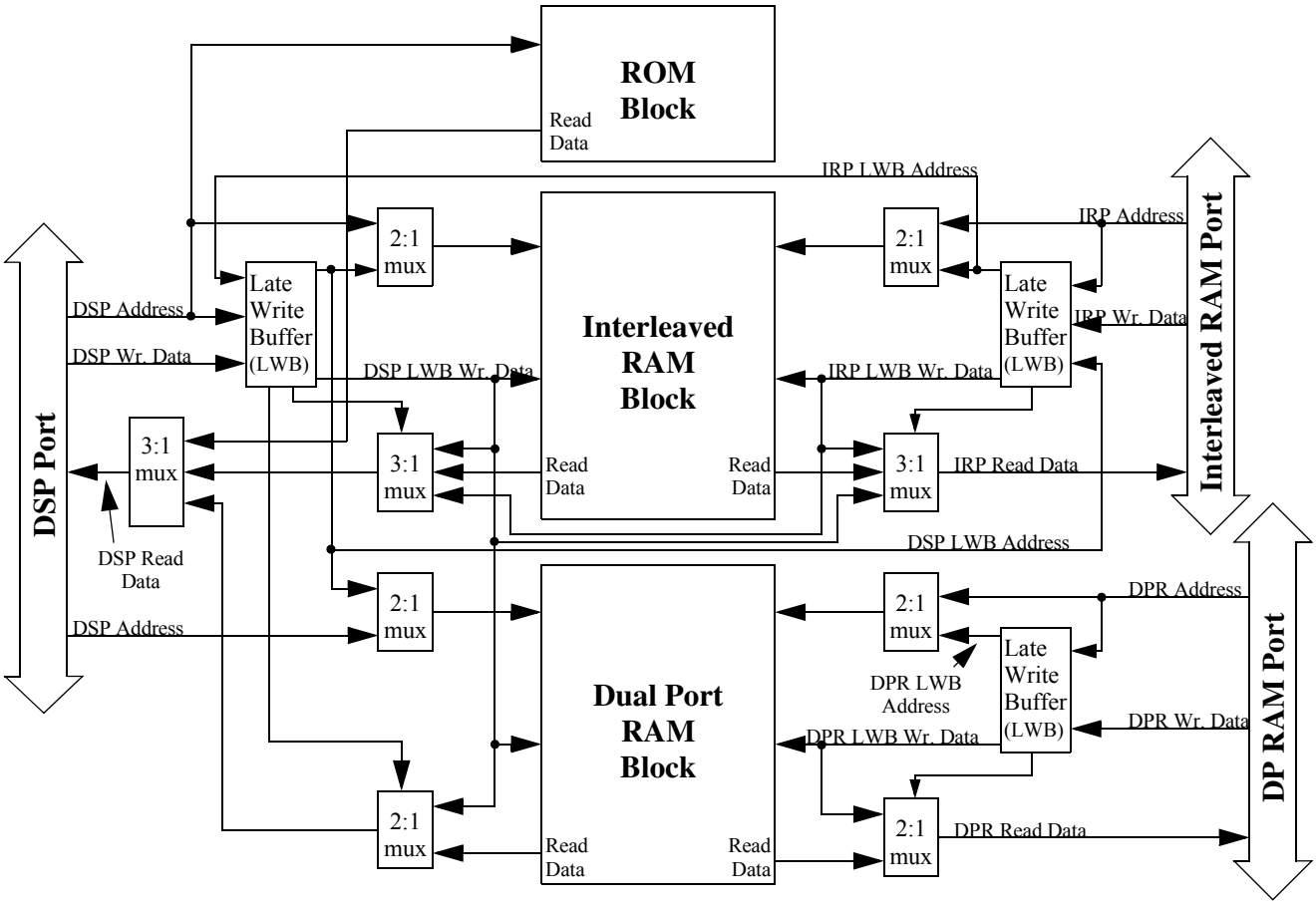


Figure 8-2. X/Y Data Memory Block Late Write Architecture

The late write memory architecture uses a late write buffer to capture address and data information presented to the memory during a write operation. A write operation to the dual port RAM block loads address and data to the late write buffer. The next write will push the current contents of the late write buffer into the memory array and load new address and data into the late write buffer. The “next write” operation may occur on the next system clock cycle immediately following a write operation or may occur an indefinite number of clock cycles later as dictated by the software application. For read operations which occur during the interim between a write and the “next write”, the address in the late write buffer is compared against the incoming read address. If the comparison results in an address match, the data contents of the late write buffer are returned as the result of the read. If the address comparison does not result in a match, the read data is returned from the addressed RAM array location. By comparing the incoming address on read operations against the address stored in the late write buffer, data coherency for the memory is guaranteed for all sequences of read and write operations.

Write operations which are aborted do not successfully load address and data information into the late write buffer for transfer into the SRAM array. In addition, aborted write operations do not alter the contents of the SRAM array.

As shown in Figure 8-2, the dual port RAM block has two independent late write buffers. The DSP port is associated to one late write buffer, while the DP RAM port uses another late write buffer. The contents of the two late write buffers are managed independently of one another for read and write operations. Writes to the DSP port only update the DSP port’s late write buffer while writes to the DP RAM port only update the DP RAM port’s late write buffer.

During reads to the DSP port, the DSP port’s read address is only compared against the DSP port’s late write buffer address contents for an address match. The read address presented on the DSP port is not compared against the address stored in the DP RAM port’s late write buffer. Reads of the dual port RAM block through the DSP port only return contents from the RAM array or the DSP port’s late write buffer.

Likewise, reads of the DP RAM port only return contents from the RAM array or the DP RAM port’s late write buffer. The read address presented to the DP RAM port is compared with the DP RAM port’s late write buffer address contents for an address match. The read address presented on the DP RAM port is not compared with the address stored in the DSP port’s late write buffer.

Data coherency is only guaranteed on each port of the dual port RAM block separately. There is no cross port data coherency checking performed in the dual port RAM block. A read of the dual port RAM block via the DSP port to the last address location written via the DP RAM port will not return correct data if that read is executed prior to a next write on the DSP port to push the late write buffer contents into the SRAM array. The converse read case via the DP RAM port also produces incorrect data.

Since there is no cross port data coherency for the dual port RAM block, the DSP late write buffer and the DPR late write buffer may be in a state where both contain different write data targeting the same destination write address within the dual port RAM block. If a “next write” occurs on the DSP port and the DPR port in overlapping system clock cycles, both late write buffer write data contents are pushed to the same destination address location in the dual port RAM at the same time. In this case, the resulting data in the target write address is indeterminate.

To guarantee cross port data coherency in the dual port RAM block, it is recommended that application software use handshaking protocols, such as semaphore register protocol, when passing data via the dual port RAM block. The following example illustrates how this handshaking protocol could be used to pass data from the DSP port to the DPR port. The sending software task which writes data into the dual port RAM block arbitrates for the write access privileges by successfully updating a semaphore register to indicate that it is loading contents in the dual port RAM. The sending software task writes the data packet into the dual port RAM block via the DSP port. To complete writing the data packet to the dual port RAM block, the sending software task issues a final “next write” to the DSP port of some arbitrarily defined data word to an predefined “write-only” location within the dual port RAM block to push the last word of the data packet from the DSP late write buffer to the dual port memory array. The write-only location is arbitrarily assigned by the software developer for use in data transfer processes involving the dual port

RAM block. Then the sending software task updates the semaphore register to indicate that the data packet is successfully loaded into the dual port RAM block. At this point, the receiving software task can observe that the sending software task has completed writing data. The receiving software task begins retrieving the data packet from the dual port RAM block by updating a semaphore register to indicate its intent to fetch the data packet, reading the data packet from the dual port RAM block via the DPR port, and then updating the semaphore register to indicate completion of fetching the data packet. Data transfers from the DPR port to the DSP port may be executed in the same manner by using the semaphore protocol to send data to the dual port RAM block via the DPR port and retrieving data from the dual port RAM block via the DSP port.

8.2.2.2 Interleaved RAM Block

The interleaved RAM block is a dual port SRAM block constructed by partitioning a single port SRAM into eight independent subblocks as shown in the sample configuration in Figure 8-3. Addressing of the interleaved RAM memory is implemented in a modulo 8 fashion. The starting address of the interleaved RAM section is $400 and is mapped into the first subblock. The next sequential address is mapped into the next subblock. Once the last block is reached, sequential addresses wrap from the last subblock (Subblock 7) back to the first subblock (Subblock 0). Addresses are counted across the eight subblocks in this fashion until the last address N, where N is the last address of the interleaved RAM block.

Subblock 0	Subblock 1	Subblock 2	Subblock 3	Subblock 4	Subblock 5	Subblock 6	Subblock 7
N-7	N-6	N-5	N-4	N-3	N-2	N-1	N
N-15	N-14	N-13	N-12	N-11	N-10	N-9	N-8
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$0420	$0421	$0422	$0423	$0424	$0425	$0426	$0427
$0418	$0419	$041A	$041B	$041C	$041D	$041E	$041F
$0410	$0411	$0412	$0413	$0414	$0415	$0416	$0417
$0408	$0409	$040A	$040B	$040C	$040D	$040E	$040F
$0400	$0401	$0402	$0403	$0404	$0405	$0406	$0407

Figure 8-3. Interleaved RAM Block Partitioning

Concurrent accesses from the DSP port and the Interleaved RAM Port (IRP) which target different subblocks of the interleaved RAM block are supported. In the event that concurrent DSP port and IRP accesses target the same subblock, the DSP port takes precedent and a contention signal is asserted on the IRP to indicate that the IRP master has been denied access to the interleaved RAM portion of the data memory. It is up to the IRP master to take action in the event of a denial of memory access.In addition to the contention conditions that have just been described, there are some additional sequences that can cause internal contention inside the RAM array. Both external and internal contention sequences are shown below:

External contention:

If the 3 low order address bits of the DSP Port and the IRP Port match, the contention signal will be asserted as long as any of the following combinations occur:

a) DSP (Read)/IRP (Read), b) DSP (Write)/IRP (Read)

c) DSP (Read)/IRP (Write) d) DSP(Write)/ IRP (Write)

Internal Contention:

If the 3 low order address bits in the late write buffer (i.e. wr_coh_buffer), and the external address bus match, contention will occur for the sequences shown below:

a) DSP (Read)/IRP (wr_coh_buffer), b) DSP (wr_coh_buffer)/IRP (Read)

c) DSP (wr_coh_buffer)/IRP (wr_coh_buffer)

8.2.2.3 ROM Block

The ROM block is a single port memory which is accessible only through the DSP port of the data memory.

8.3 Initialization/Application Information

The ROM partitions of the DSPMEM are initialized as part of the silicon fabrication process while the SRAM partitions are initialized by writing valid data into SRAM address locations prior to reading those locations.

Chapter 9
DSP Clock Generator (DCLKG_LOGIC)

Revision History

Revision	Date	Author	Changes
0.1	11/29/00	Moshe Refeali	Initial Release.
0.2	2/19/01	Shannon Osgood	Made applicable updates for v1.9 release.
0.3	02/14/01	Alon Eldar	Major Update
0.4	03/29/01	Alon Eldar	Defined software restrictions Added details and block diagram.
0.5	08/21/01	Alon Eldar	DCLKG_LOGIC pass 2 changes update
0.6	09/11/01	Alon Eldar	Format adjustments for Neptune spec.
0.7	12/21/01	David Liddle	Updated for Patriot Included PCTL registers and outputs to PLL Muxing test clocks onto gclkw, gclk Also outputting gclkw_b and gclk_b
0.8	1/10/02	David Liddle	Changed COD bit of PCTL1 register to reserved. Updated the Interface Changed PAT_REFx2 to PAT_REF for consistency to previous specs.
0.9	1/18/02	David Liddle	Removed the test_clk signal, PAT_REF should be used as the test_clk in scan mode. Clarified the MLM bit usage. Clarifed what happens when the MFN, MFI, MFD bits are updated.
1.0	2/8/02	David Liddle	Clarified the restrictions and when they apply to a Patriot configuration.
1.1	4/17/02	David Liddle	Updates after spec review. Clarified configurability by adding in conditional text for different configurations.
1.2	6/18/02	David Liddle	Updated for Neptune LTS.
1.3	6/18/02	David Liddle	Updated after Review in Adelaide.
1.4	4/30/03	David Liddle	Updated per DDTS# DSPhl15962, DSPhl16045, and DSPhl16047.
1.5	05/07/03		Updated for LTE specification release and DDTS DSPhl15841.

9.1 Introduction

The DSP clock generator (DCLKG_LOGIC) supplies clocks to SOnyxU subsystem.

DCLKG_LOGIC performs:

- Clock Pulse generation for SOnyxU subsystem.
- Low power division.

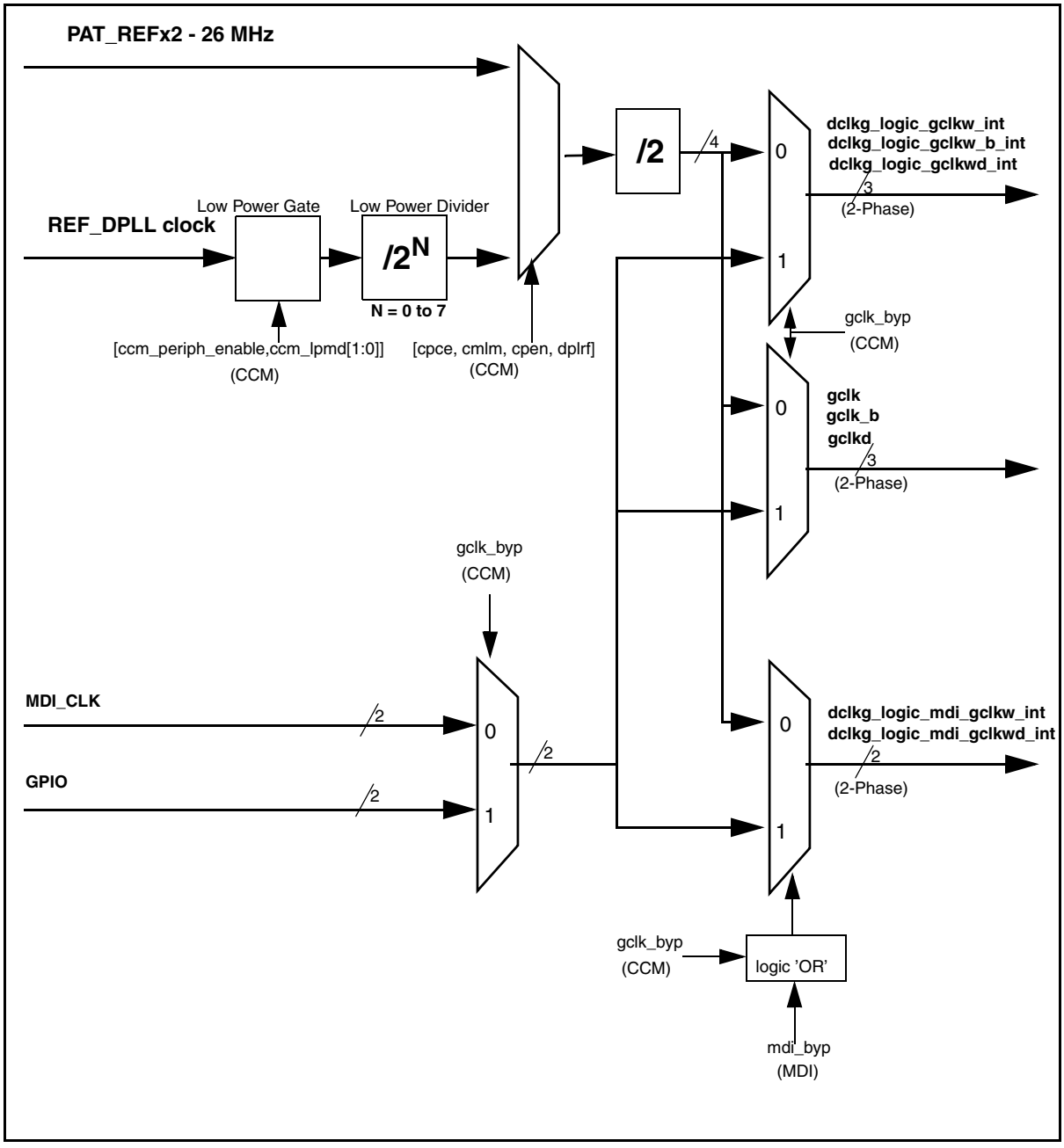


Figure 9-1. DCLKG_LOGIC Clock Sources and Clock Outputs

9.2 DCLKG_LOGIC Module Pin List

Table 9-1. DCLKG_LOGIC Module Pin List

Pin Name	Direction	Description	Remarks
CCM Interface			
dplrf	I	Dpll Lock Ready Flag	
ccm_periph_enable	I	Low Power Gate Enable	When ccm_periph_enable='0', low power gate is enabled only if ccm_lpmd[1:0] = 2'b0.
ccm_lpmd[1:0]	I	Low Power Gate Mode	
Dpll Interface			
dpdck	O	Dpll Output Clock	
dclkg_cbrmo	O	BRM Order bit to PLL	
dclkg_cpen	O	PLL Enable signal to PLL	
dclkg_cpd[3:0]	O	Pre-divide factor to PLL	
dclkg_mfi[3:0]	O	Multiplication Factor Integer component to PLL	
dclkg_mfn[3:0]	O	Multiplication Factor Numerator component to PLL	
dclkg_mfd[3:0]	O	Multiplication Factor Denominator component to PLL	
Core Interface			
pwait	I	Processing Wait	
pint	I	Processing Interrupt	
ndreq	I	Debug Request	
pstop	I	Processing Stop	
paomr6	I	OMR Bit 6	Effective only when cpstp negated. 0 = long recovery when exiting DSP stop state 1 = short recovery when exiting DSP stop state
cwait	O	Clock Wait	
clk_dis	O	Clock Disable for Wait	clk_dis signal connection from DCLKG_LOGIC to SOnyxU core should have the same* route delay as gclkw
mcdis0	I	Core Stall Phase 0	

Table 9-1. DCLKG_LOGIC Module Pin List

Pin Name	Direction	Description	Remarks
njdreq_b	I	JTAG DEBUG_REQUEST Instruction	
SPMBIF Interface			
mcab_wr[6:0]	I	Core Write address bus	
mcab_rd[6:0]	I	Core Read address bus	
mcsel_wr	I	Core write select	
mcsel_rd	I	Core read select	
gdb_wr[15:0]	I	Core Global Write data bus	
gdb_rd[15:0]	O	Core Global Read data bus	
mcdis1	I	Core pipeline stall Indication (used in SPMBIF only, hidden for the user)	
spmbif_scan_tristate_enable	I	Enable the tristates during scan mode	This is a scan mode signal
Clock Source Interface			
dclkg_logic_gclkw_int	O	Global Clock during Wait	
dclkg_logic_gclkw_b_int	O	Inverted Global Clock during Wait	New pin added for Patriot
gclk	O	Global Clock	not used in Neptune top level
gclk_b	O	Inverted Global Clock	New pin added for Patriot
gclkd	O	Global Clock Delayed	not used in Neptune top level
dclkg_logic_gclkwd_int	O	Global Clock during Wait Delayed	
dclkg_logic_mdi_gclkw_int	O	Global Clock for MDI Shared Memory	
dclkg_logic_mdi_gclkwd_int	O	Global Clock Delayed for MDI Shared Memory	
mdi_clk	I	External MDI Clock Input for Bypass	
mdi_clkd	I	External MDI Delayed Clock Input for Bypass	
mdi_byp	I	MDI Clock Bypass Control	
I/O Interface			

Table 9-1. DCLKG_LOGIC Module Pin List

Pin Name	Direction	Description	Remarks
extal	I	External Clock	This will also be used for the test clock in scan mode.
irs_b	I	System Hardware Reset	not used internally
res	I	Core Reset	functional reset
por_res	I	Power Up Reset	
irqmda_b	I	Interrupt Request A	
idreq_b	I	Debug Event	
Configuration Interface			
int_pll_ctl	I	Enable the Internal PLL controls	This should be tied to 1 if the DCLKG needs to control the DSP PLL. If the DSP PLL is controlled by the CCM then this should be tied to 0.
Test Interface			
scan_mode	I	Scan Mode	
scan_en	I	Scan Enable	
gclk_byp	I	Clock Bypass Control	
gpio_gclk	I	External Clock Input for Bypass	
gpio_gclkd	I	External Delayed Clock Input for Bypass	

* In the current process, same delay within ~0.5 ns.

9.3 DCLKG_LOGIC Internal Blocks

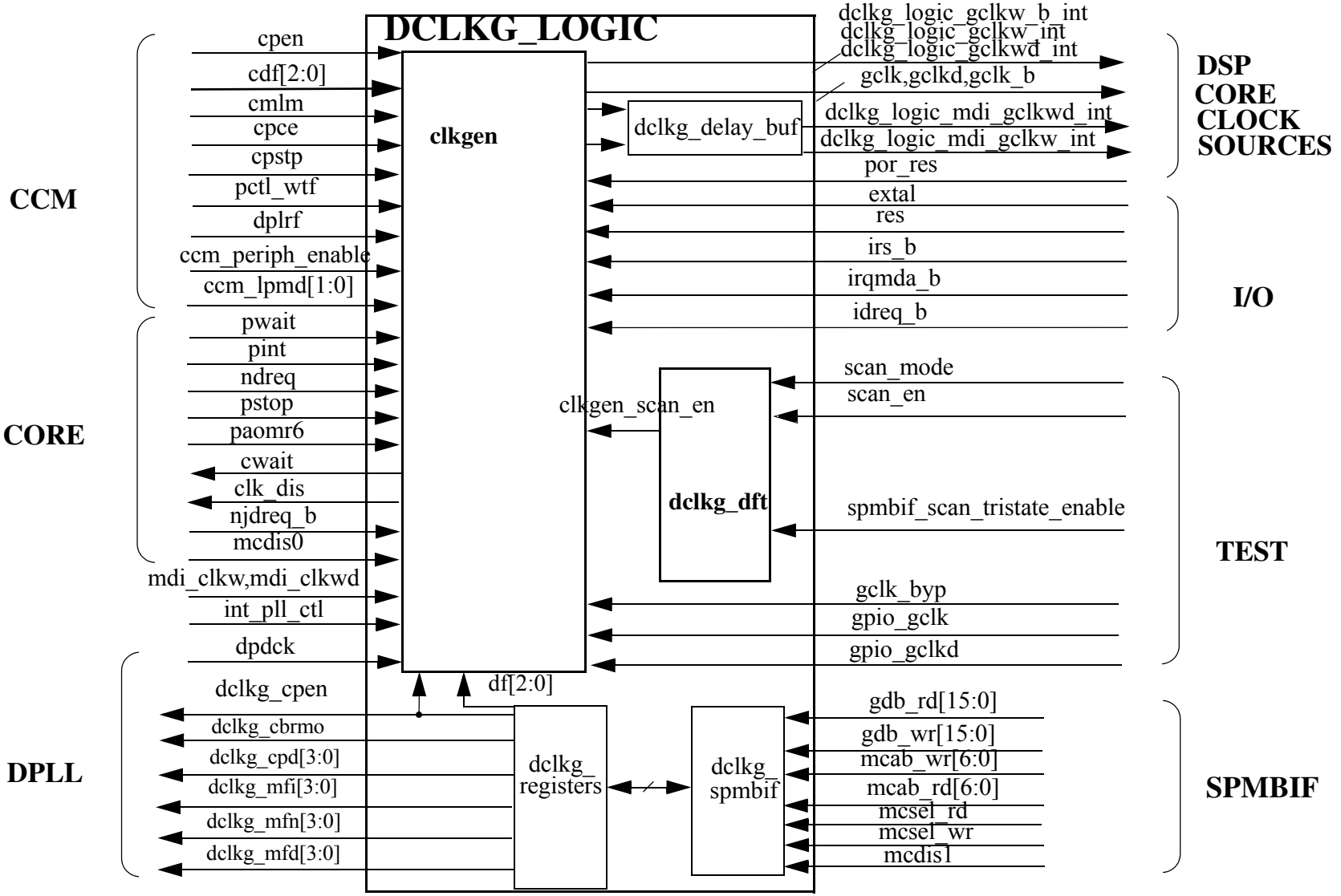
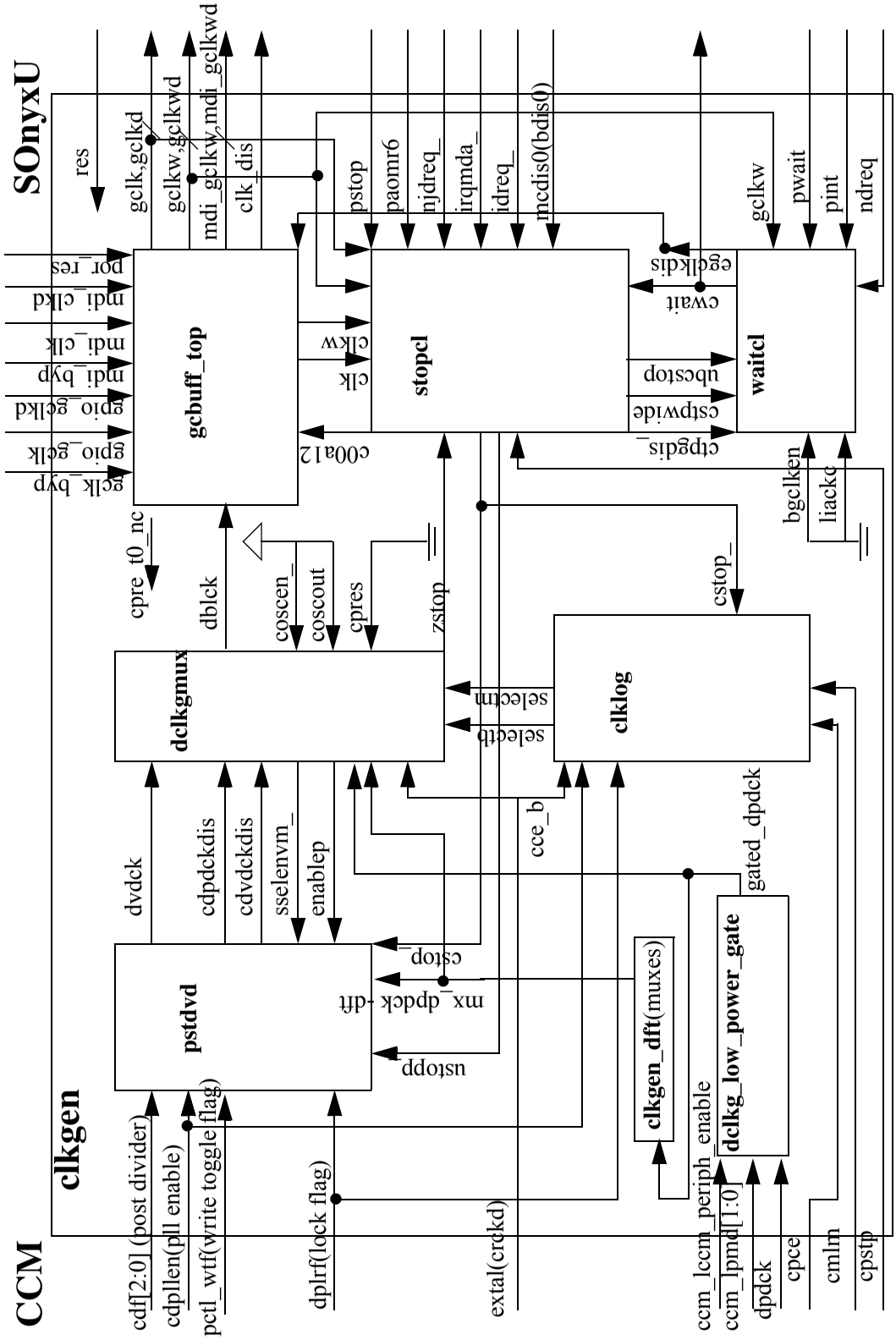


Figure 9-2. DCLKG_LOGIC General Block Diagram



* **Regarding DFT muxes, only dpdck and jstop are marked as muxed, but there are more signals that are muxed and are not shown.**

Figure 9-3. DCLKG_LOGIC Micro Architecture

9.3.1 Internal Blocks Functionality

9.3.1.1 DCLKG_LOGIC

dclkg_delay_buf contains library buffer cells which delay the mdi shared memory clocks in a way which makes its timing more similar to the other clock outputs.

dclkg_dft handles the scan data out bypassing, scan divergence, and ANDs scan_mode with scan_en to create the internal scan enable signal.

clkgen contains all the functional blocks of DCLKG_LOGIC.

dclkg_spmbif contains the SPMB Interface.

dclkg_registers contains the PCTL registers

9.3.1.2 clkgen

clkgen_dft handles the clock multiplexing and contains additional logic to improve observability and controllability. The multiplexing of the clocks is done on the signals just before they go to the Flip-Flop's clock inputs. In cases where those clocks go through additional logic before clocking a Flip-Flop they are not replaced with test_clk.

The **pstdvd** block is the low power clock post divider. The cdf[2:0] bits specify the division factor of the clock, as a power of two, in the range of 2^0 to 2^7.

gcbuff_top contains the clock buffers which divide the clocks by two. The clock bypass mux cells which appear in figure 9-1 are also part of this block.

clklog is clock source select logic according to cpen, cmlm, cpce and dplrf.

dclkmux is the main clock mux between dpdck, extal and dvdck (divided clock) from pstdvd.

stopcl is responsible for handling clocks during DSP stop state, enable an disable clocks according to the status of stop state.

waitcl is responsible for handling clocks during DSP wait state, enable an disable clocks according to the status of wait state.

dclkg_low_power_gate is responsible to gate on and off the 'dpdck' clock input port.

dclkg_pll_ctrl is responsible for selecting which signals get used to control the dclkg behavior (either from the PCTL registers or from pins connected to the CCM). It is also responsible for generating the signals to the PLL to control the PLL operation.

9.4 Modes of Operation

- Power Up Reset

  During power up reset DCLKG_LOGIC is initialized and do not generate any clocks to SOnyxU subsystem. De-assertion of power up reset is synchronized to PAT_REF clock domain. PAT_REF should be activated and oscillate at least three cycles before 'por_res' is negated. Power up reset mode should always be followed by functional reset mode.

- Functional Reset

During hardware reset SOnyxU subsystem clocks are derived directly from PAT_REF clock. During this mode the frequency of SOnyxU subsystem clocks is half of PAT_REF clock frequency.

- GPIO Clock Bypass

  SOnyxU subsystem clocks are derived directly from GPIO clocks at a special test mode (memory BIST). The mux which is controlling the GPIO clock bypass is a simple mux (not a glitchless one), and it is the system's responsibility to enable clock bypass in an appropriate time in order to prevent errors due to glitches. Note that at this mode DCLKG_LOGIC is not synchronized to the rest of SOnyxU subsystem.

- MDI_CLK Clock Bypass

  When GPIO clock bypass is disabled, it is possible to generate mdi_gclkw and mdi_gclkwd clocks directly from mdi_clk and mdi_clkd clock inputs. These output clocks reach the internal memory, which may be accessed by the MDI during DSP stop state. The mux which is controlling the clock bypass is a simple mux (not a glitchless one), and it is the system's responsibility to enable clock bypass in an appropriate time in order to prevent errors due to glitches.

- Test Mode

  The DCLKG_LOGIC module supports the following features:

  — The res signal is treated as the test reset during test mode. All the flip-flops in DCLKG_LOGIC can be reset by the res signal during test mode.

  — There are no dedicated scan ports in the DCLKG_LOGIC. They will be added during sonyxu platform scan insertion.

  — The PAT_REF (extal) signal will be used as the test clock in scan mode.

  — The DCLKG_LOGIC module has 9 different clock domains which are multiplexed with the test clock for scan.

  — The global clocks will be muxed with the test clock during scan mode for distribution to the DSP sub-system such that all functional clocks are in phase.

- Functional mode

  Please refer to Table 9-1 for ports names and Figure 5-4 in CCM chapter for clock names.

  PAT_REF_CLK clock input select:

  When PAT_REF is the selected clock input, the 2-Phase DSP clocks frequency is half of PAT_REF clock frequency.

  PAT_REF is the selected clock input at the following cases:

  — PCTL1/PEN is negated.

  — PCTL1/MLM is asserted while PCTL1/PCE is negated.

  REF_DPLL clock input select:

  When REF_DPLL clock is the selected clock input, the 2-Phase DSP clocks frequency is REF_DPLL clock frequency divided by (2*DF). DF is the Division Factor $2^{cdf[2:0]}$. While REF_DPLL clock is the selected clock input, the 2-Phase DSP clocks will oscillate as long as DPLRF is high, and stuck otherwise.

  REF_DPLL clock is the selected clock input at the following cases:

  — PCTL1/MLM is negated while PCTL1/PEN is asserted.

  — PCTL1/MLM, PCTL1/PCE and PCTL1/PEN are asserted.

  DSP Stop State:

SOnyxU subsystem will enter ’stop’ state by DSP ’stop’ command. In this state dclkg_logic_gclkw_int, dclkg_logic_gclkwd_int, gclk and gclkd clocks are disabled and stuck at low level. dclkg_logic_mdi_gclkw_int and dclkg_logic_mdi_gclkwd_int are generated from mdi_clk and mdi_clkd accordingly, while mdi_byp control is asserted. There are several ways to exit DSP stop state: hardware reset, interrupt request from pin, or debug request.

DSP Wait State:

SOnyxU subsystem will enter ’wait’ state by DSP ’wait’ command. In this state gclk and gclkd clocks are disabled and stuck at low level, while dclkg_logic_gclkw_int, dclkg_logic_gclkwd_int, dclkg_logic_mdi_gclkw_int and dclkg_logic_mdi_gclkwd_int are enabled. There are several ways to exit wait state: processing interrupt, debug request or reset.

Low Power Mode:

DCLKG_LOGIC ‘dpdck’ clock input may be gated off by a low power gate which is implemented in the module. The controls of the low power gate are driven asynchronously by the CCM and synchronized internally to ‘dpdck’ clock domain. ‘dpdck’ input port will be gated off during Low Power Mode which is recognized when ‘ccm_lpmd[1:0]’ and ‘ccm_periph_enable’ are all zeros. The user should put DCLKG_LOGIC in its DSP Stop State prior to entering Low Power Mode, and keep it in DSP Stop State until exiting Low Power Mode. This functionality is only enabled when int_pll_ctl is 0.

9.5 Programming Model

The DCLKG_LOGIC module contains the PLL control registers PCTL0 and PCTL1 that used to reside in the original OnyxU design. They should be accessed the same way they were in previous versions of Patriot.

In an environment where the DSP PLL is controlled by the CCM (as in Neptune), writing to PCTL0 and parts of PCTL1 will have no effect on the system clock.

The Register Summary lists all registers of the module by ascending address. The absolute address of each register is given, and the value of each bit for reads and writes is given using the conventions in the Legend.

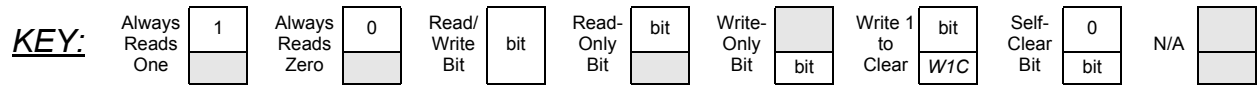


Table 9-2. DSP DPLL Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCTL0 (X:$FFFD)	R																
	W																
PCTL1 (X:$FFFC)	R	LRF			MLM	PCE					PEN	PSTP			DF[2:0]		
	W																

NOTE:

The clock to the above registers will automatically be disabled during DSP WAIT mode.

9.5.1 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various DSP DPLL registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

The DPLL control register 0 (PCTL0) is a 16 bit read/write register used to direct the operation of the DSP DPLL. The PCTL0 control bits are described in Table 9-3.

PCTL0 DSP DPLL Control Register 0 **Addr X:$FFFD**

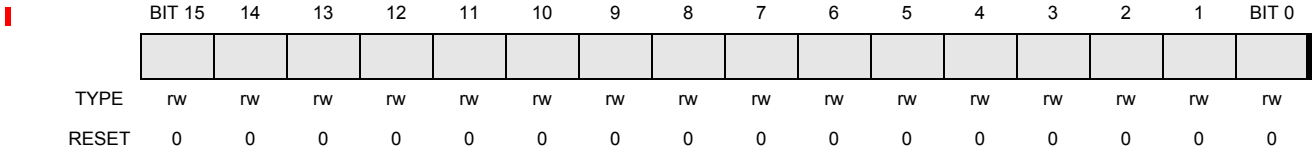


Table 9-3. PCTL0 Description

Name	Description	Settings
Bits 15-0	**Reserved Bits —** These bits are read/write bits that have no effect on system functionality. They should be written as 0 for future compatibility.	N/A

The DPLL control register 1 (PCTL1) is a 16 bit read/write register used to direct the operation of the DSP DPLL and Clock-Generator. The PCTL1 control bits are described in Table 9-4.

PCTL1 DSP DPLL Control Register 1 **Addr X:$FFFC**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	DPLRF			MLM	PCE					PEN	PSTP			DF[2:0]		
TYPE	r	r	r	rw	rw	rw	r	r	r	rw	rw	r	r	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-4. PCTL1 Description

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>DPLRF
Bit 15</td><td>Lock Ready Flag Bit-- The RLF bit is a flag that is set when the PLL is enabled and has locked on the proper phase. The RLF bit is cleared if the DSP-DPLL is not locked.</td><td>0 = DPLL not locked (default).
1 = DPLL locked on proper phase</td></tr>
<tr><td>Bits 14-13</td><td>Reserved Bits — These bits are read as 0 and should be written as 0 for future compatibility.</td><td>N/A</td></tr>
<tr><td>MLM
Bit 12</td><td>Manual Lock Mode Bit — The MLM bit controls the manner of switching the DSP-CLK from the PAT_REF/2 to the PLL output clock. When this bit is set, the switching will be performed by DSP software through the PCE bit of the PCTL1 register. When this bit is cleared, the DSP-CLK is replaced automatically after the PLL lock condition has been fulfilled. The bit should not be changed if the PLL is enabled. The MLM bit is cleared by hardware reset.</td><td rowspan="2"><table><tr><th>MLM</th><th>PCE</th><th>CLKGEN input clock</th></tr><tr><td>0</td><td>x</td><td>PAT_REF before PLL lock / PLL Output after PLL lock</td></tr><tr><td>1</td><td>0</td><td>PAT_REF</td></tr><tr><td>1</td><td>1</td><td>PLL Output</td></tr></table></td></tr>
<tr><td>PCE
Bit 11</td><td>PLL Clock Enable Bit — The PCE bit controls switching of the DSP-CLK from PAT_REF/2 to the PLL output clock in the manual mode, when the MLM bit is set. The DSP Core can force switching through this bit. When the PCE bit is set, the output clock is produced by the PLL. When the bit is cleared, the PAT_REF feeds the DSP CLKGEN block. If the MLM bit is cleared, the PCE bit is ignored. This bit should not be updated in the same operation that the MLM bit is updated. The PCE bit is cleared by hardware reset or if the PLL is disabled.</td></tr>
<tr><td>Bit 10</td><td>Reserved Bit — This bit is a read/write bit that has no effect on system functionality. This should be written as 0 for future compatibility.</td><td>N/A</td></tr>
<tr><td>Bits 9-7</td><td>Reserved Bits — Those bits should be written as 0 for future compatibility.</td><td>N/A</td></tr>
</table>

Table 9-4. PCTL1 Description

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>PEN
Bit 6</td><td>PLL Enable Bit — The PEN bit needs to reflect the enable/disable state of the PLL. Writing to this bit will have no effect on the PLL. If this bit is 0, then the PLL clock will not be used. If this bit is 1, then the PLL clock can be used depending on the programming of the MLM and PCE bits. The PEN bit may be set or cleared by software any time during the chip operation. During hardware reset this bit is set to 0, i.e. the PLL is disabled.</td><td>0 = DPLL cannot be selected.
1 = DPLL can be selected.</td></tr>
<tr><td>PSTP
Bit 5</td><td>STOP Processing State Bit — This bit, along with the SD bit in the OMR register, determines the DSP wakeup time when exiting from STOP mode. This bit is cleared by hardware reset.</td><td>
<table>
<tr><th>PSTP</th><th>OMR/SD</th><th>Delay When Exiting STOP Mode</th></tr>
<tr><td>o</td><td>o</td><td>128k cycles</td></tr>
<tr><td>o</td><td>1</td><td>16 cycles</td></tr>
<tr><td>1</td><td>0</td><td>2 cycles</td></tr>
<tr><td>1</td><td>1</td><td>2 cycles</td></tr>
</table>
</td></tr>
<tr><td>Bit 4-3</td><td>Reserved Bit — This bit is read as 0 and should be written as 0 for future compatibility.</td><td>N/A</td></tr>
<tr><td>DF[2:0]
Bits 2-0</td><td>Division Factor Bits— The Division Factor bits DF2-DF0 define the divide factor DF of the low power divider. These bits specify any power of two divide factor in the range from 2^0 to 2^7. They are used also in order to generate the needed DSP-CLK frequency from the DPLL output frequency which must be less than130MHz. The table to the right shows the programming of the DF2-DF0 bits. Changing the value of the DF2-DF0 bits will not cause a loss of lock condition. Changing DF2-DF0 may lengthen the instruction cycle following the PLL control register update; this is done in order to keep synchronization between PAT_REF and the internal chip clock. These bits are cleared (division by one) by hardware reset.</td><td>
<table>
<tr><th>DF2-DF0</th><th>Division Factor DF</th></tr>
<tr><td>$0</td><td>$2^0$</td></tr>
<tr><td>$1</td><td>$2^1$</td></tr>
<tr><td>$2</td><td>$2^2$</td></tr>
<tr><td>•</td><td>•</td></tr>
<tr><td>$7</td><td>$2^7$</td></tr>
</table>
</td></tr>
</table>

9.6 Restrictions

DCLKG_LOGIC can work in an automatic mode (MLM=0) with the following limitations: (These restrictions do not apply if int_pll_ctl is tied to 1.)

1. After the assertion of the PCTL1/PEN bit pin the 2-Phase DSP clocks will stop. DCLKG_LOGIC can not guarantee at which of the two possible states - low or high state, the clock is stopped in. The clock will start oscillating after the lock flag DPLRF at the DCLKG_LOGIC input port is asserted and internally

synchronized and proper operation of a state machine is achieved. While PCTL1/PEN is enabled at the DCLKG_LOGIC input port, the 2-Phase DSP clocks will oscillate as long as DPLRF is high, and stopped otherwise.

Following is a list of restrictions which allow proper operation of DCLKG_LOGIC:

1.

When DCLKG_LOGIC active clock input is REF_DPLL clock, the following apply:

Negation of PCTL1/PEN will cause the selection of PAT_REF input clock.

Negation of DPLRF while PCTL1/PEN is asserted will stop the DSP clocks (at low or high state).

Assertion of PCTL1/MLM while PCTL1/PCE is negated will cause the selection of PAT_REF input clock.

Negation of PCTL1/PCE while PCTL1/MLM is asserted will cause the selection of PAT_REF input clock.

For proper operation of DCLKG_LOGIC (regardless of DSP state) any of the events above should be captured ,'N' cycles of PAT_REF before REF_DPLL clock stops.

'N'=((((3x (2^cdf)) +4)x pdf)/(2x mf))+2.

cdf=Division Factor Bits DSP_CG_CTL[2:0].

pdf=Pre Division Factor of DPLL.

mf=Multiplication Factor of DPLL.

2.

When PAT_REF is the selected input, switching to REF_DPLL clock can be done in automatic or manual mode.

In automatic mode (PCTL1/MLM=0), assertion of PCTL1/PEN will stop the DSP clocks (at low or high state). Assertion of DPLRF will cause DCLKG_LOGIC to switch to REF_DPLL clock input.

In manual mode (PCTL1/MLM=1, PCTL1/PCE=0) assertion of PCTL1/PEN and DPLL Lock will not change the active input of DCLKG_LOGIC.

Assertion of PCTL1/PCE (while PCTL1/MLM=1) or negation of PCTL1/MLM, while DPLRF is negated and PCTL1/PEN is asserted will stop the DSP clocks (at low or high state).

Assertion of PCTL1/PCE (while PCTL1/MLM=1) or negation of PCTL1/MLM, while PCTL1/PEN and DPLRF are already asserted will cause DCLKG_LOGIC to switch to REF_DPLL clock.

While PCTL1/PEN is negated the active input of DCLKG_LOGIC will remain PAT_REF, regardless of PCTL1/MLM or PCTL1/PCE values.

3.

a. Once PAT_REF is selected, it should remain selected for at least 'N' cycles of PAT_REF. The parameter 'N' is the same as mentioned in paragraph 1.

b. Once REF_DPLL clock is selected, it should remain selected for at least ’M’ cycles of PAT_REF.

’M’=((((6.5x (2^cdf)) +4)x pdf)/(2x mf))+2.

cdf=Division Factor Bits DSP_CG_CTL[2:0].

pdf=Pre Division Factor of DPLL.

mf=Multiplication Factor of DPLL.

4.

When programming the PCTL1/MLM bit (going from 0 to 1) the PCTL1/PCE should not be updated.

This means that when the MLM bit of the PCTL1 register is changed from 0 to 1, the PCE bit of PCTL1 should not be updated.

5.

Note that DCLKG_LOGIC is not aware what is the real source of REF_DPLL clock in case it is muxed before DCLKG_LOGIC. All relevant limitations apply even if such mux selects other clock source.

Chapter 10
DSP Special Interrupt Handler (DSIH)

Revision History Table

Revision	Author	Date	Changes
0.0	Satish Chandra Pandey	01.27.2000	Initial
0.1	Dilip Singh	05.31.2000	Changed signal names as per Patriot guidelines
0.2	Balamurugan Selvaraj	10.17.2001	Modified for Patriot-Indy
0.3	Balamurugan Selvaraj	11.01.2001	Added mmc connectivity
0.4	Rakesh Adhikari	19.03.2002	Initial version taken from Patriot Indy,Since MMC module is not there in Neptune. Changed MMC to VOCOD.
0.5	05/07/03		Updated for LTE specification release.

10.1 Introduction

The DSP56600 Core is limited to eight peripheral maskable interrupts sources.

As the number of peripherals are more than eight there is a interrupt handler required.

DSIH(DSP Special Interrupt Handler) module prioritizes the interrupts coming from the peripheral modules and sends to the DSP to serve them.

10.2 Pin Description

The DSIH pinouts is shown in the figure 1 and described by table 1.

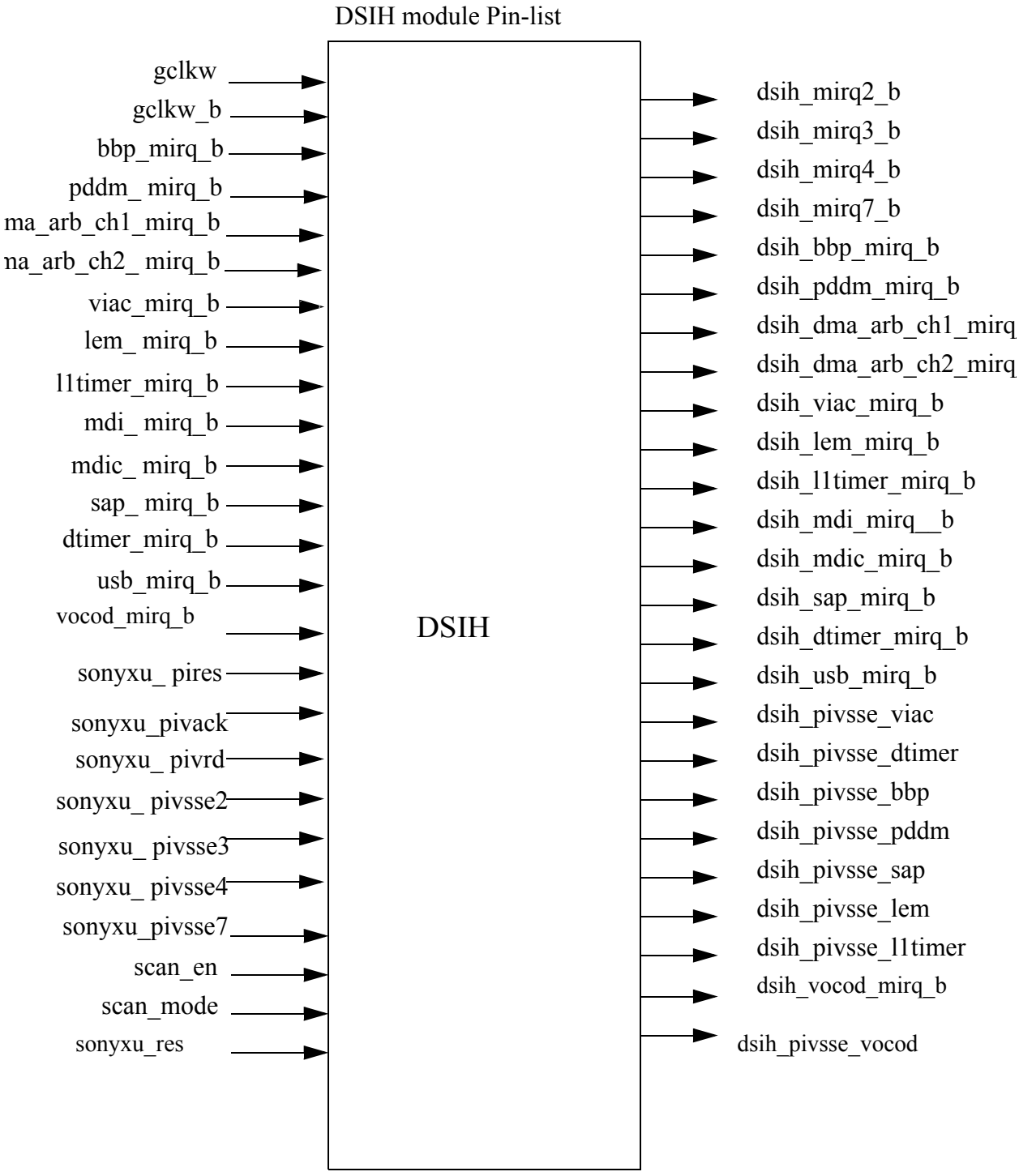


Figure 10-1.

Table 10-1. DSIH Pin List

NAME	TYPE	DESCRIPTION
gclkw	I	Input clock
gclkw_b	I	Input inverted clock
bbp_mirq_b	I	Interrupt request from Base Band Port
pddm_mirq_b	I	Interrupt request from DSP Debug Module
dma_arb_ch1_mirq_b	I	Interrupt request from DMA1 Module
dma_arb_ch2_mirq_b	I	Interrupt request from DMA2 Module
lem_mirq_b	I	Interrupt request from Layer1 Encryption Module
l1timer_mirq_b	I	Interrupt request from Layer1 Timer
mdi_mirq_b	I	Interrupt request from MCU DSP Interface Module
mdic_mirq_b	I	Interrupt request from MCU DSP Interface Module
sap_mirq_b	I	Interrupt request from Serial Audio Codec Port
dtimer_mirq_b	I	Interrupt request from DSP Timer
usb_mirq_b	I	Interrupt request from Universal Serial Bus Interface
viac_mirq_b	I	Interrupt request from Viterbi Accelerator
sonyxu_pires	I	Peripheral reset
sonyxu_res	I	Hardware reset
sonyxu_pivack	I	Peripheral Module Interrupt Vector Acknowledge
sonyxu_pivrd	I	Interrupt Vector Read Strobe
sonyxu_pivsse3	I	Interrupt vector source select signal. dsih_pivsse_pddm, dsih_pivsse_sap , can be generated only when, this signal is enabled.
sonyxu_pivsse4	I	Interrupt vector source select signal . dsih_pivsse_dtimer , dsih_pivsse_l1timer can be generated only when, this signal is enabled.
sonyxu_pivsse7	I	Interrupt vector source selectsignal. dsih_pivsse_lem and dsih_pivsse_viac can be generated only when, this signal is enabled.
ipt_clk_se	I	Scan enable signal
ipt_scan_mode	I	Scan mode signal
vocod_mirq_b	I	vocod interrupt request
dsih_mirq3_b	O	This signal is generated only when we have interrupt request from SAP orVOCOD or PDDM or all of them..

Table 10-1. DSIH Pin List

NAME	TYPE	DESCRIPTION
dsih_mirq4_b	O	This signal is generated only when we have interrupt request from DTIMER or L1TIMER or both.
dsih_mirq7_b	O	This signal is generated only when we have interrupt request from LEM or VIAC or both.
dsih_bbp_mirq_b	O	Buffered interrupt request from Base Band Port
dsih_pddm_mirq_b	O	Buffered interrupt request from DSP Debug Module
dsih_dma_arb_ch1_mirq_b	O	Buffered interrupt request from DMA1 Module
dsih_dma_arb_ch2_mirq_b	O	Buffered interrupt request from DMA2 Module
dsih_lem_mirq_b	O	Buffered interrupt request from Layer1 Encryption Module
dsih_l1timer_mirq_b	O	Buffered interrupt request from L1 TIMER
dsih_mdi_mirq__b	O	Buffered interrupt request from MDI
dsih_mdic_mirq_b	O	Buffered interrupt request from MDIC
dsih_sap_mirq_b	O	Buffered interrupt request from SAP
dsih_dtimer_mirq_b	O	Buffered interrupt request from DTIMER
dsih_usb_mirq_b	O	Buffered interrupt request from USB
dsih_viac_mirq_b	O	Buffered interrupt request from VIAC
dsih_pivsse_bbp	O	Interrupt vector source select for BBP
dsih_pivsse_pddm	O	Interrupt vector source select for DSP Debug Module
dsih_pivsse_lem	O	Interrupt vector source select for LEM
dsih_pivsse_l1timer	O	Interrupt vector source select for L1TIMER
dsih_pivsse_sap	O	Interrupt vector source select for SAP
dsih_pivsse_dtimer	O	Interrupt vector source select for DTIMER
dsih_pivsse_viac	O	Interrupt vector source select for Veterbi Accelerator
dsih_vocod_mirq_b	O	Buffered interrupt request from vocod
dsih_pivsse_vocod	O	Interrupt vector source select for vocod

10.3 Functional Description

DSIH resolves the priority between more than one interrupt requests recieved.

If more than one peripheral sends an interrupt request , DSIH sends interrupt vector source select signal (dsih_pivsse_<module>) to the module that has a higher priority .

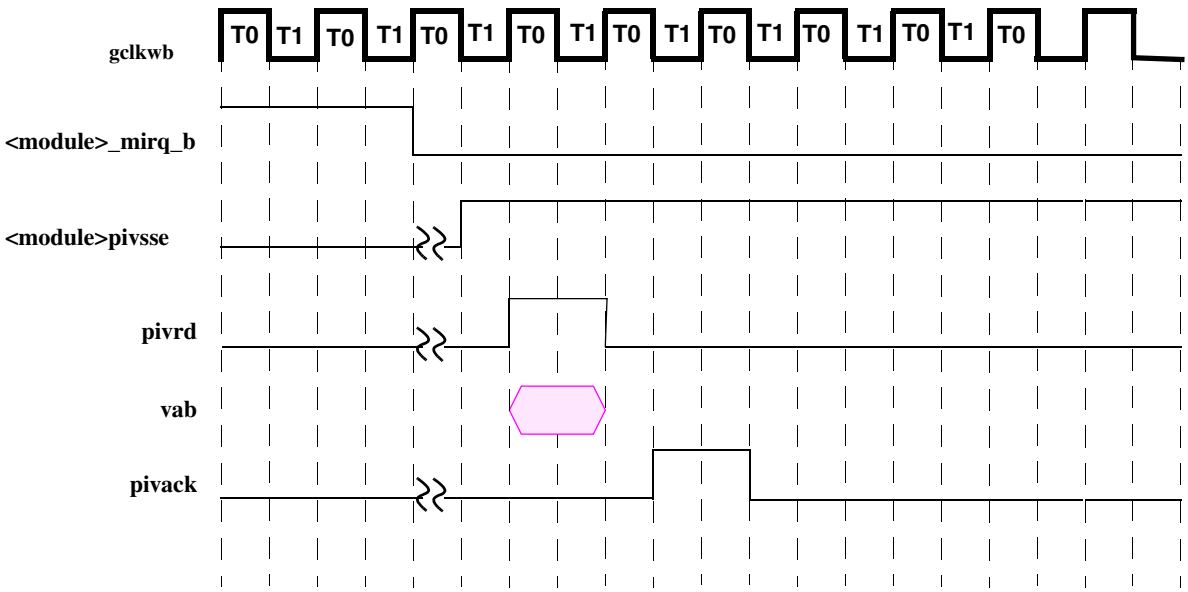
In addition to this DSIH buffers the interrupt requests (<module>_mirq) from different modules and sends the buffered interrupts (dsih_mirq_<module>) to the GPIO.

Table 10-2.

Module1	Module2	Module3	Decreasing Priority Order
VIAC	LEM	-	VIAC - LEM
L1TIMER	DTIMER	-	L1TIMER - DTIMER
SAP	VOCOD	PDDM	SAP -VOCOD- PDDM

10.4 Timing Diagram

The timing diagram drawn below shows the timing of various input signals and the source select signal generated by the DSIH block.



Note: gclkwb = ~ gclkw

Figure 10-2.

Chapter 11
Power on Reset (POR)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	05/07/03		Updated for LTE specification release.

11.1 General Description

The POR circuit will provide a reset upon Power Up to be used in certain circuits in the chip that need to be working during RESET. The output of this circuit will also be OR'ed in the chip Reset circuit to activate the chip reset during power up.

The Power On Reset Circuit will be connected to the CORE supply voltage.

This circuit will have three main elements:

- POR - Power On Reset System that recognizes the power up.
- A ring oscillator to generate clocks during the POR period.
- A Counter to count the chip system clock pulses.

11.2 Functional Description

The POR Recognition Circuit is active when the supply is below a certain predetermine value. As long as this signal is asserted the counter is in reset and it's output is "1". Only after the POR Circuit Recognition signal is negated the counter starts to count 16 pulses of the internal oscillator. The output of the counter remains "1" till the end of the count, than it is negated to "0" and the counter is disabled. At the same time the internal oscillator will be also shut off.The Power On Reset signal is the output of the counter.

One additional output of the POR circuit will be the output of the Ring Oscillator. This output can be used in places that need clock to propagate the reset signal through the logic

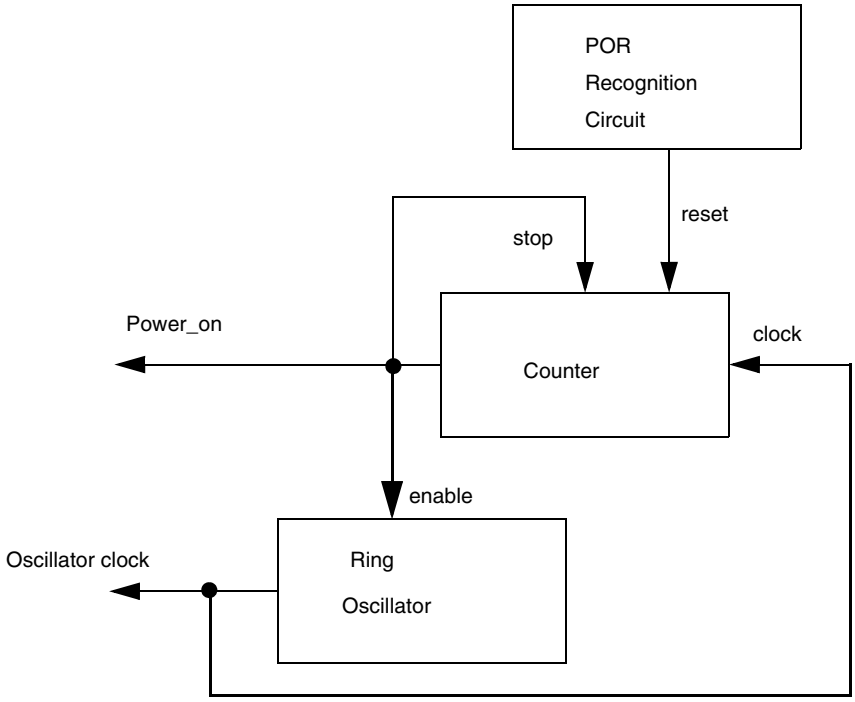


Figure 11-1. Power on Reset Block Diagram

Table 11-1. Power On Reset characteristics

Parameter	MIN	MAX	Units
Power_on assertion width*	75	850	ns
POR Recognition Circuit threshold	0.55	1.05	V

* The Power_on assertion width is measured from the time the supply pass the Recognition circuit threshold to the time the Power_on signal is negated.

11.2.1 POR Module Pin List

Table 11-2 lists all the pins in the POR module.

Table 11-2. POR Module Pin List

Pin Name	Direction	Description
por_res_b	Output	Power_On signal
por_oscl	Output	Oscillator clock

Chapter 12
Synthesizable PMBIF (SMPBIF)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	05/07/03		Updated for LTE specification release.

12.1 Overview

The New Custom Module features are:

- Up to 128 registers accessible from the Synthesisable Peripheral Module Bus(S-PMB).
- Interrupt interface.
- Scan interface.

12.2 New Custom Module Structure

The New Custom Module is composed of two main modules which connect between the PMB and IO pins:

- The S-PMB Interface.
- The Customer Specific Logic.

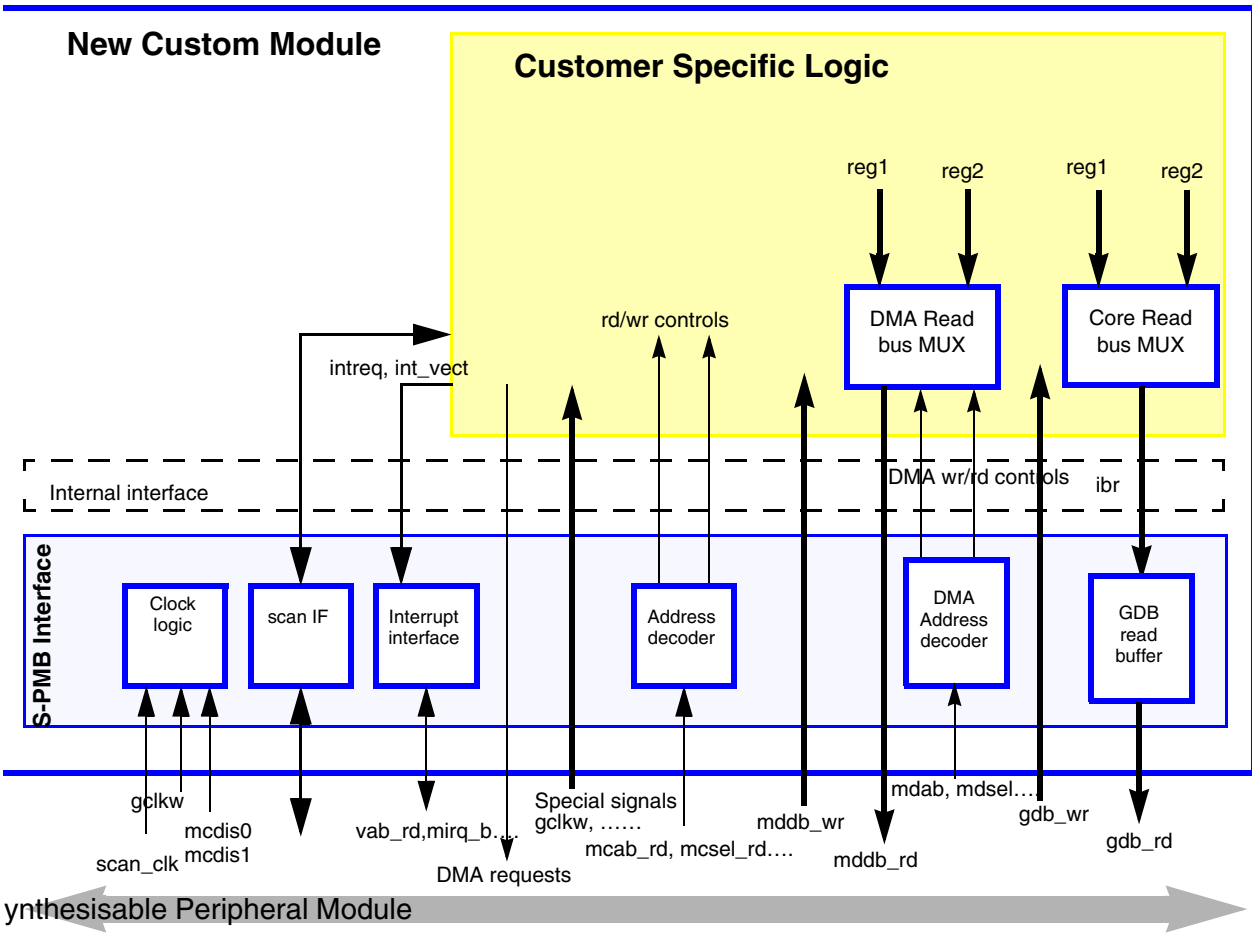


Figure 12-1. New Custom Module hierarchy

12.3 Synthesisable Peripheral Module Bus(S-PMB)

The Synthesisable Peripheral Module Bus(S-PMB) consists of a group of chip's global signals, used for control and data transfer between the Core/DMA and the New Custom Module.

This bus is connected to **Flip-Flop based** Core/DMA design. Timing of this bus is quite complicated and extensive knowledge is required in order to use these signals directly in new designs.

12.3.1 S-PMB Signals Description

Table 12-1 lists the S-PMB Interface signals and their functionality.

WARNING:

All signals directions relatively to the New Custom Module.

Table 12-1. SPMBIF Pin List

Pin Name	Direction	Description
Core Access Pins		
mcab_wr[6:0]	Input	Core Write Address Bus.
mcsel_wr	Input	X or Y I/O Write Select. Connected at the module instantiation level to one of the MCSELX_WR MCSELY_WR
gdb_wr[23:0]	Input	Core Global Write Data Bus. (GDB_WR[15:0] for 566xx family).
mcab_rd[6:0]	Input	Core Read Address Bus.
mcsel_rd	Input	X or Y I/O Read Select. Connected at the module instantiation level to one of the MCSELX_RD MCSELY_RD
gdb_rd[23:0]	Output	Core Global Read Data Bus. (GDB_RD[15:0] for 566xx family).
mcdis0	Input	Core pipeline stall indication (**used in SPMBIF only. hidden for the user**).
mcdis1	Input	Core pipeline stall indication (**used in SPMBIF only. hidden for the user**).
DMA Channel Access Pins[1]		
mdab[6:0]	Input	DMA Address Bus.
mdsel	Input	DMA-XIO Select. Connected at the module instantiation level to one of the four following PMB signals: MDSELX, MDSELY.
mdrd	Input	DMA Read.
mddb_wr[23:0]	Input	DMA Write data bus.
mddb_rd[23:0]	Output	DMA Read data bus.
dfmre	Input	DMA request enable (for fast DMA requests only).
mdrq_b	Output	DMA request.
Interrupt Request Pins		
mirq_b	Output	Peripheral Module Interrupt Request. Connected at the module instantiation level to one of the 12 interrupt request line mirqn[11:0]. Active low.

Table 12-1. SPMBIF Pin List

Pin Name	Direction	Description
mnmirq_b	Output	Non Maskable Interrupt request. (optional)
pivrd	Input	Vector Read Strobe.
pivack	Input	Peripheral Module Interrupt Acknowledge.
vab_rd[7:1]	Output	Interrupt Vector Bus.
pivsse	Input	Interrupt Vector Source Select. Connected at the module instantiation level to one of the 12 interrupt Vector Source Select line pivsse[11:0].
pinmivse	Input	Non Maskable Interrupt Vector Source Select. Used when one or more vectors correspond to a non maskable interrupt; else connected to gnd.
Scan		
scan_mode	Input	Scan Mode
se_source	Input	Scan Enable source signal (for gating with scan_mode).
scan_clk	Input	Test Clock for New Custom Module.
scan_res	Input	Peripheral Module's Test Mode reset.
Misc.		
gclkw	Input	Raw clock for New Custom Module.
ndbgctl	Input	'The core is in the DEBUG mode' indication.
bcstop	Input	This signal indicates that the core is in stop mode. It is asserted four cycles before the **gclkw** goes idle.
pires	Input	Peripheral Module's Software reset.
res	Input	Peripheral Module's Hardware reset.

1. 56300 Family only

12.4 S-PMB Interface

The S-PMB Interface (called S-PMBIF) module is a Verilog RTL netlist, created by the New Custom Module's designer from RTL subcomponents, supplied by Motorola according to the instructions and the structure of the New Custom Module.

This section describes the main building blocks of the S-PMBIF, refer to Figure 12-1., "New Custom Module hierarchy" for schematic view.

12.4.1 SPMBIF Internal Clock Logic

The gclkw clock is supplied to the S-PMB Interface and added with the test clock functionalities inside it. The following figure describes the implementation of the clock multiplexing within the S-PMB Interface.

The generated clocks are used by the S-PMB Interface only.

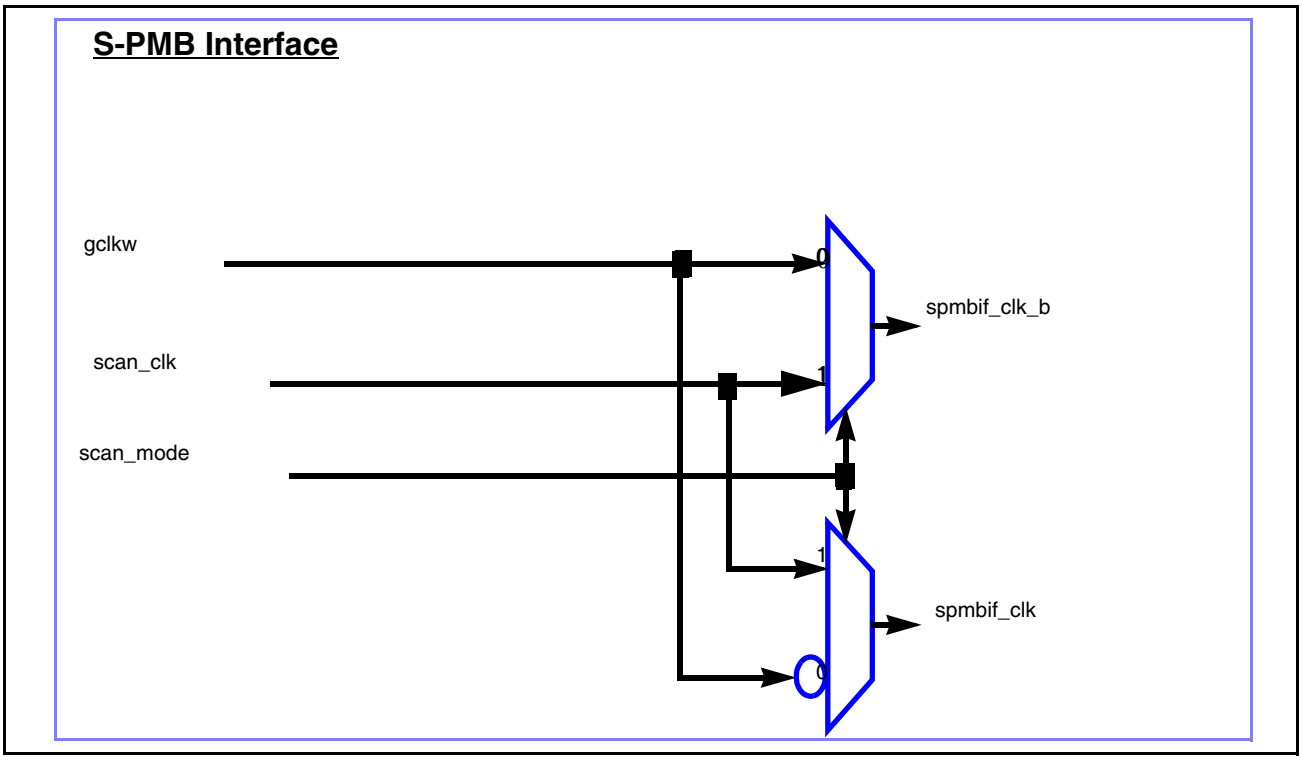


Figure 12-2. S-PMB Interface Clock Logic

WARNING:

The New Custom Module designer must implement the New Custom Module clock in this way and use only the posedge of this clock.

12.4.2 Address Decoder

This is the most important part of the S-PMB Interface. The address decoder generates strobe enable signals for each and every Customer Specific Logic register as described below.

12.4.2.1 Write Control Signals

For register accessed by the core only, the address decoder generates one signal for each register:

- An enable signal (**mcwen_<regname>**), which is a non-clock signal.

The following figure displays the usage of this signal in the Customer Specific Logic:

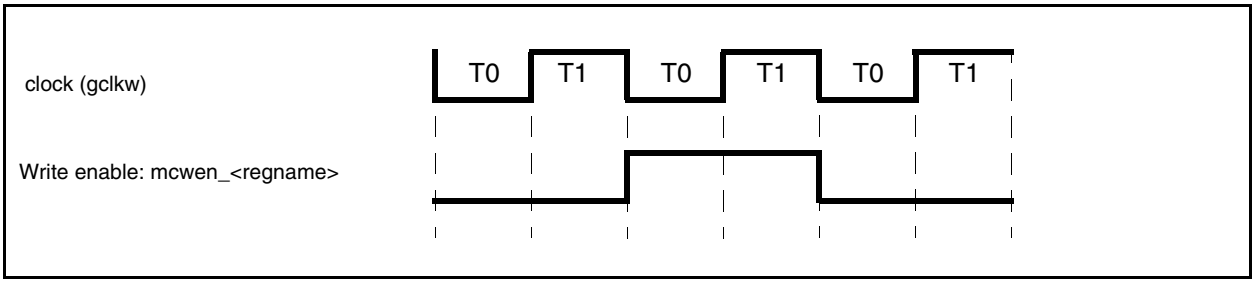


Figure 12-3. Write Enable Signal Timing Diagram

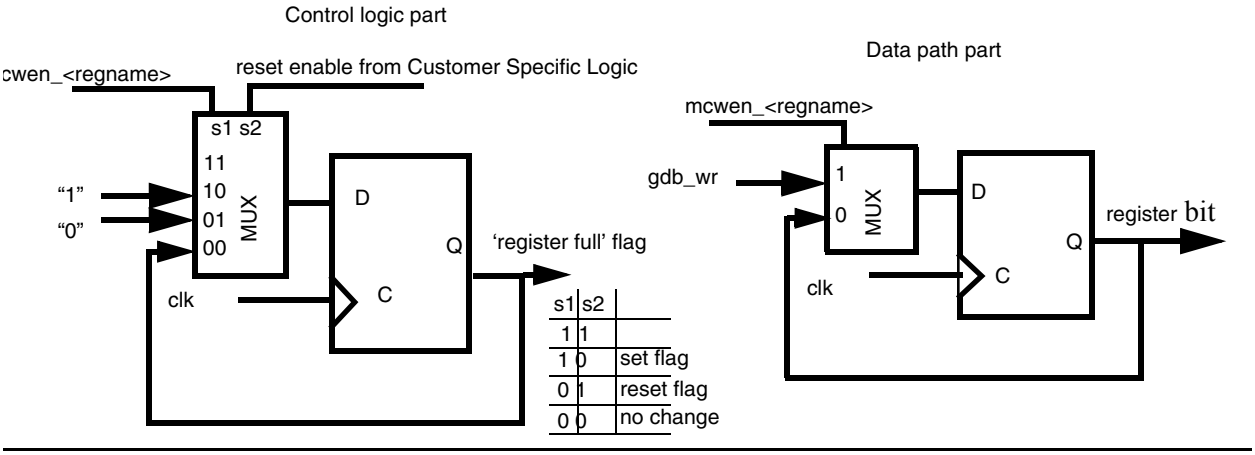


Figure 12-4. Write Enable Signals Connections

For more details on Core write access timing, refer to Figure 12-15., "Core Write Access Timing Diagram".

For registers accessed by core and DMA 2 signals are generated:

- Two Write Strobe Enable signals
  - — **mcwen_<regname>**
  - — **mdwen_<regname>**

which are active respectively for core access and for DMA access.

Core has priority over DMA access when the access is performed at the same cycle and care must be taken by the designer to ensure that priority.

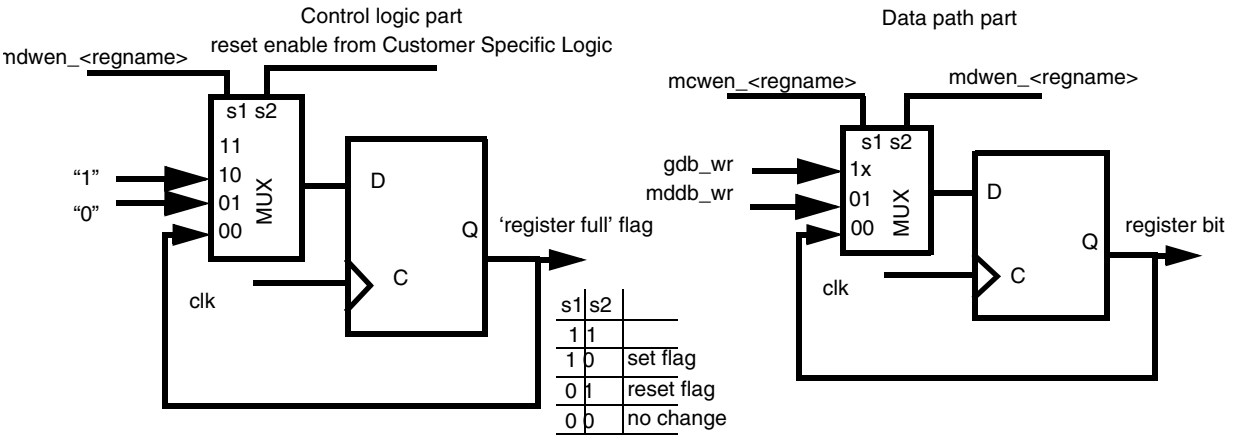


Figure 12-5. Write Enable Connections, DMA Accessible Register

For more details on DMA access timing, refer to Figure 12-17., "DMA Write Access Timing Diagram"

For Verilog code examples refer to Section 12.10.1.1, "Write Accessible Registers" and Section 12.10.1.3, "DMA Accessible Registers".

12.4.2.2 Read Control Signals

The address decoder generates one signal for core read access:

- An enable signal, which is a non-clock signal.

The following figure displays the usage of this signal:

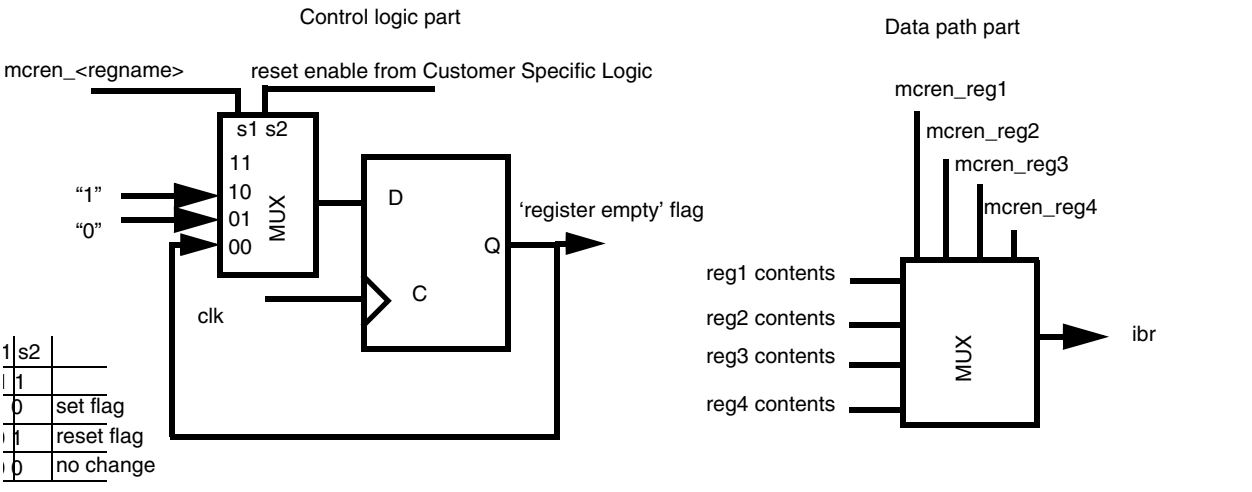


Figure 12-6. Read Strobe and Enable Signals Connections.

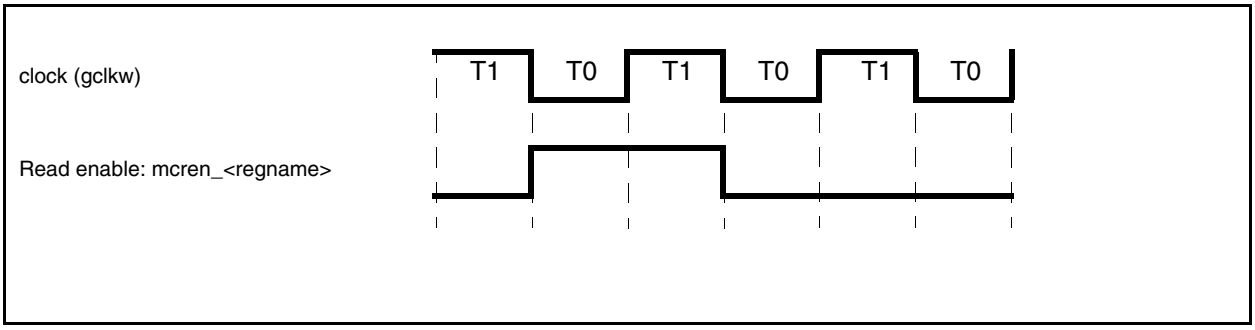


Figure 12-7. Read Enable Signal

For more details on Core read access timing, refer to Figure 12-16., "Core Read Access Timing Diagram".

For DMA accessed registers, the S-PMB Interface generates also an enable signal corresponding to the DMA access.

For more details on DMA access timing, refer to Figure 12-18., "DMA Read Access Timing Diagram"

For Verilog code examples refer to Section 12.10.1.2, "Read Accessible Registers".

The MUX between all readable registers is done in the costumer specific logic and the data is driven out on ibr (for core access) or mddb_rd (for DMA access).

12.4.3 Data Buffers

The S-PMB Interface implements data bus buffering logic, designed to fit the DSP56300 core pipeline.

12.4.3.1 Core Data Bus Read Buffer

In the SPMBIF there is a backup buffer that is used to store data under certain conditions and then restore it onto the **gdb_rd**. For DMA read access there is no buffer and the data is driven from the module to the **mddb_rd**.

For Core and DMA write access registers, there is no data bus buffer. These registers are written directly from the DMA data bus (**mddb_wr**).

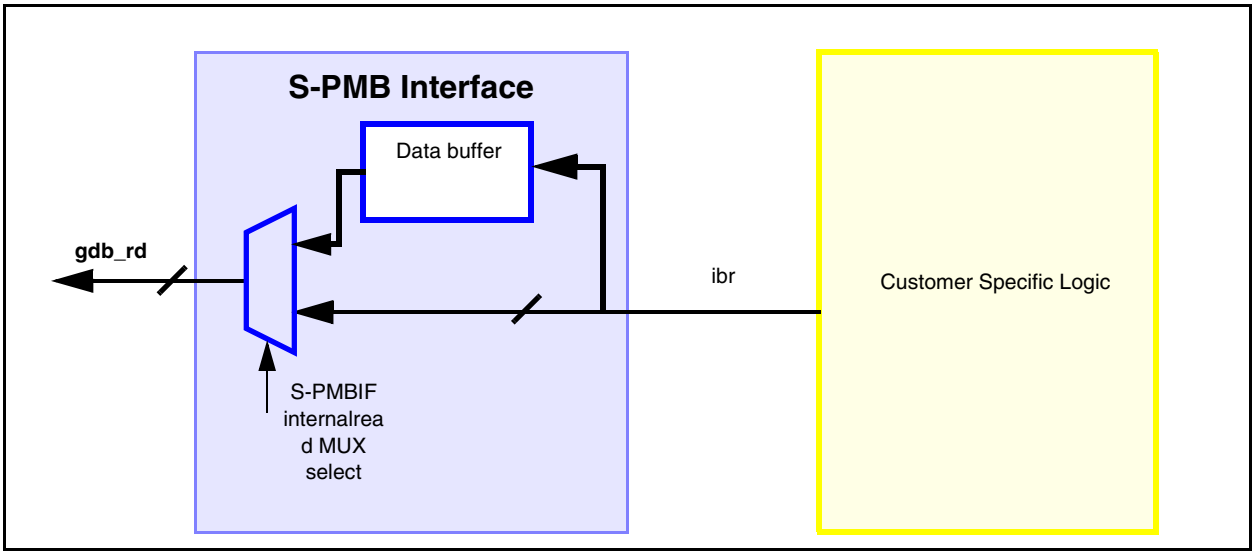


Figure 12-8. Core Data Read Buffer

12.4.4 The Interrupt Interface

This part ensures the correct timing between the Customer Specific Logic and the S-PMB. This interface deals with the following signals:

12.4.4.1 The Interrupt Request

There are two types of interrupt requests: A regular interrupt request (**mirq_**) and a non maskable interrupt request (**mnmirq_**). There is one interrupt request available for each peripheral module (12 for the 56300 family and 8 for the 56600 family), and only one non maskable interrupt request for a derivative, which means that only one peripheral module in a derivative may use this interrupt type. These two signals (**mirq_** and **mnmirq_**) are the sampling of corresponding signals (**intreq** and **nmireq**) from the Customer Specific Logic. The designer should generate these signals according to the examples given in Section 12.10.3, "Interrupt Logic".

The following Figure 12-9., "Interrupt Request" describes the implementation of the interrupt request mechanism in the S-PMB Interface. (The **mnmirq_** is implemented in a similar manner).

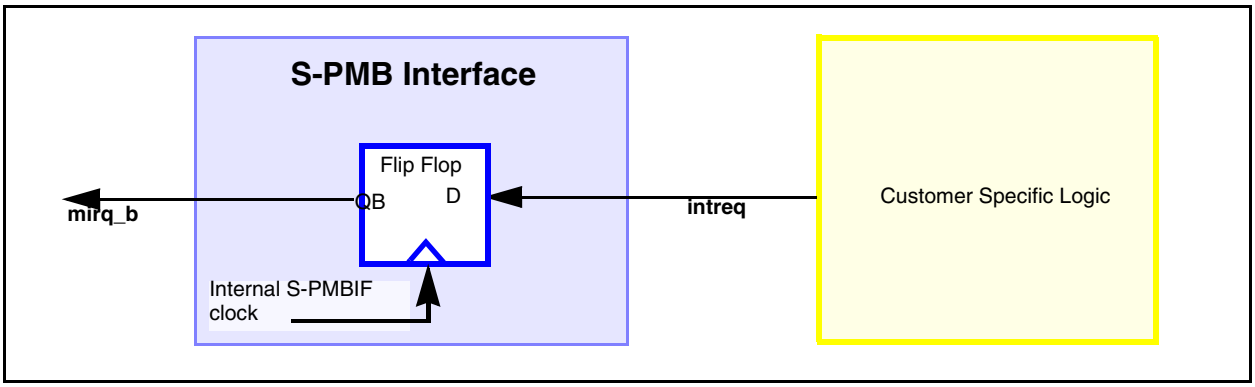


Figure 12-9. Interrupt Request

The **intreq, nmireq** signals must be the direct combinational result of the status and the interrupt enable signals in the Customer Specific Logic. The designer must not delay the **intreq, nmireq** by inserting more sequential logic in the Customer Specific Logic. Doing so might cause additional erroneous interrupts to be serviced by the core after an interrupt has been serviced.

WARNING:

For chip's production testing purposes it makes sense to ensure by the design of the Customer Specific Logic that when the New Custom Module is not enabled, its interrupt request is not asserted.
That way, the module will not interfere with the test program flow of the core when only the core is being tested

Refer to Figure 12-21., "Interrupt Timing Diagram" for timing description.

12.4.4.2 The Vector Address Bus

This bus (bits 7 to 1) consists of two parts. The lower part (the variable part), encodes the different vectors needed by the New Custom Module. The number of bits of this part depends on the number of interrupt vectors the New Custom Module needs:

Table 12-2. VAB Bits Partition vs. Number of Vectors

Number of vectors	High bits Constant part	Low bits Variable part
0…1	vab_rd[7:1]	--
2	vab_rd[7:2]	vab_rd[1]
3…4	vab_rd[7:3]	vab_rd[2:1]
5…8	vab_rd[7:4]	vab_rd[3:1]
9…16	vab_rd[7:5]	vab_rd[4:1]
17…32	vab_rd[7:6]	vab_rd[5:1]
33…64	vab_rd[7]	vab_rd[6:1]

These bits are supplied by the Customer Specific Logic with timing according to the timing constraints and following the examples given in Section 12.10, "Verilog Code Examples". The upper part corresponds to the vectors set base address. The vector base address is defined at the chip configuration stage, when the New Custom Module is instantiated and configured.

The following Figure 12-10., "Interrupt Vector Address Bus" describes the implementation of the Vector Address Bus mechanism in the S-PMB Interface.

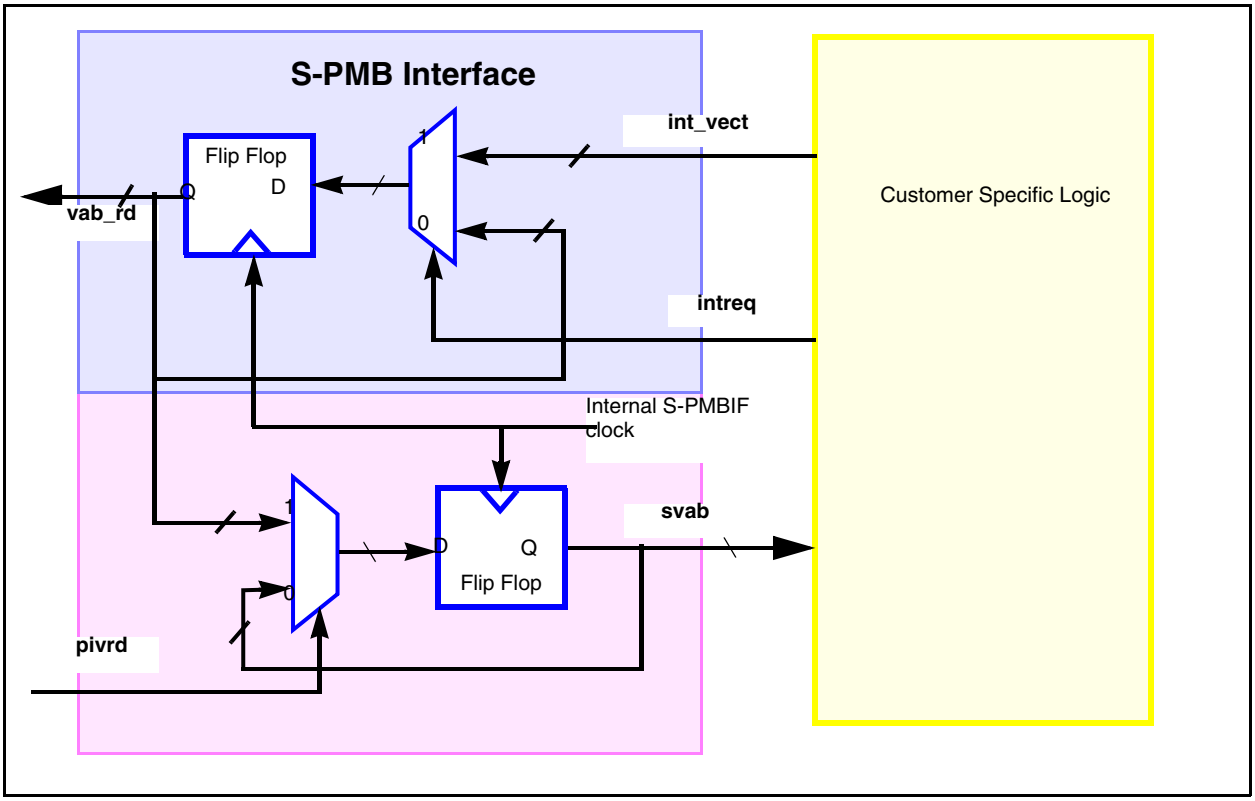


Figure 12-10. Interrupt Vector Address Bus

The **svab** represents the actual vector that has been sampled by the core. It is required in cases where the **int_vect** bus can change to a higher priority after the interrupt request has been acknowledged by the core. If the actual accesses to related registers are not used to de-assert the request this signal should be decoded in order to de-assert the correct interrupt request after one of the pending requests has been serviced.

For code examples refer to Section 12.10.3, "Interrupt Logic".

> The **int_vect** must be the direct combinational result of the status and the interrupt enable signals in the Customer Specific Logic. If additional sampling is inserted on the **int_vect**, switching to a higher priority vector after the interrupt request has been acknowledged by the core will not be possible.

Refer to Figure 12-21., "Interrupt Timing Diagram" for timing description.

12.4.5 The Scan Interface

This part of the S-PMB Interface supplies the Customer Specific Logic with the signals required for scan chain testing.

The Customer Specific Logic's scan chain testing requires the following dedicated signals to be present in the design:

- **Serial Scan In** (scan_in).
  This signal is usually shared with one of the New Custom Module's external IO pins.
- **Serial Scan Out** (scan_out).
  This signal is usually shared with one of the New Custom Module's external IO pins.

- **Scan Enable** (scan_en).
  This signal is usually shared with one of the New Custom Module's external IO pins. Also, a chip level common **scan_en** signal can be used.
- **Scan Mode** (scan_mode).
  This signal is supplied by a chip level signal. This can be a primary pin or a signal, arriving from a chip level test module. Refer to: Section 12.4.5.1, "The Scan Mode (scan_mode) Signal".
- **Scan Clock(s)**.
  This is a set of signals that is supplied by the S-PMB Interface. In some cases scan clock signals can be shared with the New Custom Module's external IO pins. Refer to: Section 12.8.2, "Clock Signals" for description of S-PMBIF clock methodology.
- **Scan Reset** (scan_res).
  This signal is connected to a primary pin and functions as the asynchronous reset of the design FFs.

12.4.5.1 The Scan Mode (scan_mode) Signal

This signal is supplied by a mandatory external pin of the derivative. It is held high by the external tester during the New Custom Module test Mode.

The tester is required to synchronize the assertion of the scan_mode signal to the low level of the core clock.

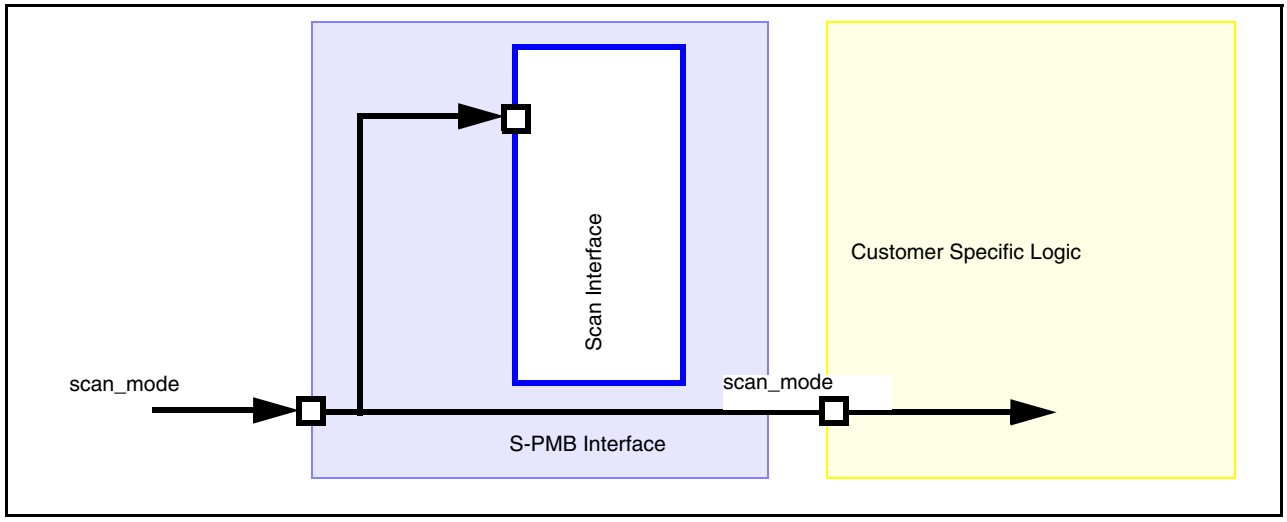


Figure 12-11. The Scan Mode Signal.

12.4.5.2 The Scan Enable Pin (scan_en)

This signal is usually supplied by one of the chip's external I/O pins. It can be one of the New Custom Module's external I/O pins. This pin can have regular functionality in the non-test mode or function as a dedicated Scan Enable I/O pin.

The Scan Enable signal from the I/O is connected to the S-PMB Interface and gated there with the Scan Mode signal (regular functionality not affected). The result of the gating is fed back into the Customer Specific Logic.

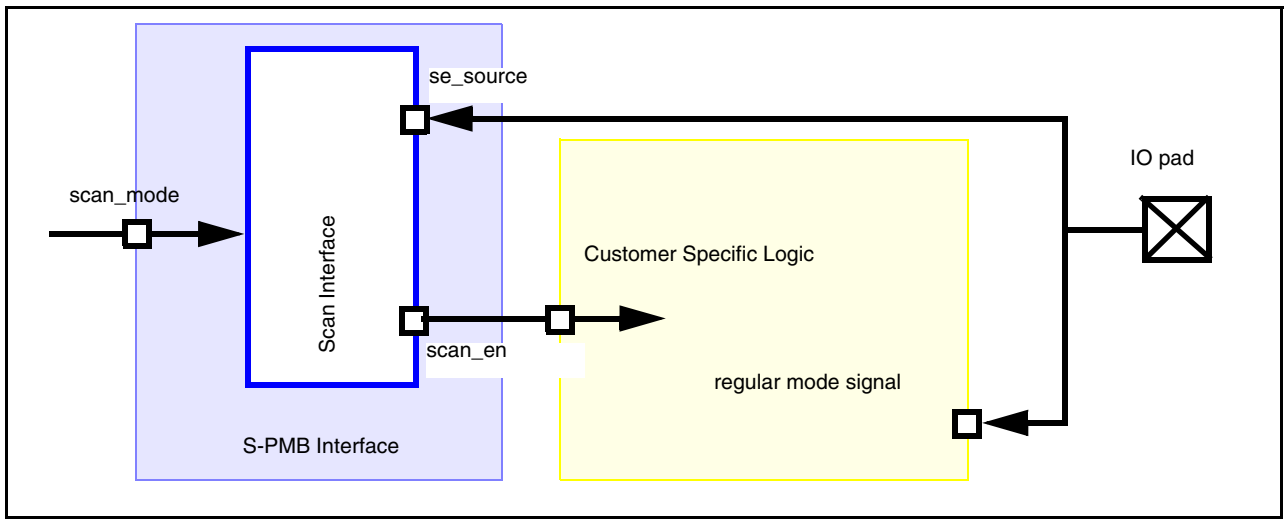


Figure 12-12. The Scan Enable Signal

12.4.5.3 The Scan Clock Source (scan_clk)

This signal is supplied to the New Custom Module from the chip level.

It connects to the S-PMB Interface as well as to the Customer Specific Logic.

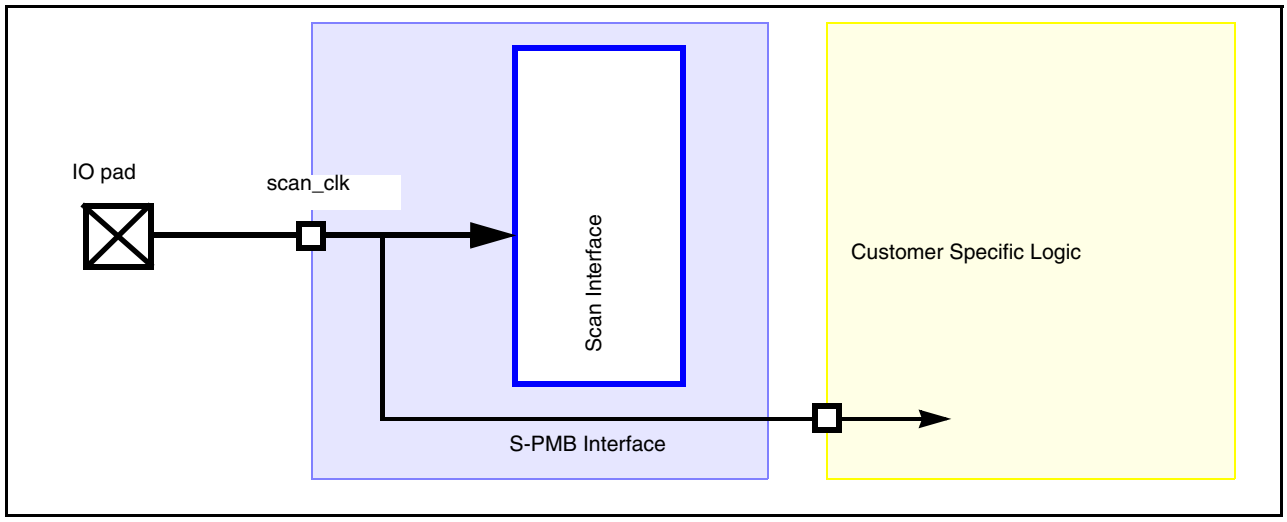


Figure 12-13. The Scan Clock Source (scan_clk)

12.4.6 The SCAN_RES Signal

This signal is used to reset all of the on-chip peripheral modules in the test mode. During the module's test mode (scan) it is activated from a primary pin.

Care should be taken not to overload this signal by the Customer Specific Logic.

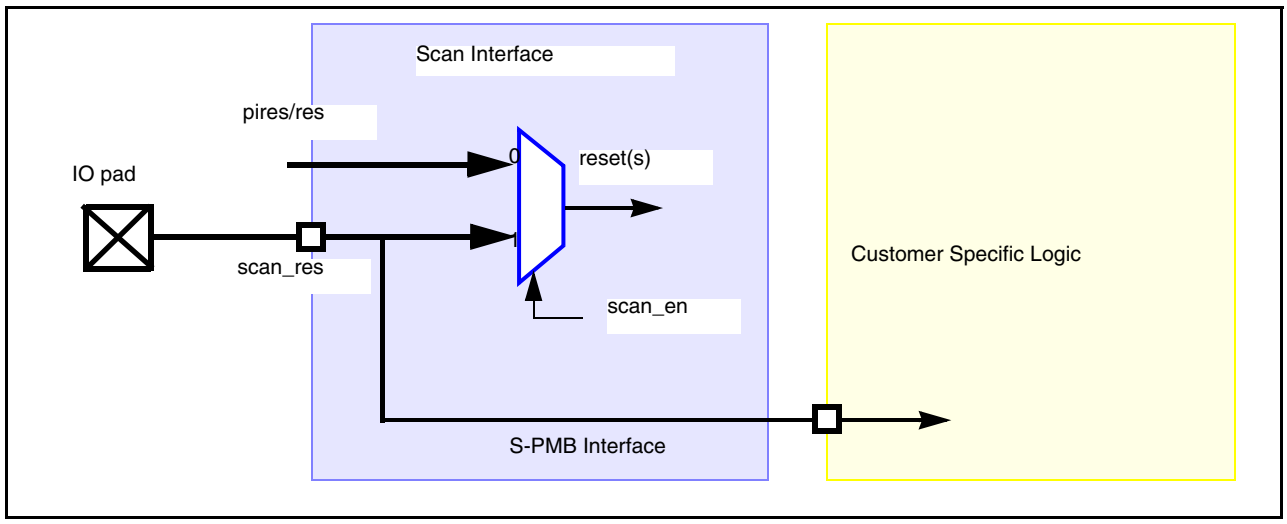


Figure 12-14. Asynchronous Reset Implementation

12.5 Directly Connected Signals

This section describes special signals which are connected directly to/from the Customer Specific Logic from/to the Synthesisable Peripheral Module Bus(S-PMB) or other on-chip ports without any interface.

12.5.1 The DMA Request Signal

Each peripheral requiring DMA servicing is assigned one (or more) of the DMA request lines, **mdrqn[21:0]**[1] through the signals ***drqn*** defined by the designer in the New Module Composer. The New Custom Module should assert its ***drqn*** line to trigger one of the DMA channels to transfer data according to the channel configuration. Each request line must be driven by only one peripheral. The requests lines must be negated between DMA service requests. Upon assertion, ***drqn*** should not be negated until after the DMA has serviced the request (Figure 12-22., "DMA Request Timing Diagram"). These request lines may operate as mentioned above, or operate as edge triggered requests if the corresponding DFMREN(21:18) line is asserted, i.e. they may be pulsed by the module (fast DMA request protocol). The corresponding DMA channel will service the request when able (**Figure 12-23., "Fast DMA Request Timing Diagram."**). The DMA request lines are active low.

WARNING:

For chip's production testing purposes it makes sense to ensure by the design of the Customer Specific logic that when the New Custom Module is not enabled, its DMA requests are not asserted.
That way, the module will not interfere with the program flow of the core when only the core is being tested

12.5.2 The Core Write Bus (gdb_wr)

This bus is used to transfer data by the core to on-chip peripheral modules.

The **gdb_wr** is a broadcast bus which is connected directly to the **Customer Specific Logic**.

1. The 56300 family only.

This is a global chip bus and care should be taken not to overload it by the Customer Specific Logic.

12.5.3 The DMA Write Bus (mddb_wr)

This bus is used to transfer data by the DMA controller to on-chip peripheral modules.

The **mddb_wr** is a broadcast bus which is connected directly to the **Customer Specific Logic**.

This is a global chip bus and care should be taken not to overload it by the Customer Specific Logic.

12.5.4 The BCSTOP Signal

This active high signal indicates that the Core has entered the Low Power Mode, initiated by the execution of the **STOP** instruction.

This is a global chip signal and care should be taken not to overload it by the Customer Specific Logic.

Refer to Section 12.8.2.2, "Low Power Considerations" for detailed description of the signal.

12.5.5 The PIRES Signal

This signal is used to reset all of the on-chip peripheral modules. It is active during the chip's execution of the **RESET** instruction by the Core.

Care should be taken not to overload this signal by the Customer Specific Logic.

12.5.6 The RES Signal

This signal is used to reset all of the on-chip peripheral modules. It is active during the chip's hardware reset.

Care should be taken not to overload this signal by the Customer Specific Logic.

12.5.7 Shared Memory Signals

This set of signals is connected directly from the New Custom Module to some of the on-chip memory modules.

The New Custom Module becomes one of the masters of these memory modules instead of the DMA controller.

Refer to: Table 12-1 "SPMBIF Pin List" for signals' list.

It is the designer's responsibility to implement the exact protocol for memory access. Refer to:Figure 12-19., "Shared Memory Write Access" and Figure 12-20., "Shared Memory Read Access" for the protocol's description.

12.6 S-PMB Interface Timing Diagrams

This paragraph describes some functional timing of core and DMA accesses to registers, interrupts and DMA requests.

12.6.1 Core Write Access

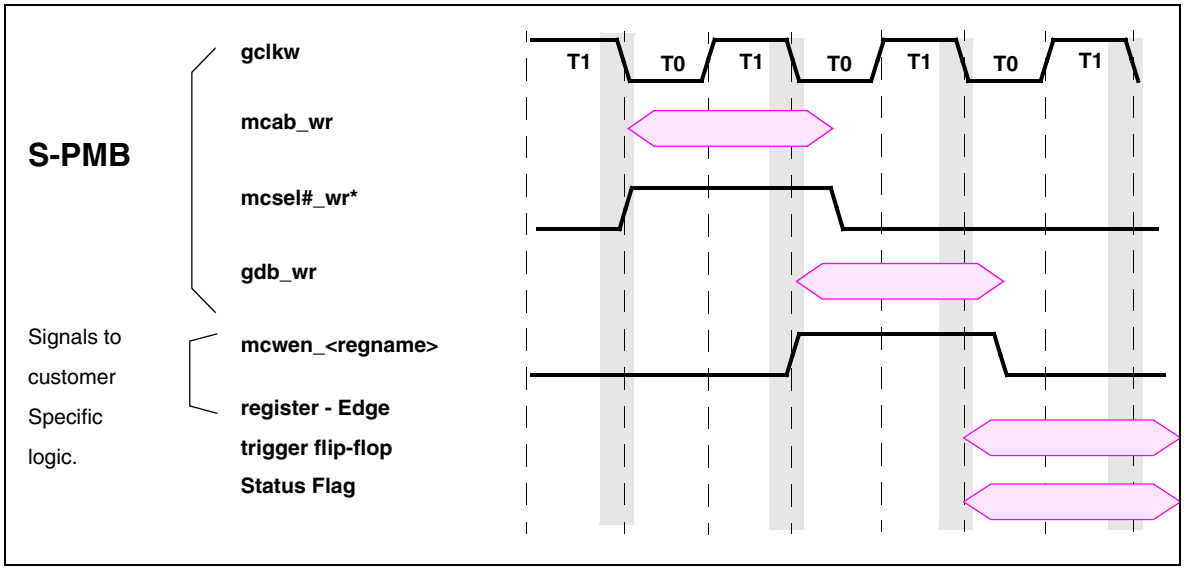


Figure 12-15. Core Write Access Timing Diagram

* Can be one of the following core access I/O space selects: **mcselx_wr,mcsely_wr.**

WARNING:

If the Write Strobe Enable signal needs to be sampled on the rising edge of the T0 clock, care should be taken to check possible hold time violations of that signal.
A certain delay of the Write Strobe Enable signal relatively to the T0 clock is ensured by the PMBIF design and is process and load dependent.

12.6.2 Core Read Access

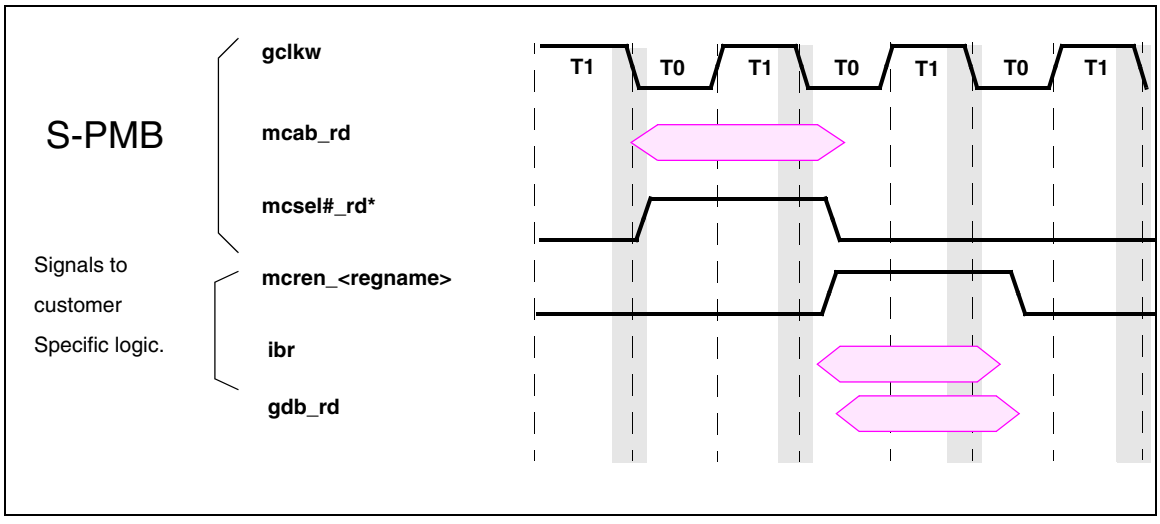


Figure 12-16. Core Read Access Timing Diagram

* Can be one of the following core access I/O space selects: **mcselx_rd,mcsely_rd.**

The two preceding timing diagrams do not reflect possible pipeline stops. These pipeline stops are caused by core external access wait states and are revealed by the assertion of the signals **mcdis0** and **mcdis1**. The pipeline stops are dealt within the S-PMB Interface and are transparent to the module designer.

12.6.3 DMA Write Access

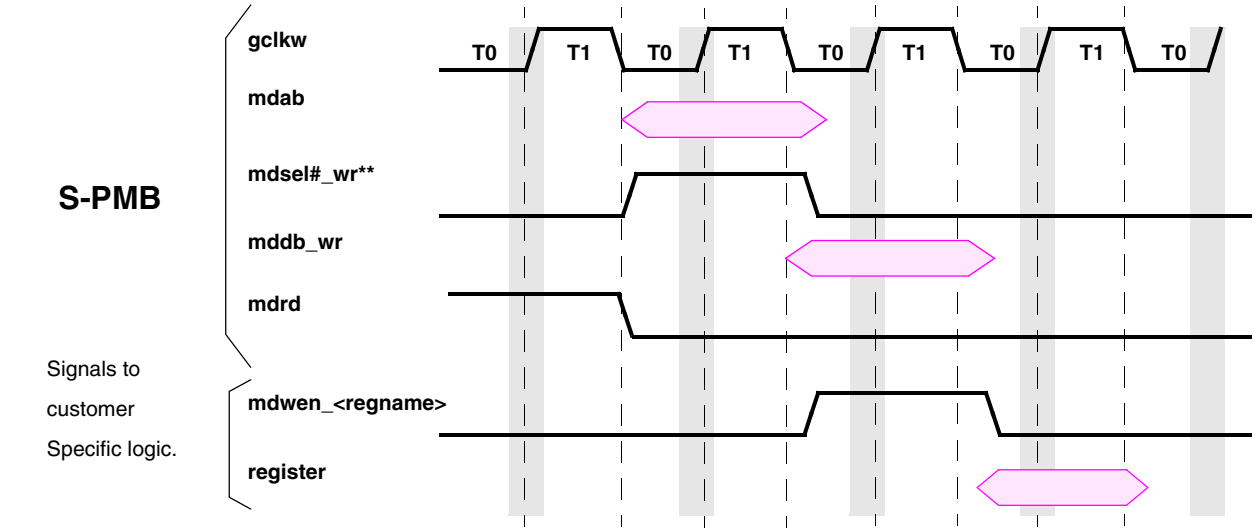


Figure 12-17. DMA Write Access Timing Diagram

** Can be one of the following DMA access I/O space selects:
mdselx_wr,mdsely_wr,mdselxe_wr,mdselye_wr.

12.6.4 DMA Read Access

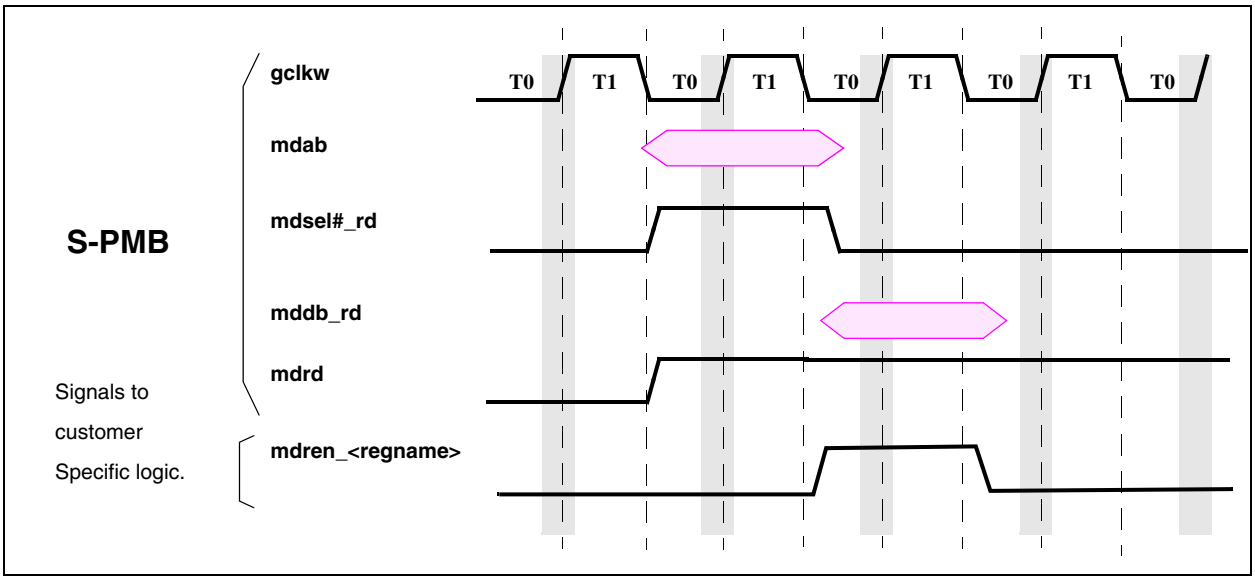


Figure 12-18. DMA Read Access Timing Diagram

** Can be one of the following DMA access I/O space selects:
mdselx_rd,mdsely_rd,mdselxe_rd,mdselye_rd.

12.6.5 Shared Memory Write Access.

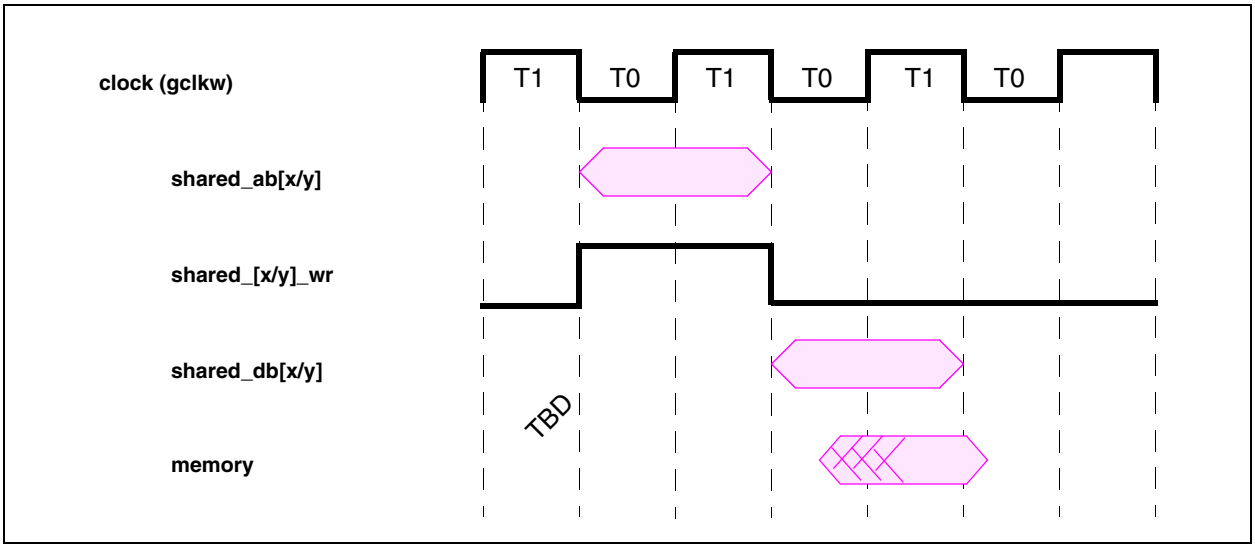


Figure 12-19. Shared Memory Write Access

12.6.6 Shared Memory Read Access.

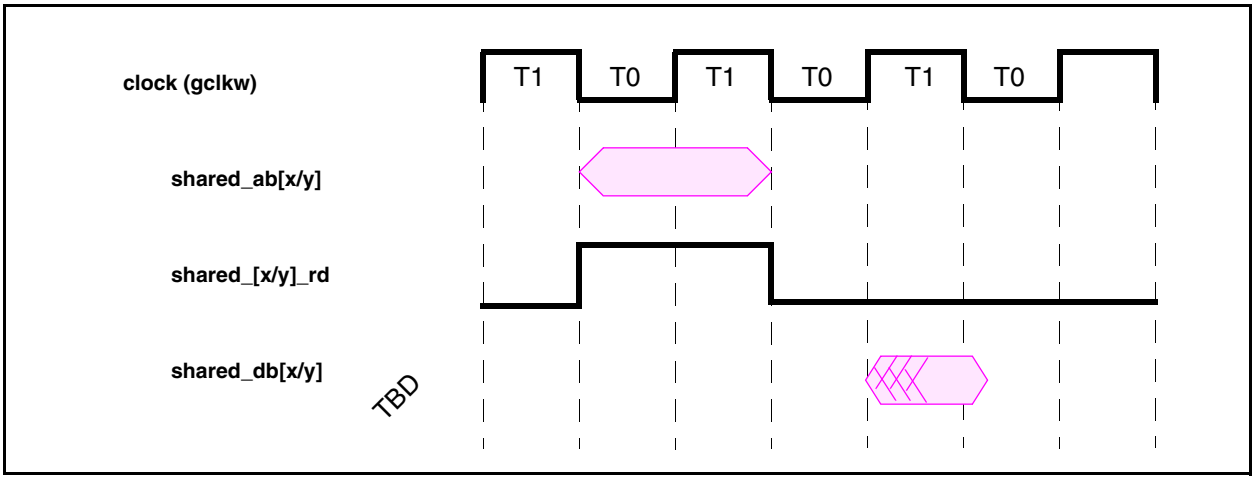


Figure 12-20. Shared Memory Read Access

12.6.7 Interrupts

The following diagram describes the interrupt request handshaking between a module and the core via the PMB.

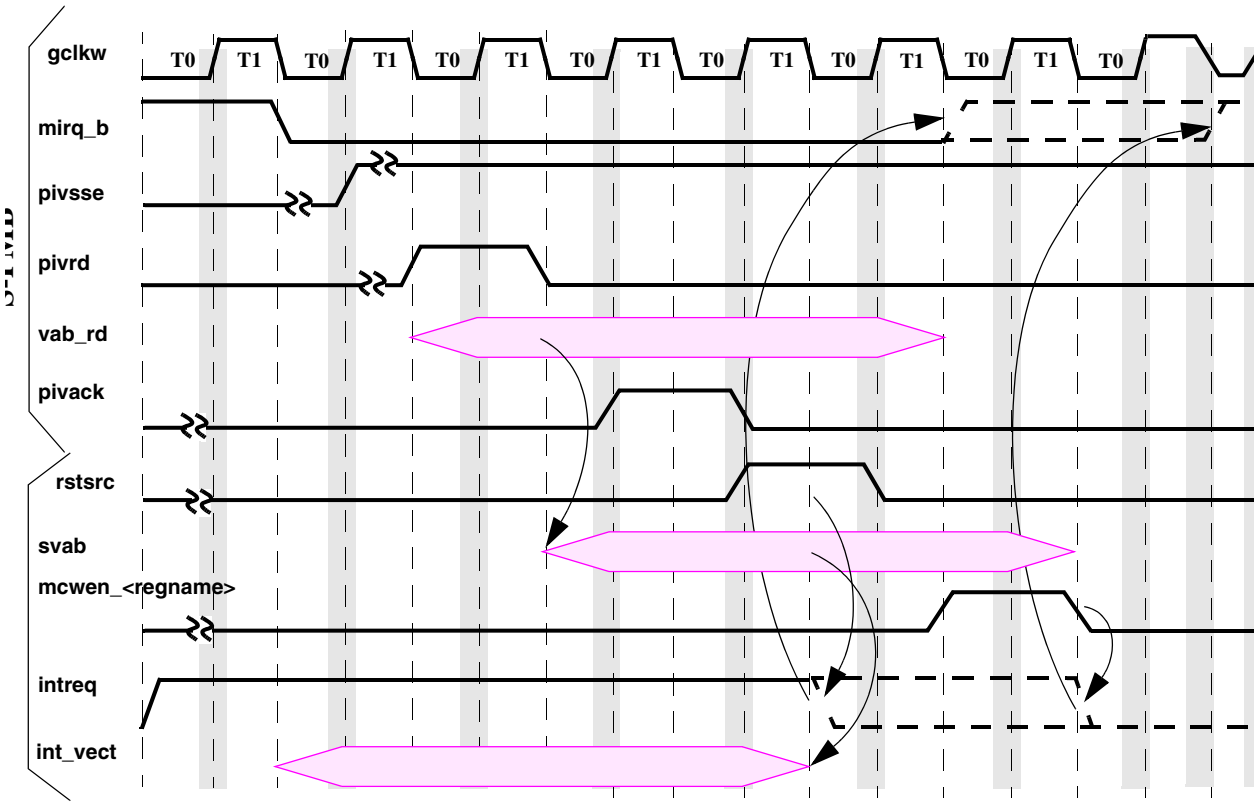


Figure 12-21. Interrupt Timing Diagram

The interrupt request of Customer Specific Logic is first asserted and the interrupt vector is driven. The S-PMB Interface samples the interrupt request onto the **mirq_** line. Then, the program interrupt controller in the core selects a peripheral module according to the priority encoding (signal **pivsse**), and after that asserts the **pivrd** signal (which is common to all the peripheral modules) to read the vector address from **vab_rd**. Then the vector is accepted in the instruction flow, and this is revealed by the interrupt vector acknowledge (**pivack**) assertion. This signal is used within the S-PMB Interface to generate the **rstsrc** signal. The peripheral module may negate the interrupt assertion upon receiving this signal. This is done, typically, when the interrupt servicing does not read nor write any register inside the module. Else, the interrupt servicing is the reading or the writing of a module register, in which case the interrupt request negation is triggered by the read or write enable signals (**mcwen_<regname>/mcren_<regname>**). This is the recommended way to implement the interrupt request negation.

12.6.8 The DMA Request

The following diagram describes the DMA request handshaking between a module and the core via the PMB.

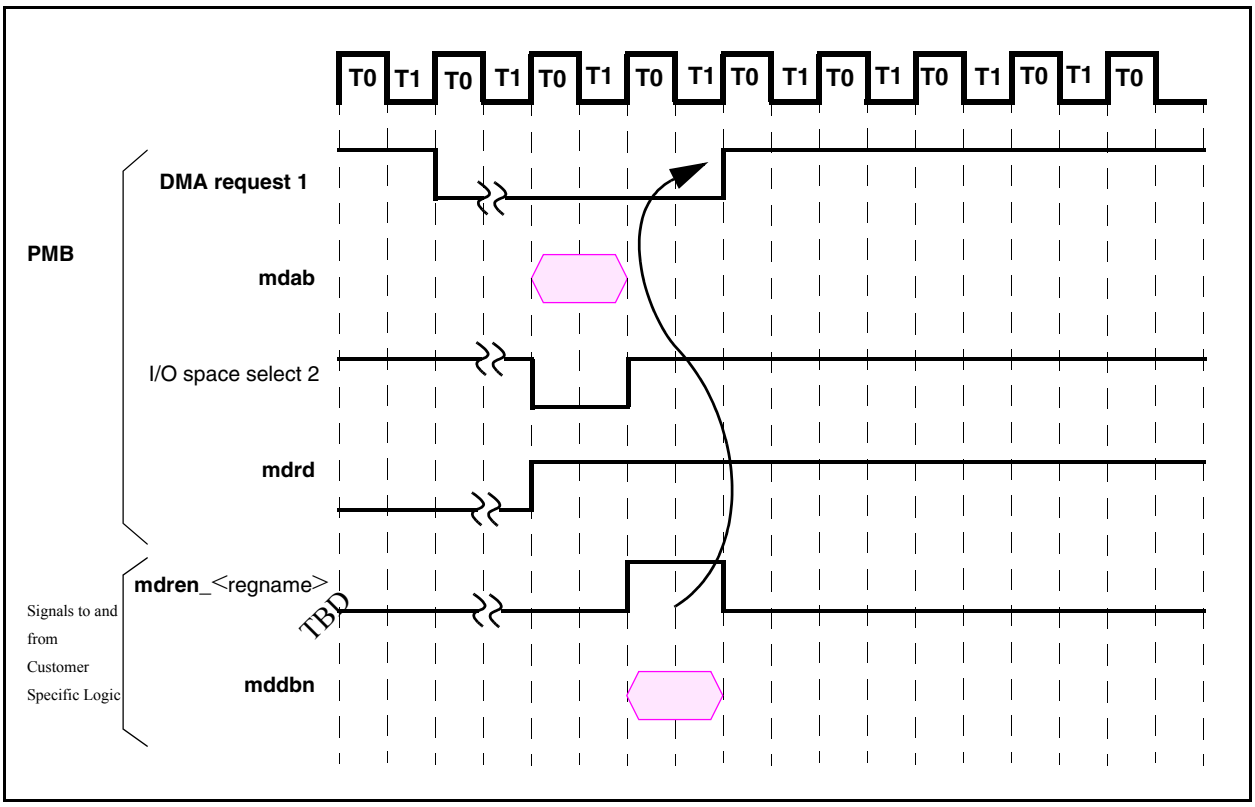


Figure 12-22. DMA Request Timing Diagram

5. The DMA request of the Customer Specific logic is connected directly to the PMB without any PMB interface glue logic.
6. Can be one of the following DMA access I/O space selects: **mdselx_,mdsely_,mdselxe_,mdselye_.**

Refer to Section 12.5.1, "The DMA Request Signal" for additional description of the DMA request signal.

The DMA request of the Customer Specific Logic is first asserted. The DMA controller then drives the address being accessed onto the DMA Address Bus, **mdab**, and asserts the I/O space select signal, according to DMA priority encoding. The DMA read signal, **mdrd**, is changed to indicate the type of DMA access. The appropriate enable signals and strobes are then generated and the request is serviced. The DMA request source can be negated using the read/write enables assertion during the DMA access (assuming that the peripheral module triggers the DMA to initiate a transfer to/from itself).

In addition to the above DMA request handshaking four additional Fast DMA requests are allowed in a derivative based on the DSP56300 core family. The following diagram describes a fast DMA request protocol using an additional DMA request enable signal from the core.

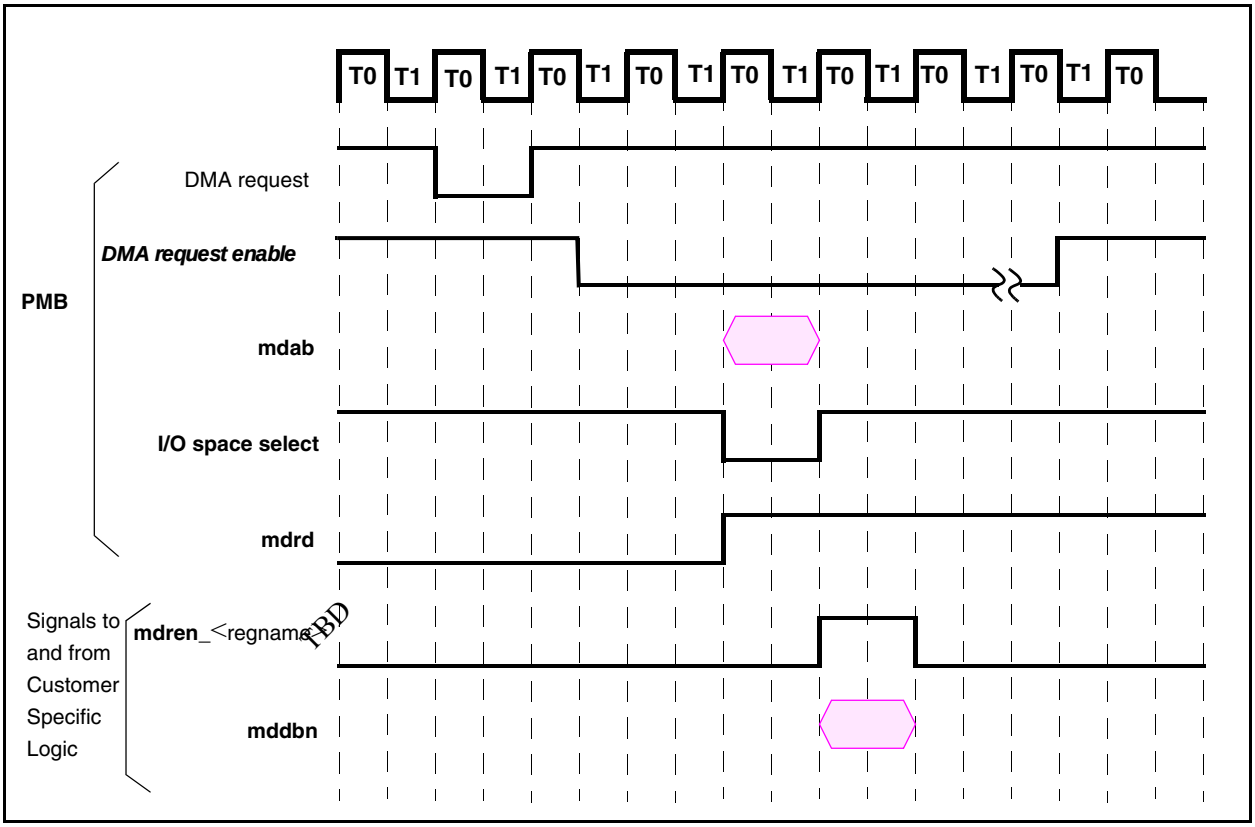


Figure 12-23. Fast DMA Request Timing Diagram.

WARNING:

The DMA accesses, described in the above diagram are read accesses. A write access is quite identical from the DMA request handshaking point of view. In that case, the **mdrd** signal will not be driven low by the DMA controller and the enable signal, generated will be write enable (**mdwen_r**).

12.6.9 The NDBGCTL Signal

The following diagram describes the functionality of the NDBGCTL signal.

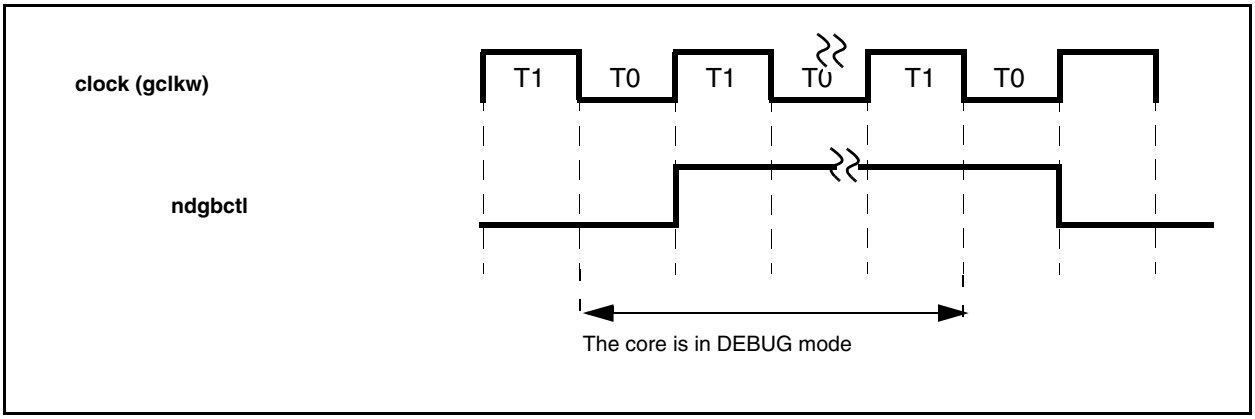


Figure 12-24. The NDGCTL timing diagram

12.7 The Customer Specific Logic Module

This is the module designed by the customer. The module's designer connects the Customer Specific Logic to the S-PMB Interface signals as well as directly to some of the Synthesisable Peripheral Module Bus(S-PMB) signals. These signals are described in the following table.

WARNING:

The direction of the signals is relatively to the Customer Specific Logic.

Table 12-3. Customer Specific Logic Signals

Type	Signals	Dir	Description
Core Access Signals	gdb_wr[23:0]	in	Global Data Write Bus. Chip level broadcast bus.
	ibr[23:0]	out	Internal module Bus for Read
	mcwen_<regname>	in	Register Core Write Enable (one for each core writable register)
	mcren_<regname>	in	Register Core Read Enable (one for each core readable register)
DMA Channel Access Signals	mdrq_b	out	DMA request
	dfmre	in	DMA request enable (for fast DMA requests only)
	mddb_wr[23:0]	in	DMA Write Data Bus.
	mddb_rd[23:0]	out	DMA Read Data Bus.
	mdwen_<regname>	in	Register DMA Write Enable (one for each DMA writable register)
	mdren_<regname>	in	Register DMA Read Enable (one for each DMA readable register)
Interrupt Request Signals	svab	in	The actual interrupt vector that has been sampled by the core. Used with the **rstsrc** signal to de-assert the appropriate interrupt request source.
	int_vect	out	Interrupt vector. The number of lower bits is deduced from the number of interrupt vectors.
	intreq	out	Interrupt request.
	nmireq	out	Non-maskable interrupt request.
	rstsrc	in	Reset interrupt request source.
Scan	scan_mode	in	Scan Mode signal
	scan_en	in	Scan Enable source signal
	scan_clk	in	Scan Clock signal
	scan_res	in	Test Mode reset
Misc.	gclkw	in	Raw clock for New Custom Module.
	ndbgctl	in	‘The core is in DEBUG mode’ indication.
	bcstop	in	Core executing the STOP instruction

12.8 Customer Specific Logic Hardware Design

The following sections define a few design rules that have to be complied with while designing a New Custom Module using the S-PMBIF system.

12.8.1 General Restrictions

- The Verilog design of the Customer Specific Logic must not use capital letters.

12.8.2 Clock Signals

The global clock is supplied to the Customer Specific Logic directly from the Synthesisable Peripheral Module Bus(S-PMB). It is the designer's responsibility to multiplex this clock with the test clock and add the required clock gating functionality to it.

Refer to Section 12.4.1, "SPMBIF Internal Clock Logic" and Section 12.10.4, "Clocks" for details.

There are two types of clock signals:

12.8.2.1 Rising and Falling Edge

The designer should make an effort to use only clock's rising edge in his/her design. However, sometimes some operations have to be proceeded in half a cycle and then the positive edge of the inverted clock should be used.

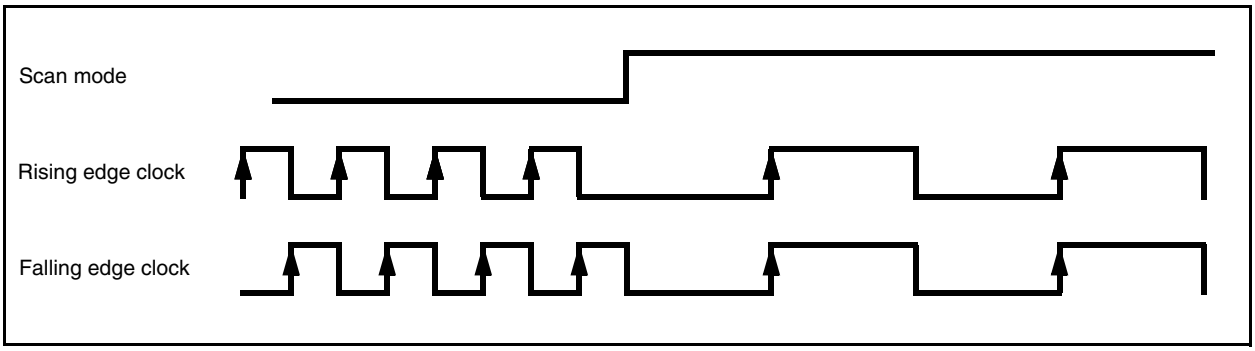


Figure 12-25. Clock in Scan Mode

This way, during scan mode, all the FFs are activated at the same actual edge, and there is no need for different clock domains.

12.8.2.2 Low Power Considerations

The family features instructions to reduce the total chip's power consumption. One of them is the '**STOP**' instruction. During the **STOP** processing state the majority of the core's and all peripheral clock signals are disabled. Among these clock signals is the clock source signal for every synchronous clock within the New Custom Module (GCLKW). The signal that indicates the **STOP** processing state is BCSTOP. After the assertion of the BCSTOP signal, the GCLKW is active for 2 to 4 cycles and then goes IDLE at logic low state. If the New Custom Module has a special state during the **STOP** mode, the designer must make an effort for the design to reach the desired state within these clock cycles of the New Custom Module's synchronous clock. The following diagram describes the behavior of New Custom Module clocks during the **STOP** mode.

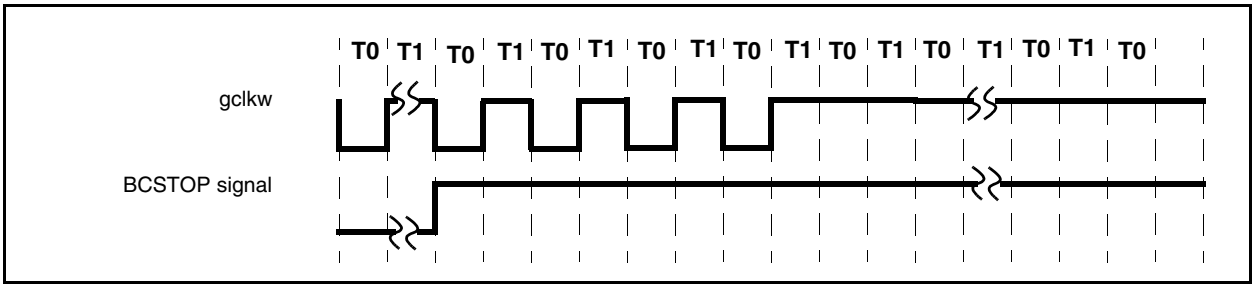


Figure 12-26. Clock Behavior in STOP Mode

12.9 Test Mode Considerations

This chapter describes the specific DFT considerations of the S-PMB Interface

12.9.1 Asynchronous Resets

It is important to have the New Custom Module's asynchronous resets controllable in test mode from a primary pin.

That is why an additional test mode reset was created.

12.10 Verilog Code Examples

This chapter proposes some implementation methods to the Customer Specific Logic designer of parts of the Customer Specific Logic.

12.10.1 Read and Write Registers through PMB

These simple examples illustrate the implementations of some of the Customer Specific Logic design rules in Verilog code.

12.10.1.1 Write Accessible Registers

```
// tx register write from GDB:
reg[23:0]           tx_reg;
always @(posedge clk)
                    if (reset) tx_reg <= 24'b0;

if (mcwen_tx_reg) tx_reg <= gdb_wr;

// ctl register written from GDB with asynchronous reset:

wire reset; // Software + Hardware reset
assign reset = scan_mode ? scan_res: pires | res;


reg[23:0]ctl_reg;
always @(posedge clk or posedge reset)
          if (reset) ctl_reg <= 24'b0;                    // sw and hw reset
          else if (mcwen_ctl_reg) ctl_reg <= gdb_wr;      // register writing
```

Figure 12-27. Write Accessible Registers

12.10.1.2 Read Accessible Registers

Figure 12-28 implements register read using the read enable signals.

```
// read data path FF's implementation
wire reset; // Software + Hardware reset

assign reset = scan_mode ? scan_res : pires | res;

reg [23:0] ibr_out;
reg [23:0] ibr_int_buffer;
wire enable_or;

always @(mcren_stat_reg or mcren_ctl_reg)
begin
case(1'b1)
            mcren_stat_reg : ibr_out = stat_reg;
            mcren_ctl_reg : ibr_out = ctl_reg;
            default: ibr_out = 'b0;
endcase

assign enable_or = |{mcren_stat_reg,mcren_ctl_reg};

always @(posedge clk or posedge reset)
if (reset)
   ibr_int_buffer <= 'b0;
else if (enable_or)
   ibr_int_buffer <= ibr_out;

assign ibr = enable_or ? ibr_out : ibr_int_buffer;
```

Figure 12-28. Core Readable Registers - Flip Flop Implementation.

```
// read data path latch implementation
wire reset; // Software + Hardware reset

assign reset = scan_mode ? scan_res : pires | res;


always @(reset or mcren_stat_reg or mcren_ctl_reg)
if (reset)
 ibr <= 'b0;
case(1'b1)
            mcren_stat_reg : ibr = stat_reg;
            mcren_ctl_reg : ibr = ctl_reg;
endcase
```

Figure 12-29. Core Readable Registers - Latch Implementation.

12.10.1.3 DMA Accessible Registers

These registers need an additional MUX to select the write source of the data written. Core accesses have priority over the DMA accesses.

```
// tx register written from GDB with asynchronous reset:
wire reset; // Software + Hardware reset
assign reset = scan_mode ? scan_res: pires | res;
reg[23:0]     tx_reg;
always @(posedge clk or posedge reset)
            if (reset) tx_reg <= 24'b0;                          // sw and hw reset
            else if (mcwen_tx_reg) tx_reg <= gdb_wr;             // Usually the core has
            else if (mdwen_tx_reg) tx_reg <= mddb_wr;// priority
```

Figure 12-30. DMA Write Accessible Registers

For DMA read accessible register, no Verilog code has to be written since this read access is fully implemented in the S-PMB Interface. The Customer Specific Logic has only to supply the register contents to the bus provided in the verilog template file using a simple Verilog 'assign' directive.

```
always @(mdren_stat_reg or mdren_ctl_reg)
case(1'b1)
                mdren_stat_reg: mddb_rd = stat_reg;
                mdren_ctl_reg: mddb_rd = ctl_reg;
                default: mddb_rd = 24'b0; // Very important to drive '0'.
endcase
```

Figure 12-31. DMA Read Accessible Registers

12.10.2 Flag Logic Management

This paragraph provides some examples of flag's implementation in the control logic part.

12.10.2.1 Switching Flags Using Access Enable Signals

```
// clk is the synchronous clock
reg             tx_empty,
                rx_full;                                    // Transmit flag:
always @(posedge clk or posedge reset)
                if (reset) tx_empty <= 1'b1;                // set at sw or hw reset
                else if (mcwen_tx | mdwen_tx) tx_empty <= 1'b0;
                else if (read_reg) tx_empty <= 1'b1;

always @(posedge clk or posedge reset)
                if (reset) rx_full <= 1'b0; // cleared at sw or hw reset
                else if (mcren_rx | mdren_rx) rx_full <= 1'b0;
                else if (write_reg) rx_full <= 1'b1;
```

Figure 12-32. Receive and Transmit Register Status Flags

12.10.2.2 Sticky Status Bit Management.

Sometimes, the status bit is a sticky bit than needs to be cleared by software explicit write of a certain value to the bit. The following example demonstrates such an implementation.

```
'define EVENTTEST<internal event that sets the bit>
'define BIT <bit number of the sticky status>

reg             sticky;
wire            eventreset;

assign eventreset = mcwen_status & ibw['BIT];

always @(posedge clk or posedge reset)
                if (reset) sticky <= 1'b0;// clear at sw or hw reset
                else if ('EVENTEST) sticky <= 1'b1; // priority MUX
                else if (eventreset) sticky <= 1'b0;
```

Figure 12-33. Sticky Status Bit Implementation

12.10.3 Interrupt Logic

The interrupt logic is composed of three parts:

- The interrupt request
- The vector address
- The status resetting by the request servicing. This is described in Figure 12-32., "Receive and Transmit Register Status Flags" and Figure 12-33., "Sticky Status Bit Implementation".

WARNING:

The proposed implementation of makes sure that no additional delay is introduced during the de-assertion of the interrupt request.
The interrupt request, itself, **must** be a combinational function of the status signals and the specific interrupt request enable signals in the control register.

In the following example, **cr** is the control register which includes the control signals: Transmit Interrupt Enable (**tie**) and Receive Interrupt Enable (**rie**). The status bits are the receive and transmit flags **tx_empty** and **rx_full**:

```
‘define          TIE              cr[1]
‘define          RIE              cr[0]
output[1:1]      int_vect;

// Interrupt request:
intreq = ‘TIE && tx_empty || ‘RIE && rx_full;

// Interrupt vector. Here, the receive interrupt has priority
// over the transmit interrupt:
assign int_vect[1] = ‘RIE && rx_full;
```

Figure 12-34. Interrupt Logic

When the interrupt request is not serviced by a core access, like in the case of timers where the interrupt triggers a general operation but not a core access to the timer, the interrupt request is reset by the interrupt acknowledge implemented in the signal **rstsrc**. In the following example, **endcnt0** is status bit which indicates that the counter 0 has reached 0.

```
‘define          CNT0IE           cr[0]              // Count 0 interrupt enable
‘define          CNTEND_VECT      <vector> // The assigned vector to the request
assign
// Interrupt request:
                 intreq = ‘CNT0IE && endcnt0 || ….
always @(posedge clk or posedge reset)
                 if (reset) endcnt0 <= 1’b0;
                 else if (rstsrc && svab[MSB:0] == ‘CNTEND_VECT) endcnt0 <= 1’b0;
                 else if (cnt0 == 0) endcnt0 <= 1’b1;
```

Figure 12-35. Using Interrupt Acknowledge to De-assert the Request

Refer to Figure 12-21., "Interrupt Timing Diagram" for timing diagrams of the Interrupt protocol.

12.10.4 Clocks

This section demonstrates the clock methodology of the S-PMB Interface.

12.10.4.1 The Clock Logic

All the Customer Specific Logic clocks should originate from generators within the Customer Specific Logic. The designer should implement the generators in certain manner in order to insure full compliance with the Design For Test (DFT) methodology.

```
// clk is the synchronous clock

wire clk; // T0 clock
wire clk_b; // T1 clock

assign clk = scan_mode ? scan_clk: ~gclkw; // The gclkw is the global
                                                                    // T1 clock
assign clk_b = scan_mode ? scan_clk: gclkw;

// aclk is the asynchronous clock (from external pin or
// internally generated)

wire aclk;
wire aclk_b;

assign aclk = scan_mode ? scan_clk: clksrc;
assign aclk_b = scan_mode ? scan_clk: ~clksrc;
```

Figure 12-36.

12.10.4.2 Clock Disable Logic

All the Customer Specific Logic synchronous clocks (or at least most of them) may be gated, especially when the New Custom Module is not enabled.
The clock gating should comply with the DFT methodology.

```
reg clkdis;                                    // Clock disable signal

always @(posedge clk or posedge reset)
if (reset) clkdis <= 1'b1;                                    // clocks disabled at reset
else clkdis <= new_module_en;

wire mainclk; // Main module's clock. Inactive if the module is not enabled
wire gatedaclk; // Generic clock with a certain gating function

assign mainclk = scan_mode ? scan_clk: (~clkdis & clk);
assign gatedaclk = scan_mode ? scan_clk: (~clk_gate & aclk);
```

Figure 12-37. Clock Gating.

12.10.5 Shared Memory Logic

TBD

TBD.

Chapter 13
Layer 1 Encryption Module (LEM)

Revision History

Revision	Date	Author	Changes
0.0	2/22/02	Pritam Kabe	Initial Release.Taken from Neptune LT Baseband IC specification.
0.1	03/14/02	Pritam Kabe	Initial version for Neptune LTS: Removed the Prescaler bits from the control register. Clock and DFT methodology different from Neptune LT. Removed the unwanted ports. Replaced or renamed the appropriate ports to match the latest STUB.
0.2	06/20/02	Pritam Kabe	Updated per DDTS# DSPhl14833. Removed the stck bit i.e. the bit11 of the Control Register. This bit does not exist in the new RTL taken from Patriot Indy for Neptune LTS.
0.3	05/07/03		Updated for LTE specification release.

13.1 Module Overview

13.1.1 General Description

The Layer 1 Encryption module (LEM) is a hardware accelerator for the A5/1 and A5/2 cipher algorithm and the FIRE error-correction code.

13.1.1.1 A5 Ciphering

When used for A5 ciphering, the software supplies the 64-bit cipher key and a 22-bit count value is derived from the TDMA frame number. After the data is written to the data registers, the LEM is configured via the CNTRL register. Once configured, the clk_en bit and the enable bit are set to start the processing. When the LEM finishes, the data registers contain what are known as BLOCK1 and BLOCK2. BLOCK1 is used to decipher data from a downlink connection. BLOCK2 is used to encipher data to an uplink

connection. If the software needs to decipher data, it reads BLOCK1 and xor's it with the cipher text, generating the clear text. If the software needs to encipher data, it reads BLOCK2 and xor's it with the clear text, generating the cipher text.

13.1.1.2 FIRE Encode/Decode

When using the FIRE algorithm, the software supplies the data to be processed by writing it to the data registers of the LEM. If FIRE encode is selected, 184 bits should be written to the data registers. The parity bits will be generated by the LEM and placed in the output registers, bits 184 to 223. If FIRE decode is selected, 224 bits should be written to the data registers: 184 data bits and 40 parity bits. Once the data is written, the LEM needs to be configured for operation and enabled.

- If the LEM is to perform encode, the original data will still be in bit locations 0 to 183 in the data registers. Bit locations 184 to 223 in the data registers are the generated parity bits.
- If the LEM ran a fire decode on the data, the status flags in the STAT register show the results of the decode operation. If the data contained an uncorrectable error, then the data registers contain invalid data due to errors and should not be used. If the data contained correctable errors, bits 0 to 183 of the data registers contain the corrected data. Bits 184 to 223 are undefined, and should not be used. If no errors were found, then bits 0 to 183 of the data registers contain a copy of the original good data while bits 184 to 223 are undefined and should not be used.

13.1.2 Connectivity

The LEM is connected to the DSP56600 via the Synthesizable Peripheral Module Bus Interface (S-PMBIF). The S-PMBIF is described in the SPMBIF_BlockUserGuid, SPMBIF_CreationGuide, SPMBIF_IntegrationGuide, and the SPMBIF spec. A top level view of the LEM is shown in Figure 13-1. The figure shows the main blocks in the LEM and the basic interconnect between them.

Figure 13-2 shows the top level I/O connection between the LEM and the S-PMBIF. Table 13-1 lists the I/O and gives a brief description of each signal. It is worth mentioning that all connections to the S-PMBIF are through the Register block.

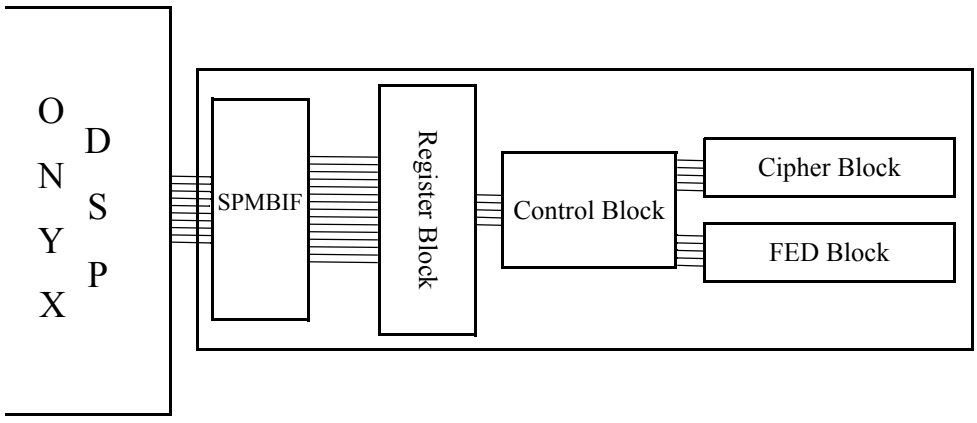


Figure 13-1. Top Level LEM Overview

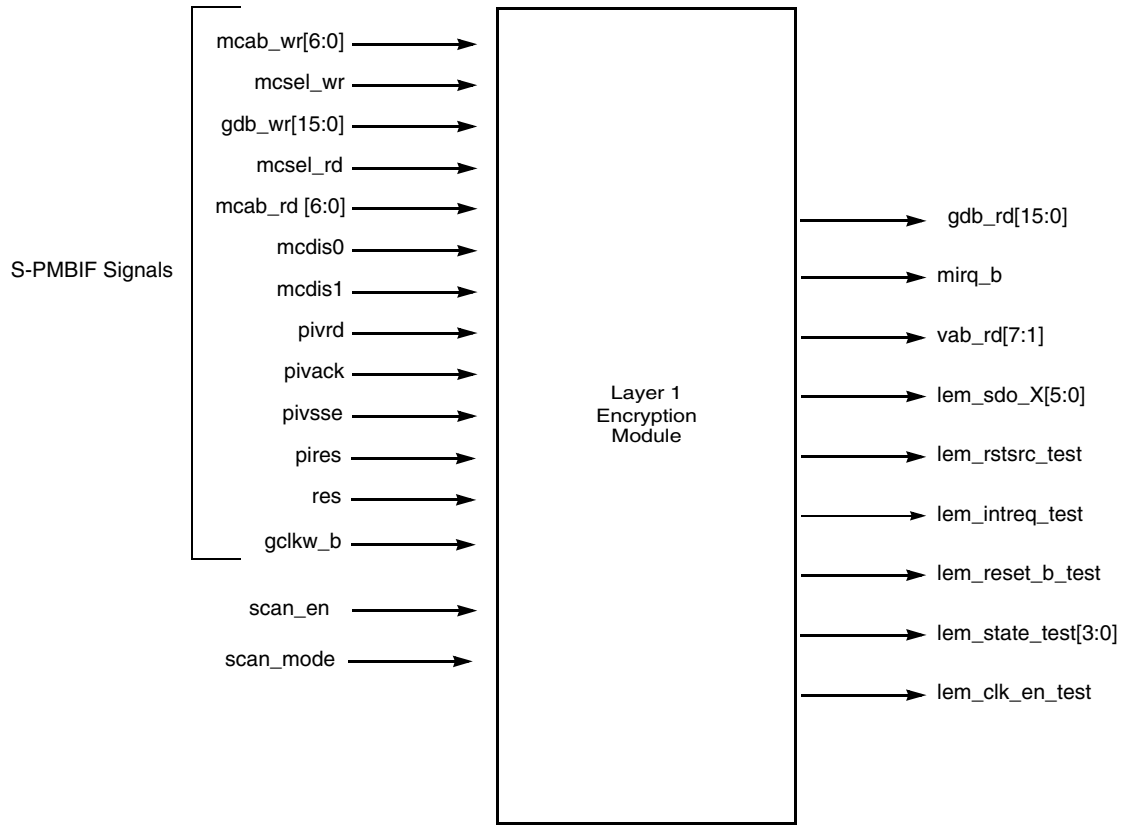


Figure 13-2. Layer 1 Encryption Module I/O

13.2 LEM Module Pin List

Table 13-1 is a listing of all the pins in the Layer 1 Encryption Module.

Table 13-1. LEM Module Pin List

Pin Name	Direction	Description	Active level
gdb_rd [15:0]	OUTPUT	Register read data bus	-
mirq_b	OUTPUT	LEM interrupt request line	L
vab_rd [7:1]	OUTPUT	Vector Address Bus Indicating Interrupt Vector	-
lem_sdo_X [5:1]	OUTPUT	Scan data out. X = 1..6	-
lem_rstsrc_test	OUTPUT	For Debug Only -- Signal from the S-PMBIF used to automatically clear the interrupt request	H
lem_intreq_test	OUTPUT	For Debug only -- lem interrupt request to S-PMBIF (debug signal)	H

Table 13-1. LEM Module Pin List

Pin Name	Direction	Description	Active level
lem_reset_b_test	OUTPUT	For debug only -- global reset (debug signal)	L
lem_state_test [3:0]	OUTPUT	For debug only -- FSM current state. (debug signal))	-
lem_clk_en_test	OUTPUT	Encryption clock. (debug signal)	-
mcab_wr[6:0]	INPUT	Register write address bus	-
mcsel_wr	INPUT	X or Y I/O write select	H
gdb_wr[15:0]	INPUT	Register write data bus	-
mcab_rd[6:0]	INPUT	Register read address bus	-
mcsel_rd	INPUT	X or Y I/O read select	H
mcdis0	INPUT	T0 clock disable	H
mcdis1	INPUT	T1 clock disable	H
pivrd	INPUT	Vector Read Strobe	H
pivack	INPUT	Peripheral module interrupt acknowledge	H
pivsse	INPUT	Interrupt vector source select	H
scan_en	INPUT	Signal indicating module is in scan	H
scan_mode	INPUT	Signal indicating chip is in scan	H
scan_lem_sdi_X[0:5]	INPUT	Scan data in. X = 1...6	-
scan_lem_sdi_X_bypass	INPUT	Scan bypass data input. X=1...6	-
gclkw_b	INPUT	Functional clock input	-
pires	INPUT	Hard reset	H
res	INPUT	Peripheral reset	H

13.3 Encryption Module Architecture

Figure 13-3 shows the connectivity between the different blocks in the LEM.

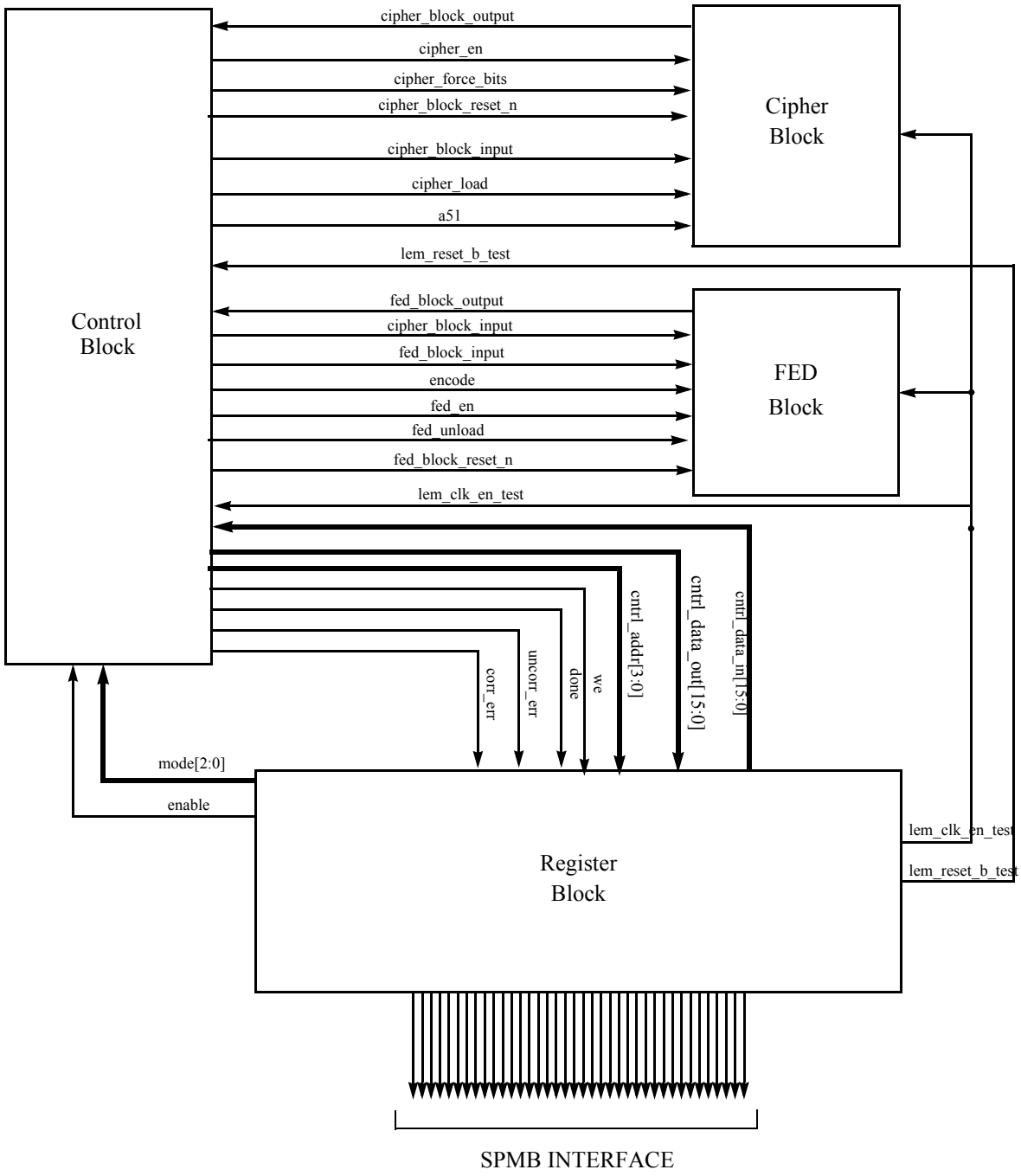


Figure 13-3. Block Connectivity of Major Signals

13.4 LEM Block Description

13.4.1 Control Block

The control block contains the state machine and logic necessary to control cryptographic block generation (A5/1 or A5/2) and data validation (FIRE encode or FIRE decode). Figure 13-4 shows the block level I/O, and Table 13-2 lists the I/O. The FSM is broken into three sub-parts: Fire encode, Fire decode, and Cipher block generation.

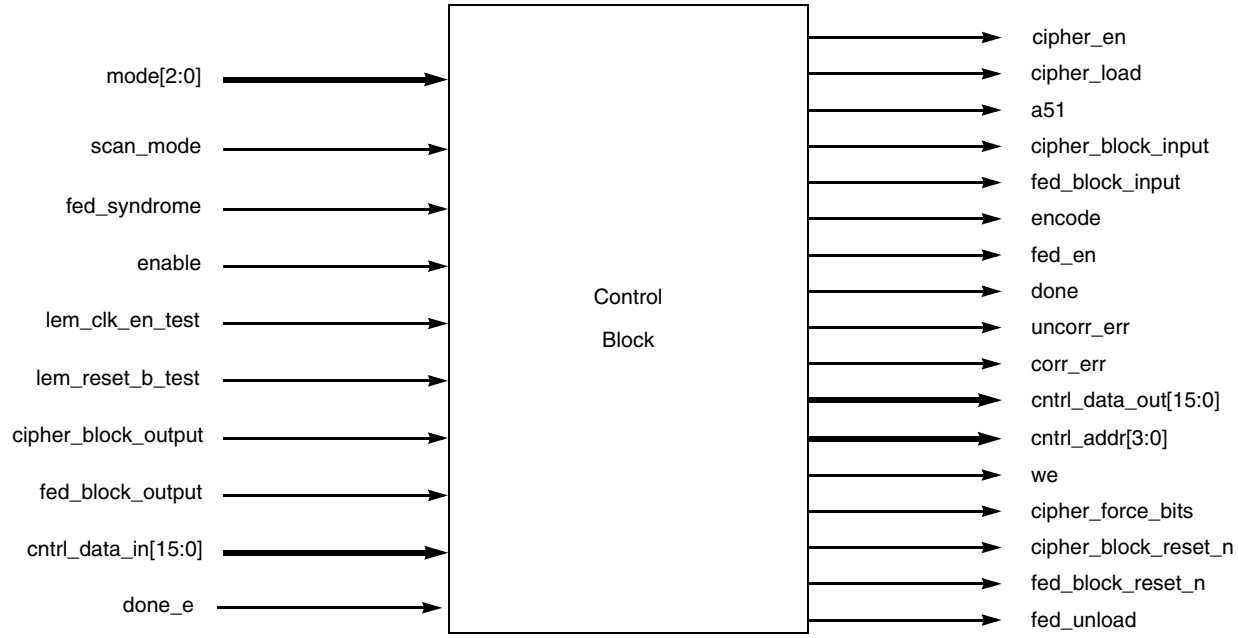


Figure 13-4. Control Block I/O Diagram

Table 13-2. Control Block I/O List

Name	Active level	Type	Description
cipher_en	H	OUTPUT	Signal to Cipher_Block to process data
cntrl_data_out[15:0]	-	OUTPUT	Data bus for writes to output register file
cipher_load	H	OUTPUT	Load signal to the Cipher block. This signal is used to load the data into the cipher block in accordance with the A5 algorithm
a51	H	OUTPUT	Cipher algorithm selection signal. If A51 = 1, use A5/1 algorithm; else use A5/2
cipher_block_input	-	OUTPUT	Serial data from the Control block to the Cipher block
fed_block_input	-	OUTPUT	Serial data from the Control block to the FED block
encode	H	OUTPUT	If encode = 1, use encode algorithm; else used decode algorithm

Table 13-2. Control Block I/O List (Continued)

Name	Active level	Type	Description
fed_en	H	OUTPUT	Enable signal to FED block
done	H	OUTPUT	Signal to the STAT register to set the done bit and generate interrupt if enabled
uncorr_err	H	OUTPUT	Uncorrectable errors bit in STAT register
corr_err	H	OUTPUT	Correctable errors bit in STAT register
cntrl_data_out[15:0]	-	OUTPUT	Data from the Control block. This data has been processed, and is headed to the output data registers in the register block
cntrl_addr[3:0]	-	OUTPUT	Address to write the cntrl_data_out to
we	H	OUTPUT	Active high write enable signal. Used to signal when the register block's data registers should latch in data
cipher_force_bits	H	OUTPUT	Signal from FSM to cipher block. Used for A5/2 cipher algorithm to set certain bits in the LFSRs according to spec
cipher_block_reset_n	L	OUTPUT	Cipher block reset signal. Logical OR of the system reset signal, xrsn, and the output of the FSM cipher block reset
fed_block_reset_n	L	OUTPUT	FED block reset signal. Logical OR of the system reset and the output of the FSM
fed_unload	H	OUTPUT	Signal to FED block used to set the LFSR into a shift register only mode. This allows the data in the LFSR to be shifted out serially without the effect of the XOR gates
mode[2:0]	-	INPUT	The mode bits from the CNTRL register. See Section 13.6.2 for a more detail discussion on what the mode bits mean
tck	-	INPUT	Test clock for scan testing
scan_mode_en	H	INPUT	Active high scan mode enable. This signal is the logical AND of the top level scan_mode and scan_en from the register block
fed_syndrome	H	INPUT	Signal from FED block. High when the low 28 bits of the FIRE LFSR are zero.
enable	H	INPUT	The enable bit from the CNTRL register
lem_clk_en_test	-	INPUT	The encryption clock divided from gclkw_b. This clock is enabled after the enable bit is set, and disabled when the DONE bit has been set
lem_reset_b_test	L	INPUT	The encryption reset signal. This signal is a logical or between the XIO reset signal XRSN, and the reset bit in the CNTRL register

Table 13-2. Control Block I/O List (Continued)

Name	Active level	Type	Description
cipher_block_output	-	INPUT	The serial output data from the Cipher block to the Control block
fed_block_output	-	INPUT	The serial output data from the FED block to the Control block
cntrl_data_in[15:0]	-	INPUT	Data bus for reads from the input register file
done_e	H	INPUT	done signal synced to lem_clk_en_test domain

The Control Block contains a Finite State Machine (FSM) that controls the LEM. The FSM controls the Cipher Block and the FED Block control signals. It also handles reading data from the data registers, and writing data to the data registers in the Register Block.

In addition to the FSM, the control block has 5 other registers to supplement the FSM. Figure 13-5 shows the control block and it's supplemental registers: SR_out, SR_in, bit_cntr, cntrl_addr, and err_cntr.

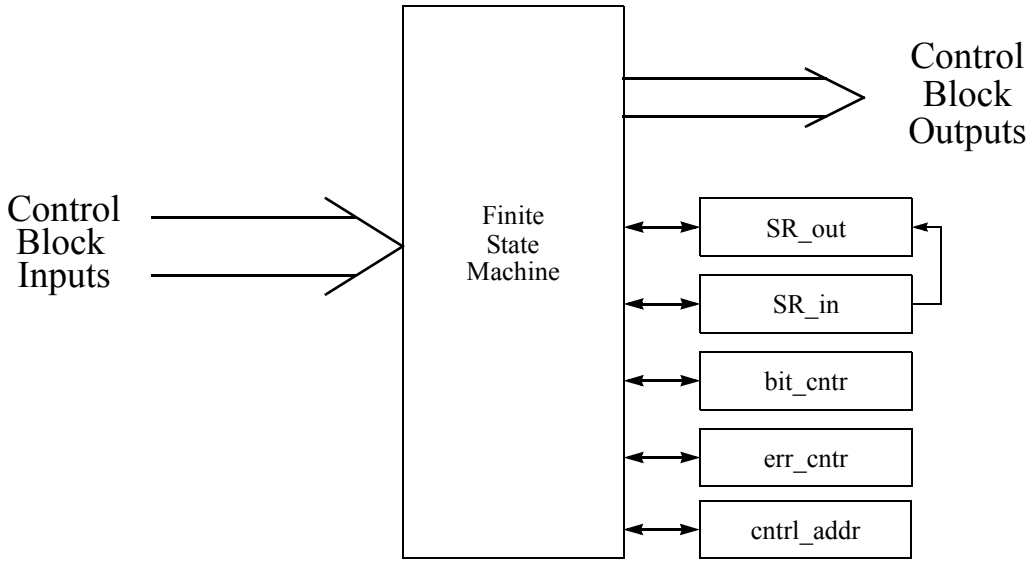


Figure 13-5. Internal Control Block Diagram

The five supplemental registers are:

- bit_cntr
  9-bit counter used to keep track of the position of the FSM relative to the data bits.
- SR_in
  16-bit Parallel-In-Serial-Out (PISO) shift register. This is used to parallel load the input data. It is shifted, least significant bit (LSb) first, into the cipher block, FED block, or SR_out. When the last bit is exhausted from SR_in, it is reloaded from the input data.
- SR_out
  16-bit Serial-In-Parallel-Out (SIPO) shift register. This is used to perform the serial to parallel conversion of the output from the cipher block, FED block, or the SR_in shift register. When full, its contents are parallel written to the appropriate data register location, as determined by the current value in bit_cntr.

- err_cntr
  4-bit register used for holding the error count when running FIRE decode. Once a FIRE decode sequence has found the syndrome (lower 28 bits of LFSR zero), the FSM corrects the next 12 bits in the data stream. When err_cntr is exhausted, the FSM stops correcting the data and moves any remaining data to the data registers in the Register Block.
- cntrl_addr
  This is the address used by the Register block's input and output registers. It is generated based off the bit_cntr value.

The FSM is broken into 3 sub-branches: FIRE encode, FIRE decode, and Cipher block generation. Note that A5/1 and A5/2 are both contained in the Cipher block generation branch of the FSM. The following sections discuss the 3 sub-branches.

13.4.1.1 FIRE Encode FSM Branch

Table 13-3 shows the transition mnemonic definition. This mnemonic definition is only valid for FIRE encode. Figure 13-6 shows the state transition diagram for both FIRE encode and FIRE decode. All encode states are prefixed with 'encode_' while all decode states are prefixed with 'decode_'. A more detailed discussion of each state of the encode branch, and what it does, is given below:

- idle
  - if enable not set, next state is idle
  - if enable is set, next state depends on mode bits, performs following actions:
    reset bit_cntr
    reset appropriate LFSR
    parallel load SR_in
    clear STAT register
- encode_Pre_load
  - increment bit_cntr
  - enable encode polynomial
- encode_Load_data
  - serial shift SR_in to the right so the next data bit is presented to the FED block.
  - if bit_cntr [3:0] is 0Fh, parallel load SR_in from the next input data set.
  - increment bit_cntr
  - load single bit SR_out from SR_in.
  - when SR_out is full, parallel store the appropriate output data location.
  - enable encode polynomial
- encode_Write_parity
  - invert contents of LFSR, store to appropriate out registers
  - If out_bit_rev is low, the data will be bit-flipped. If out_bit_rev is high, the data will not be bit-flipped.

In normal operation (out_bit_rev=0) the parity bits will be bit-flipped. In bit-reversed operation (out_bit_rev=1), the parity bits will not be bit-flipped.

- FSM_done
  - assert the done bit

Table 13-3. FIRE Encode Transition Mnemonic Definition

Input Name	Definition
FE_BIT_CNT	Asserted true (1) when the bit_cntr == 182. Bit 182 is the second to last bit that should be processed when generating the parity bits

13.4.1.2 FIRE Decode FSM Branch

Table 13-4 shows the transition mnemonic definition. This mnemonic definition is only valid for FIRE decode. Figure 13-6 shows the state transition diagram for both FIRE encode and FIRE decode. All encode states are prefixed with ‘encode_’ while all decode states are prefixed with ‘decode_’. A more detailed discussion of each state and what it does is given below.

- idle
  - if enable not set, next state is idle
  - if enable is set, next state depends on mode bits, performs following actions:
    reset bit_cntr
    reset appropriate LFSR
    parallel load SR_in
    clear STAT register
- decode_Pre_load
  - increment bit_cntr
  - enable both decode and encode polynomial
- decode_Load_data
  - serial shift SR_in to the right so the next bit is presented to the FED block
  - if bit_cntr[3:0] is 0Fh, parallel load SR_in from the next data set
  - increment bit_cntr
  - enable both encode and decode polynomial
- decode_Pre_syndrome
  - If FIRE LFSR[27:0] (syndrome) is zero, save the upper 12 bits of the LFSR to err_bits.
  - If not zero, set bit_cntr to 1, enable encode polynomial, disable decode polynomial, set input data to zero.
  - parallel load SR_in from first location of input data set
- decode_Syndrome_chk
  - set input data to zero
  - increment bit_cntr
  - enable encode polynomial
  - clock the LFSR until the syndrome becomes zero, or bit_cntr == 223. If the syndrome never

becomes zero, an uncorrectable error has occurred. If the syndrome becomes zero, save the upper 12 bits of the LFSR to err_bits.

- Serial shift SR_in to the right so the next bit is presented to SR_out as input.
- if bit_cntr[3:0] is 0Fh, parallel load SR_in from next input data set
- if bit_cntr[3:0] is 00h, parallel store SR_out to next output data set

- decode_Error_correct
  - xor data out from SR_in with the MSb of err_bit register. Use this as the input to SR_out.
  - left shift a zero into the LSb of err_bits register.
  - serial shift SR_in to the right so the next bit is presented to the SR_out register
  - if bit_cntr[3:0] is 0Fh, parallel load SR_in from the next input data set
  - if bit_cntr[3:0] is 00h, parallel store SR_out to the next output data set
  - increment bit_cntr
- decode_Last_store
  - parallel store SR_out register to last output data set location
- decode_Uncorrectable_errors
  - assert the UE bit in the STAT register
  - perform a final store of the SR_out register.
- FSM_done
  - done bit asserted

Table 13-4. FIRE Decode Transition Mnemonic Definition

Input Name	Definition
FD_BIT_CNT	Asserted true (1) when the bit_cntr == 222. Bit 222 is the second to last bit that should be processed when decoding the data
SYN_ZERO	Asserted true (1) when the syndrome (lower 28 bits of the FIRE LFSR) is zero.

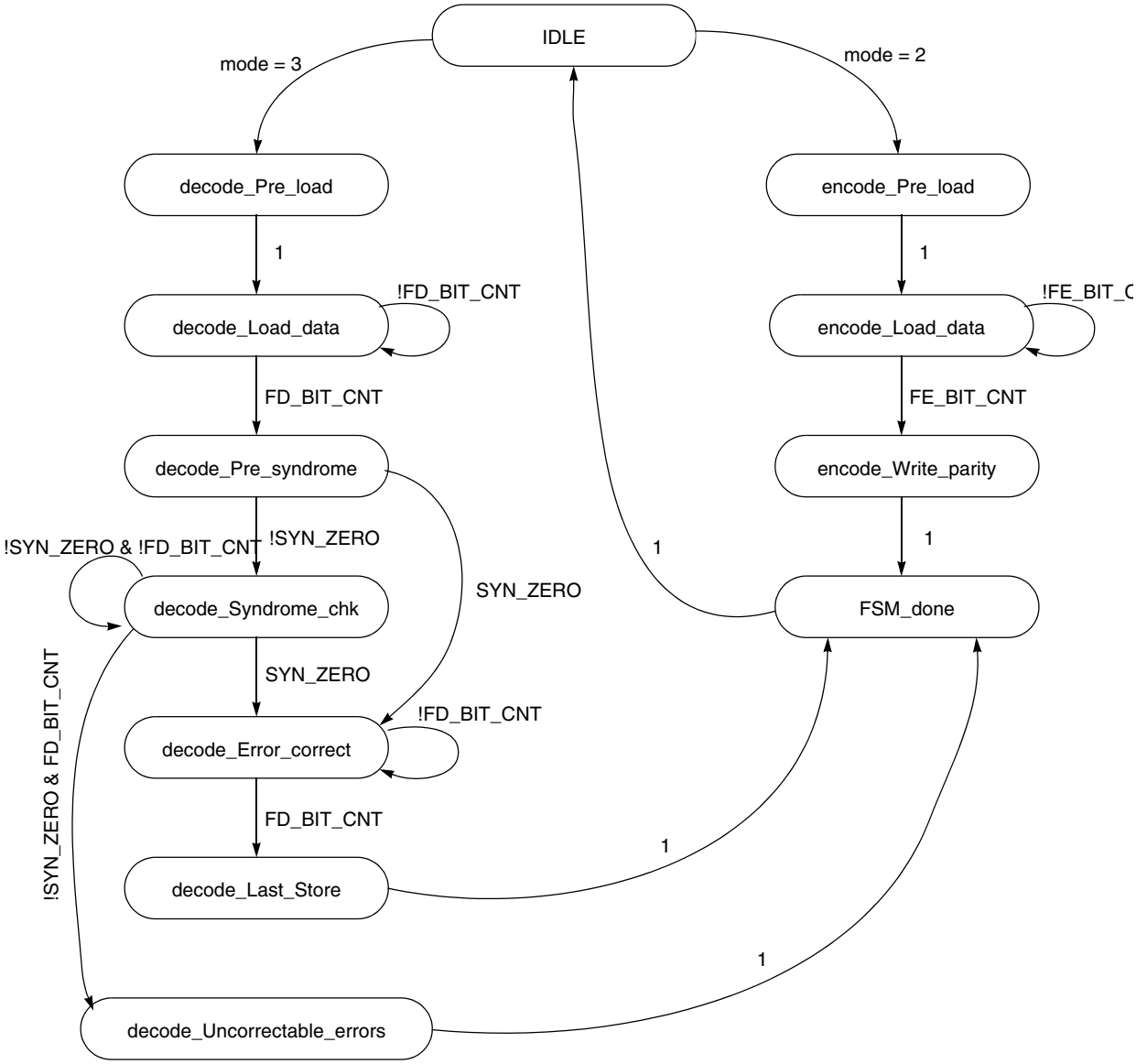


Figure 13-6. Fire Portion of FSM

13.4.1.3 Cipher Block Generation FSM Branch

Figure 13-7 shows the state diagram of the cipher portion of the FSM. Table 13-5 lists the mnemonic definition. This mnemonic definition is valid for only the cipher portion of the FSM. A detailed discussion of each state is given below.

- idle
  - if enable not set, next state is idle
  - if enable is set, next state depends on mode bits, perform following actions:
    reset bit_cntr
    reset appropriate LFSR
    parallel load SR_in
    clear STAT register
- cipher_Load
  - load data into the LFSRs
  - reset the bit_cntr register when BIT_CNT
- cipher_Runup
  - cycle the LFSRs
  - reset bit_cntr register when BIT_CNT
- cipher_Force
  - Set/Clear certain bits in LFSRs
- cipher_Block1
  - cycle LFSRs 114 times to produce BLOCK1
  - reset bit_cntr register when BIT_CNT
  - cipher_Block2
  - cycle LFSRs 114 times to produce BLOCK2
- FSM_done
  - done bit asserted

Table 13-5. Cipher Mnemonic Definition

Input Name	Description
C_BIT_CNT	The bit_cntr register has reached a defined value. This value can change between states
A51_SEL	Generated from the mode bits. Indicates that the cipher algorithm chosen is A5/1

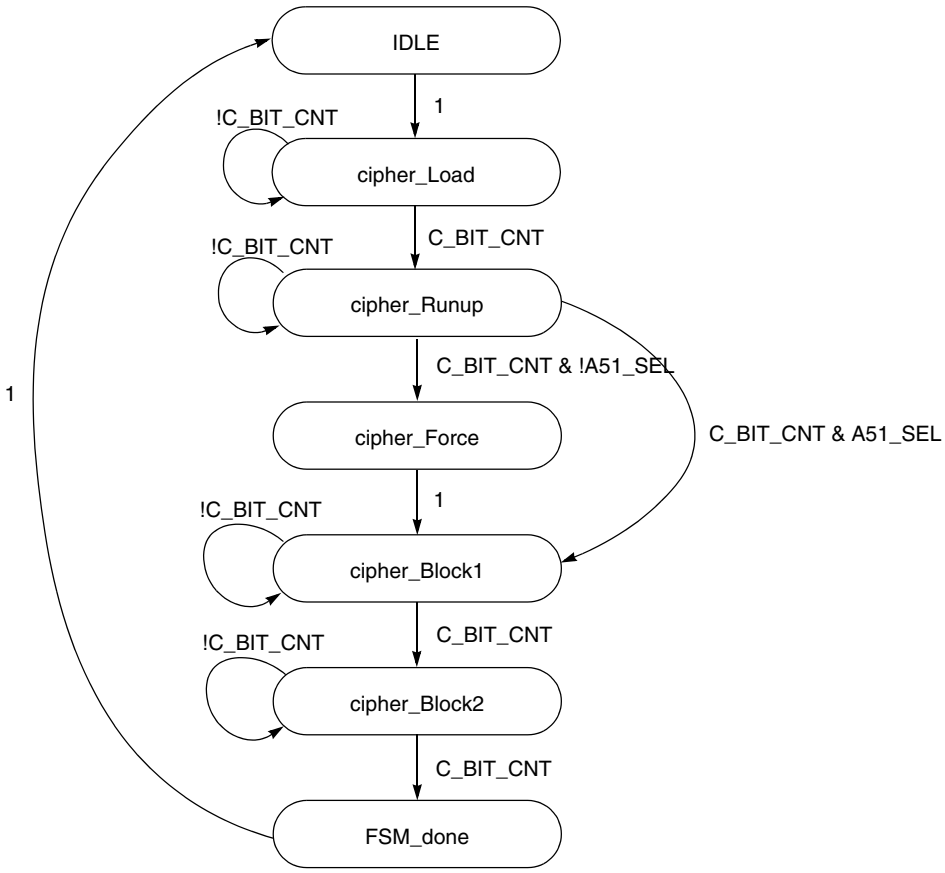


Figure 13-7. Cipher Portion of FSM

13.4.2 Register Block

The register block contains the register file used to hold all input and output data, as well as the LEM interface registers. Figure 13-8 shows the block level I/O, and Table 13-6 gives a list and short description of each signal. The S-PMBIF input signals are listed separately. For more information on the S-PMBIF input signals, see Table 13-1 on page 13-3.

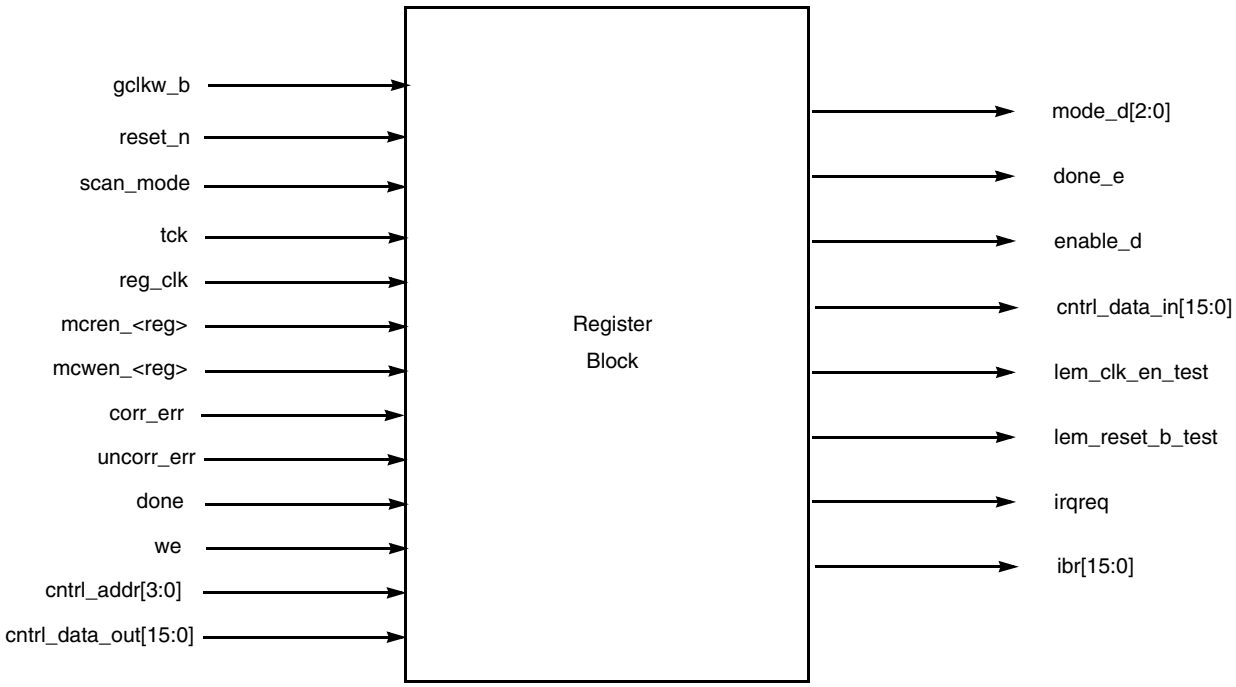


Figure 13-8. Register Block I/O Schematic

Table 13-6. Register Block I/O List

Name	Active level	Type	Description
mode_d[2:0]	-	OUTPUT	Mode register bits. Determines if the LEM will be performing FIRE encode, decode, A5/1 or A5/2 generation
done_e	H	OUTPUT	Done bit synced to lem_clk_en_test domain
enable_d	H	OUTPUT	Enable register bit
ibr[15:0]	-	OUTPUT	S-PMBIF read data bus
cntrl_data_in[15:0]	-	OUTPUT	Data bus for reads from input registers
irqreq	H	OUTPUT	INT line to S-PMBIF. Asserted until lem_rstsrc_test asserted
lem_clk_en_test	-	OUTPUT	Encryption clock selected by the clk_sel register bits
lem_reset_b_test	L	OUTPUT	The reset signal from the XIO XRSN signal or the reset bit
gclkw_b	-	INPUT	Raw dsp clock
reset_n	L	INPUT	Peripheral reset
scan_mode	H	INPUT	Scan mode signal
tck	-	INPUT	Test Clock
reg_clk	H	INPUT	Register write pulse
mcren_<reg>	H	INPUT	Register read enable

Table 13-6. Register Block I/O List (Continued)

Name	Active level	Type	Description
mcwen_<reg>	H	INPUT	Register write enable
corr_err	H	INPUT	Correctable errors register bit input
uncorr_err	H	INPUT	Uncorrectable errors register bit input
done	H	INPUT	Done register bit input
we	H	INPUT	Write enable from cntrl block
cntrl_addr[3:0]	-	INPUT	Register address from control block
cntrl_data_out[15:0]	-	INPUT	Data bus for writes to output registers

All writable registers are clocked off a strobe from the S-PMBIF. They are asynchronous to the operation of the LEM. Since the registers are asynchronous, they are still clocked when the LEM is disabled (enable = 0) and when the LEM clock is turned off (clk_en = 0)

The CNTRL register is synchronized by a clock enabling scheme. The data registers are not synchronized at all. In the programming model, the input data is written to the data registers before the *enable* bit is set. By the time the LEM recognizes the *enable* bit, the data should be stable and unchanging. The same logic holds true for the output data in the data registers. By the time the LEM signals that it is done processing, the data is stable and unchanging. For this reason, polling on the output data while processing is not supported.

WARNING:

Reading the data registers while processing is not supported. Erratic results may occur.

13.4.2.1 CNTRL Register

The CNTRL register controls the setup and operation of the LEM. Notable changes from the previous version of the LEM (WCP 1.0 ENCR block):

1. The enable bit controls the gating of the LEM's operational clock. Since the enable bit is the last bit to be written in the software programming model (see Section 13.5), this effectively synchronizes all CNTRL register bits without the use of flip-flops as synchronizers.
2. The clk_en bit is the master clock enable bit. This bit must be enabled to turn on the LEM's clock. See the programming model, Section 13.5 for more detailed information.

13.4.2.2 STAT Register

The stat register is a read only register.

The STAT register bits are cleared when the enable bit is set. However, it takes 2 DSP clock cycles for the done bit to clear. As such, software must allow 2 DSP clock cycles between setting the enable bit and polling on the done bit.

WARNING:

Software must allow a minimum of two dsp_clk cycles between the setting of the enable bit, and any polling on the done bit. Failure to do so may cause software to see the done bit set incorrectly on the first read.

13.4.3 Cipher Block

13.4.3.1 Description

The cipher block is used to generate the BLOCK1 and BLOCK2 data using either the A5/1 or A5/2 cipher algorithm. When the cipher block is finished, it will set the *done* bit in the STAT register. If enabled (via the irq_en bit), it will also generate an interrupt to the DSP.

Figure 13-9 shows the I/O around the cipher block. Table 13-7 lists the I/O and gives a short description. The output of the cipher block is a serial stream of data. Inspection of Figure 13-3 on page 13-5 will show that the cipher block does not write to the output register file. The control logic FSM handles the serial-to-parallel conversion and all writes to the register file.

The A5/1 and A5/2 algorithms both use multiple Linear Feedback Shift Registers (LFSRs) and decision making combinational logic. While both algorithms utilize LFSRs, the tap points differ. Also, the decision making logic differs between the algorithms.

To reduce gate count, the cipher block uses one set of LFSRs for both algorithms. The tap points for the algorithms are selected when the software sets the *mode* bits in the CNTRL register (see Table 13-22 for more information). The *mode* bits also determine which blocks of combinational logic are used for decision making. Figure 13-10 is a block diagram showing the organization of the cipher block.

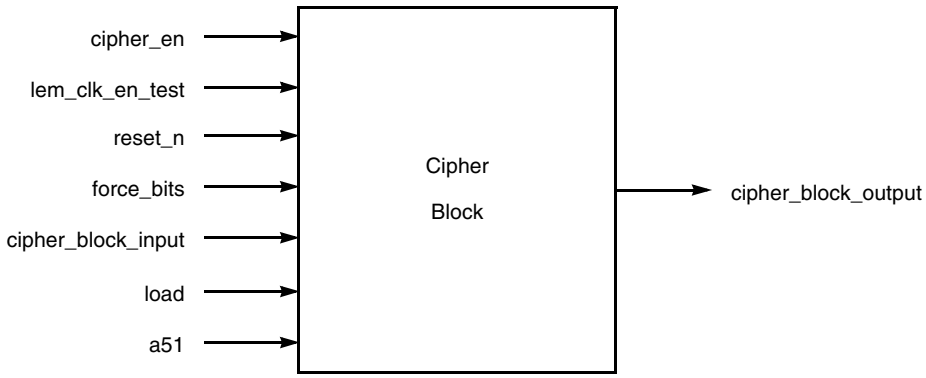


Figure 13-9. Cipher Block I/O Schematic

Table 13-7. Cipher Block I/O List

Name	Active level	Type	Description
cipher_block_output	-	OUTPUT	The current cipher bit generated by the block
cipher_en	H	INPUT	Signal to enable the cipher block
lem_clk_en_test	-	INPUT	The encryption module's internal clock
reset_n	L	INPUT	The encryption module's internal reset signal
force_bits	H	INPUT	Force data bits high for a52
cipher_block_input	-	INPUT	The data input
load	H	INPUT	Signal to cipher block to load the data
a51	H	INPUT	Signal to determine A5/1 or A5/2 algorithm. A5/1=1, A5/2=0

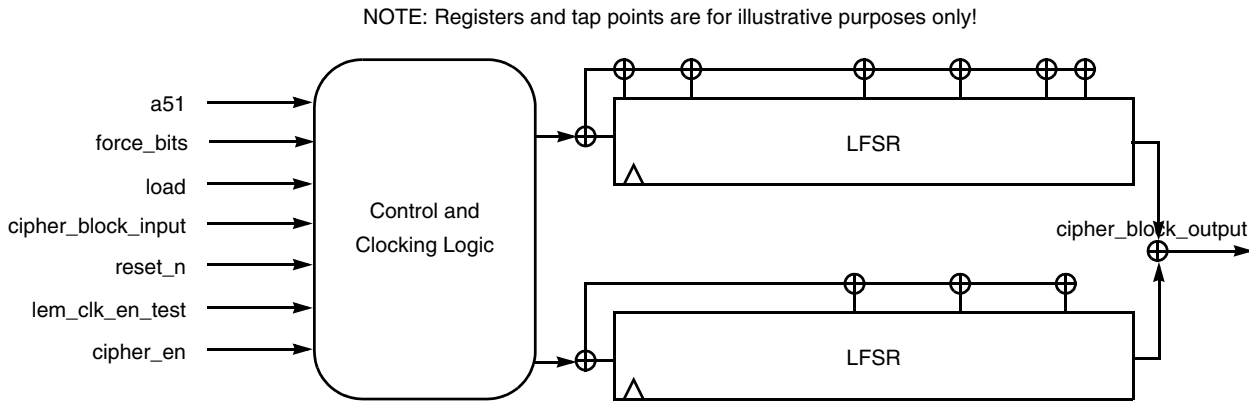


Figure 13-10. Cipher Block Architecture

13.4.4 FED Block (Fire Encode/Decode Block)

13.4.4.1 Description

The FED block is used to perform FIRE encode or decode on the data. The FED block can be used to either generate the FIRE parity for data encoding, or it can process the data with parity for decoding.

The FED block consists of a 40-bit LFSR with two distinct polynomials. The encode polynomial is used for generation of the 40 parity bits, while both the encode and decode polynomial are used for decoding.

The FED block works on a serial bit stream input. Figure 13-11 shows the LFSR used in the FED block. The data to be shifted in is presented to the FED block by the FSM of the control block, described in Section 13.4.1, “Control Block,” on page 13-6. During decode, the serial output stream is read by the control block, parallelized, and written to the register file. After encode, the register contents are shifted out and written to the appropriate bits in the register file.

Figure 13-12 shows the I/O connections, while Table 13-8 lists them.

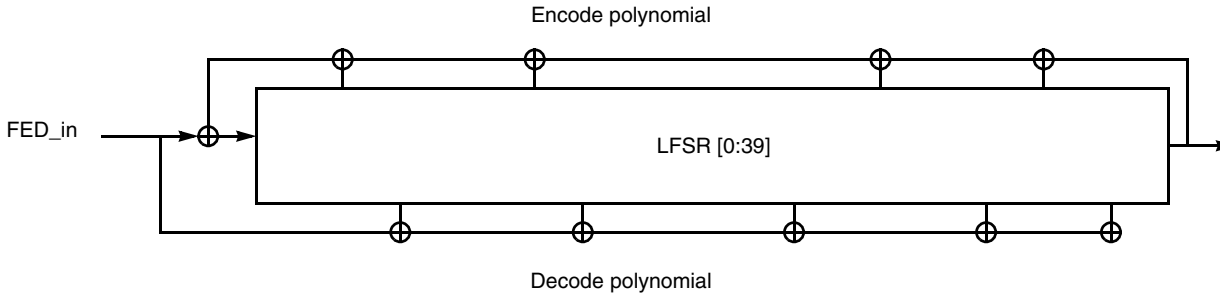


Figure 13-11. FED Block LFSR

The encode polynomial for FIRE is $X^{26} + X^{23} + X^{17} + X^3 + X^0$. The decode polynomial is $X^{39} + X^{33} + X^{29} + X^{27} + X^{16} + X^{10} + X^6 + X^4 + X^0$.

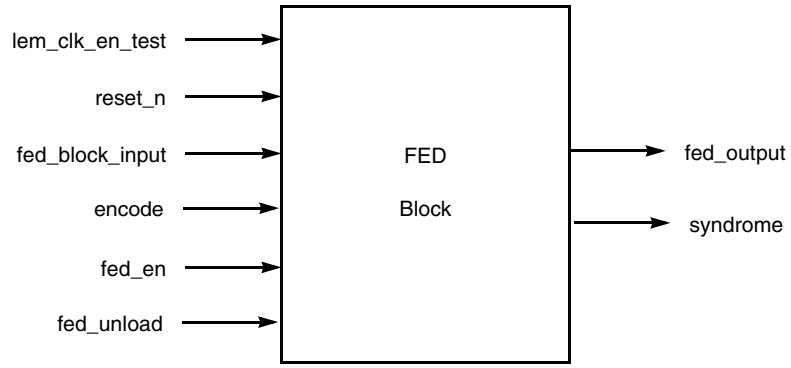


Figure 13-12. FED Block I/O

Table 13-8. FED Block I/O List

Name	Active level	Type	Description
fed_output	-	OUTPUT	The serial output of the MSb of the LFSR
syndrome	H	OUTPUT	We have aligned the error with the output
lem_clk_en_test	-	INPUT	The encryption module's internal clock
reset_n	L	INPUT	The encryption module's internal reset signal
fed_block_input	-	INPUT	The data input
encode	H	INPUT	Encode/Decode signal: 1 = encode, 0 = decode
fed_en	H	INPUT	FED enable signal
fed_unload	H	INPUT	LFSR should just shift data

13.4.4.2 FIRE Encode Process

This section details the FIRE encode process for any future designers that may need to know it.

The hardware's parity bit generation process is:

- Assert the fed_en signal. This enables the block and starts processing.
- Feed data stream to FED.
- When all 184 bits have been shifted into FED, deassert fed_en signal. This disables the processing.
- Invert LFSR contents. If out_bit_rev = 0, bit flip the parity bits and store to OUT registers. If out_bit_rev = 1, do not bit flip parity bits and store to OUT registers.

13.4.4.3 FIRE Decode Process

This section details the FIRE encode process for any future designers that may need to know it.

When FIRE decode is complete, the data in the data registers is the corrected data unless an uncorrectable error existed. Any errors will be flagged in the STAT register (Table 13-23). The hardware process for fire decode is given below.

- Assert the fed_en signal, and feed the data stream via fed_in.
- After all 224 data bits have been shifted into the FED, check the syndrome (lower 28 bits of LFSR). If the syndrome is zero, proceed to the correction phase.

- If the syndrome is not zero, set the encode signal to '1' to enable the encode polynomial. The encode polynomial will be used for the rest of FIRE decode. Also, make sure the fed_block_input bit is '0'. This forces the LFSR to consider only the data currently in the register.
- Continue clocking the register until either the syndrome becomes zero or the LFSR has cycled 224 times. If the LFSR cycles 224 times and the syndrome has not become zero, an uncorrectable error has occurred.
- As soon as the syndrome becomes zero, the erroneous data has been located in the data stream. The upper 12 bits are necessary to perform the corrections on the data. XOR the upper 12 bits with the next 12 data bits, MSb first. After the 12 bits are exhausted, simply move the rest of the input data to the output registers.
- After data moving is complete, the FIRE decode is finished.

13.5 Programming Model

This section contains the programming model for the FIRE encode/decode and the A5/1 and A5/2 cipher algorithms.

13.5.1 Cipher Block Programming

Here's the intended programming model:

1. Write cipher key, K_c, to data registers DATA0-DATA3. Table 13-9 shows registers DATA0 to DATA3 with K_c written if IBR = 0. Table 13-10 shows the registers written if IBR = 1. See Section , "LEM Control Register," on page 13-30 for a description of IBR and OBR.

Table 13-9. Writing of Cipher Key, K_c, if IBR=0

Register		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA0		K15	.	.	.	K11	.	.	.	K7	.	.	.	K3	.	.	K0
DATA1		K31	.	.	.	K27	.	.	.	K23	.	.	.	K19	.	.	K16
DATA2		K47	.	.	.	.	.	.	.	.	.	.	.	.	.	.	K32
DATA3		K63	.	.	.	.	.	.	.	.	.	.	.	.	.	.	K48

Table 13-10. Writing of Cipher Key, K_c, if IBR=1

Register		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA0		K0	K1	.	.	.	K5	.	.	.	K9	.	.	.	.	.	K15
DATA1		K16	.	.	.	.	.	.	.	.	.	.	.	.	.	.	K31
DATA2		K32	.	.	.	.	.	.	.	.	.	.	.	.	.	.	K47
DATA3		K48	.	.	.	.	.	.	.	.	.	.	.	.	.	.	K63

2. Write COUNT value into registers DATA4-DATA5. The COUNT value is written as T1:T3:T2. Table 13-11 shows DATA4 to DATA5 if IBR = 0. Table 13-12 shows the registers if IBR = 1.

Table 13-11. Writing of COUNT if IBR=0

Register		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA4		T1	.	.	T1	T3	.	.	T3	.	.	T3	T2	.	.	.	T2
DATA5												T1	.	.	.	.	T1

Table 13-12. Writing of COUNT if IBR=1

Register		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA4		T2	.	.	.	T2	T3	.	.	T3	.	.	T3	T1	.	.	T1
DATA5		T1	.	.	.	.	T1										

3. Setup the desired mode of operation in the CNTRL register by writing the mode bits. Detail on the CNTRL register can be found in Section , “LEM Control Register,” on page 13-30.
4. If desired, enable the generation of an interrupt when finished by setting *irq_en* in the CNTRL register.
5. Set the clk_en bit in the CNTRL register
6. Set the *enable* bit in the CNTRL register. This must be a separate write in order to avoid a potentially glitchy clock from the S-PMBIF.
7. After 2 dsp_clk cycles, the *done* bit can be polled on. If it is asserted the cipher algorithm is finished, and software may read BLOCK1 and/or BLOCK2.

Table 13-13. BLOCK1 output if OBR=0

Register		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA0		O15	.	.	O12	.	.	O9	.	.	O6	.	.	O3	.	.	O0
DATA1		O31	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O16
DATA2		O47	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O32
DATA3		O63	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O48
DATA4		O79	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O64
DATA5		O95	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O80
DATA6		O111	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O96
DATA7																O113	O112

Table 13-14. BLOCK1 output if OBR=1

Register		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA0		O0	.	.	O3	.	.	O6	.	.	O9	.	.	O12	.	.	O15
DATA1		O16	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O31
DATA2		O32	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O47
DATA3		O48	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O63
DATA4		O64	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O79
DATA5		O80	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O95
DATA6		O96	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
DATA7		O112	O113														

Table 13-15. BLOCK2 output if OBR=0

Register		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA8		O15	.	.	O12	.	.	O9	.	.	O6	.	.	O3	.	.	O0
DATA9		O31	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O16
DATA10		O47	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O32
DATA11		O63	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O48
DATA12		O79	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O64
DATA13		O95	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O80
DATA14		O111	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O96
DATA15																O113	O112

Table 13-16. BLOCK2 output if OBR=1

Register		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA8		O0	.	.	O3	.	.	O6	.	.	O9	.	.	O12	.	.	O15
DATA9		O16	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O31
DATA10		O32	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O47
DATA11		O48	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O63
DATA12		O64	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O79
DATA13		O80	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O95
DATA14		O96	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
DATA15		O112	O113														

WARNING:

All data needs to be written in accordance to the IBR bit in the CNTRL register so that the hardware knows which is the most significant bit. See Section , “LEM Control Register,” on page 13-30 for more information on the CNTRL register.

13.5.1.1 Cipher Block Programming Flow Diagram

Figure 13-13 shows a condensed programming flow diagram. For a more detail discussion of programming the Cipher block, please read Section 13.5.1, “Cipher Block Programming,” on page 13-20, Cipher block programming.

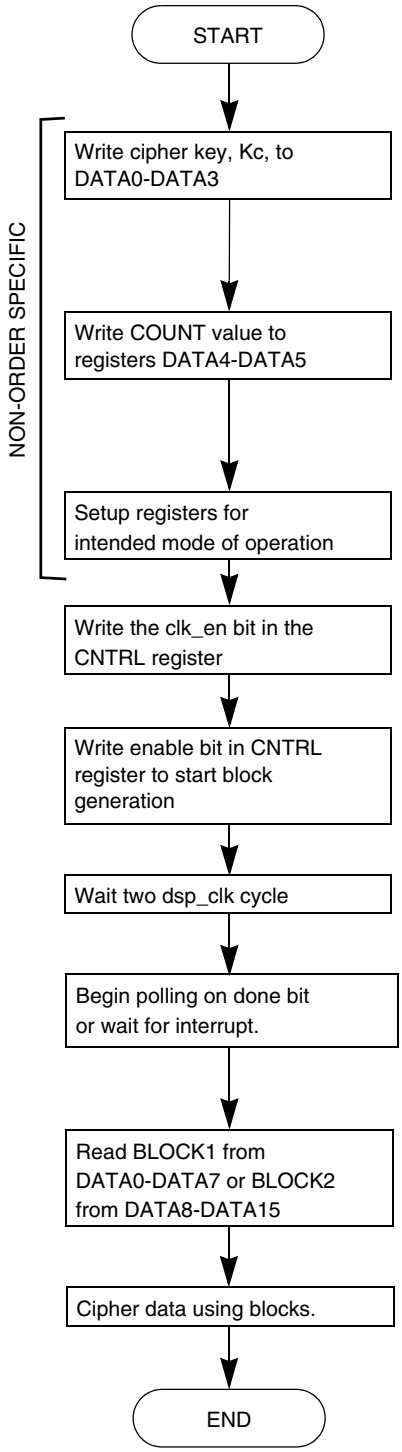


Figure 13-13. Flow Diagram of Cipher Block Usage

13.5.2 FED Block Programming

To use the FED block, write the data to be processed to the data registers, setup the CNTRL register (with mode, clk_en bits, etc.) and then set the enable bit to initiate the process.

The intended programming model is:

1. Write the data to be processed to the data registers. Table 13-17 shows the registers written for IBR = 0, and Table 13-18 shows the registers being written for IBR = 1.
   - — If encoding the data, only the first 184 bits need to be written.
   - — If decoding the data, all 224 bits need to be written

Table 13-17. Writing data to input for FED processing with IBR=0

Register		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA0		I15	.	.	I12	.	.	I9	.	.	I6	.	.	I3	.	.	I0
DATA1		I31	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I16
DATA2		I47	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I32
DATA3		I63	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I48
DATA4		I79	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I64
DATA5		I95	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I80
DATA6		I111	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I96
DATA7		I127	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I112
DATA8		I143	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I128
DATA9		I159	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I144
DATA10		I175	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I160
DATA11		I191	.	.	.	.	.	.	I184	I183	.	.	.	.	.	.	I176
DATA12		I207	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I192
DATA13		I223	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I208

Table 13-18. Writing data to input for FED processing with IBR=1

Register		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA0		I0	.	.	I3	.	.	I6	.	.	I9	.	.	I12	.	.	I15
DATA1		I16	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I31
DATA2		I32	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I47
DATA3		I48	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I63
DATA4		I64	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I79
DATA5		I80	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I95
DATA6		I96	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I111
DATA7		I112	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I127
DATA8		I128	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I143
DATA9		I144	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I159
DATA10		I160	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I175
DATA11		I176	.	.	.	.	.	.	I183	I184	.	.	.	.	.	.	I191
DATA12		I192	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I207
DATA13		I208	.	.	.	.	.	.	.	.	.	.	.	.	.	.	I223

2. Setup the desired configuration in the CNTRL register. Details on the CNTRL register can be found in Section , “LEM Control Register,” on page 13-30.
3. If desired, enable the generation of an interrupt when finished by setting *irq_en* in the CNTRL register.
4. Set the *enable* bit in the CNTRL register.

NOTE:

When generating parity, the parity bits will be the last 40 bits of the data registers (output bits 184 to 223). Therefore, only the last 3 data registers (DATA11, DATA12, and DATA13) need to be read to get the parity bits.

WARNING:

All data needs to be written in accordance to the IBR bit in the CNTRL register so the hardware knows what is the most significant bit.

5. Two dsp_clk cycle after the enable bit was set, the *done* bit can be queried. If asserted, processing is finished. If encoding data, software may read the data from the data registers. If decoding data, the UE bit in the STAT register should be read to ensure the data is good. If UE is clear (UE = 0), then the data can be read from the data registers. Table 13-19 shows the data in DATA0 - DATA13 if OBR = 0, and Table 13-20 shows the data if OBR = 1.

WARNING:

The software must wait at least two dsp_clk cycles after the enable bit is set before polling the done bit.

Table 13-19. FED output data if OBR=0

Register		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA0		O15	.	.	O12	.	.	O9	.	.	O6	.	.	O3	.	.	O0
DATA1		O31	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O16
DATA2		O47	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O32
DATA3		O63	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O48
DATA4		O79	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O64
DATA5		O95	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O80
DATA6		O111	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O96
DATA7		O127	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O112
DATA8		O143	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O128
DATA9		O159	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O144
DATA10		O175	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O160
DATA11		O191	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O176
DATA12		O207	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O192
DATA13		O223	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O208

Table 13-20. FED output data if OBR=1

Register		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA0		O0	.	.	O3	.	.	O6	.	.	O9	.	.	O12	.	.	O15
DATA1		O16	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O31
DATA2		O32	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O47
DATA3		O48	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O63
DATA4		O64	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O79
DATA5		O80	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O95
DATA6		O96	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O111
DATA7		O112	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O127
DATA8		O128	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O143
DATA9		O144	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O159
DATA10		O160	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O175
DATA11		O176	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O191

Table 13-20. FED output data if OBR=1 (Continued)

Register		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA12		O192	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O207
DATA13		O208	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O223

13.5.2.1 FED Block Programming Flow Diagram

Figure 13-14 shows a condensed flow diagram for programming the FED block. More detail can be found in Section 13.5.2, “FED Block Programming,” on page 13-24, FED block programming.

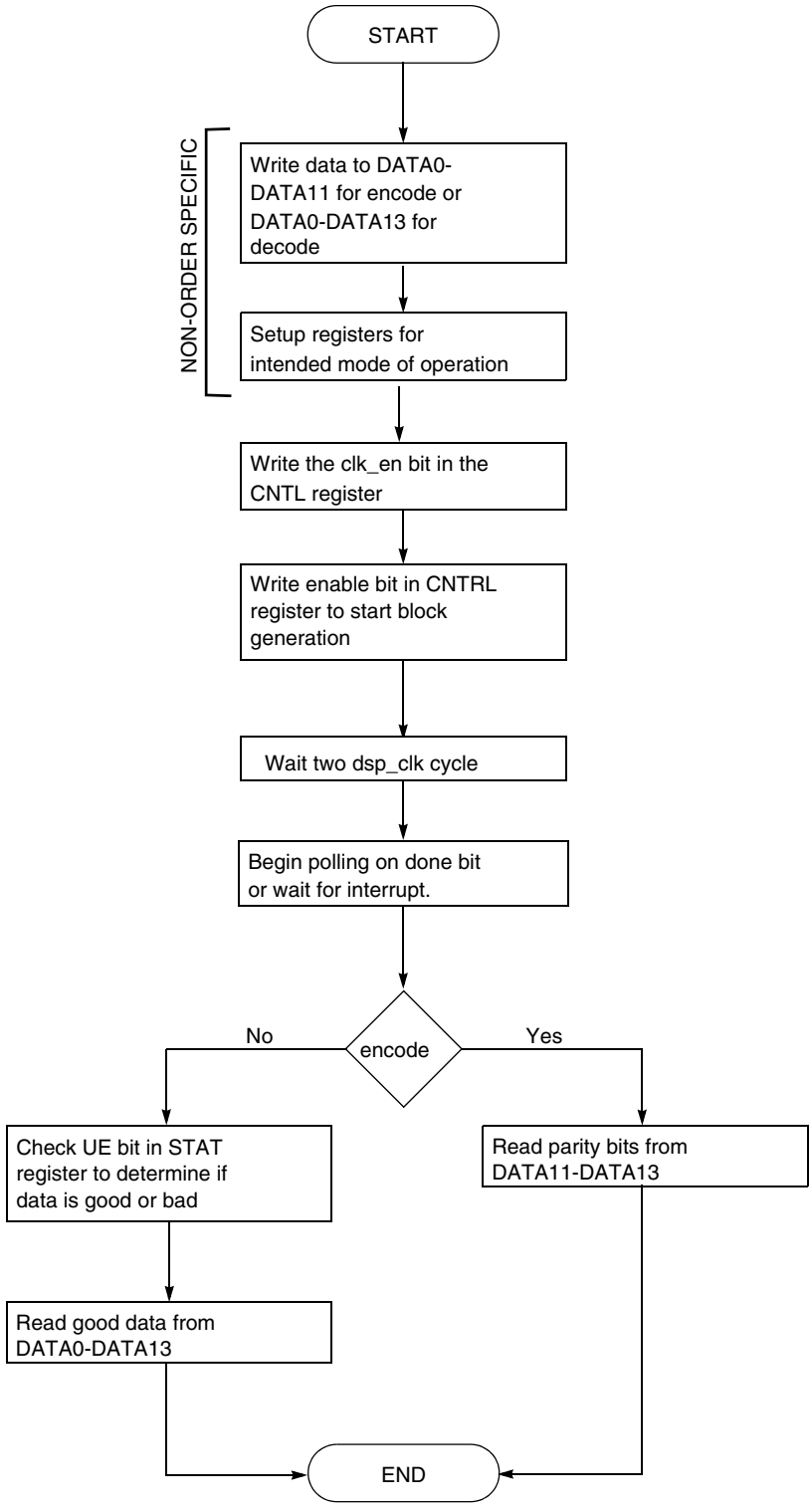


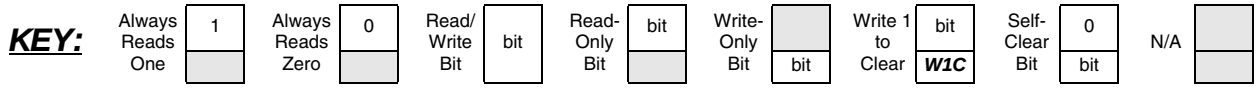
Figure 13-14. Flow Diagram of FED Usage

13.5.2.2 Low Power Mode Behavior

The LEM will process data whenever gclkw_b is active. If gclkw_b is gated while the LEM is processing data, it will continue processing from the point the clock was gated. For Neptune, gclkw_b is gated to the LEM in STOP mode and active during WAIT and RUN mode. This means, the LEM will be active in all but STOP mode.

13.6 LEM Register Summary

This section contains the register summary, as well as a detailed explanation of each register and its usage.



13.6.1 Register Summary

Table 13-21. LEM Register Summary Y:$FFA0- Y:$FFB1

Register		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTRL (Y:$FFA0)	R	0	0	0	0	0	0	0	irq_en	OBR	IBR	0	enable	clk_en	mode[2:0]		
	W											reset					
STAT (Y:$FFA1)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CE	UE	done
	W																
DATA0 (Y:$FFA2)	R	Data Register 0															
	W																
DATA1 (Y:$FFA3)	R	Data Register 1															
	W																
DATA2 (Y:$FFA4)	R	Data Register 2															
	W																
DATA3 (Y:$FFA5)	R	Data Register 3															
	W																
DATA4 (Y:$FFA6)	R	Data Register 4															
	W																
DATA5 (Y:$FFA7)	R	Data Register 5															
	W																
DATA6 (Y:$FFA8)	R	Data Register 6															
	W																
DATA7 (Y:$FFA9)	R	Data Register 7															
	W																
DATA8 (Y:$FFAA)	R	Data Register 8															
	W																
DATA9 (Y:$FFAB)	R	Data Register 9															
	W																
DATA10 (Y:$FFAC)	R	Data Register 10															
	W																
DATA11 (Y:$FFAD)	R	Data Register 11															
	W																
DATA12 (Y:$FFAE)	R	Data Register 12															
	W																
DATA13 (Y:$FFAF)	R	Data Register 13															
	W																

Table 13-21. LEM Register Summary Y:$FFA0- Y:$FFB1 (Continued)

Register		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA14 (Y:$FFB0)	R W	Data Register 14															
DATA15 (Y:$FFB1)	R W	Data Register 15															

13.6.2 Register Description

The following figures and associated text give detailed descriptions of the various LEM registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

CNTRL — LEM Control Register — **Addr Y:$FFA0**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
								irq_en	OBR	IBR	reset	enable	clk_en	mode[2:0]		
TYPE	r	r	r	r	r	r	r	rw	rw	rw	slfclr	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-22. CNTRL Description

Name	Description	Settings
Bits 15-12	Reserved	N/A
irq_en Bit 8	**Interrupt enable —** When set, the LEM will assert an interrupt when finished processing	0 = Do not generate an interrupt when finished processing (Default) 1 = Generate an interrupt when finished processing
OBR Bit 7	**Output Bit Reverse —** When set, the data will be written to the data registers in a 'bit reversed' format. The data is stored with the MSB in bit location 0 of the data register and the LSB stored in bit location 15 of the data register.	0 = Do not enable 'output bit reverse' storage of output data (Default) 1 = Enable 'output bit reverse' storage of output data
IBR Bit 6	**Input Bit Reverse —** When set, this indicates to the LEM that the input data has been written in a 'bit reversed' fashion+.	0 = Input data was not written to the data registers in a 'bit reversed' fashion (Default) 1 = Input data was written to the data registers in a 'bit reversed' fashion
reset Bit 5	**Software reset —** Software can perform a reset of the entire LEM by writing the bit to 1. It is a self-clearing bit. **Note:** Software must allow two DSP clock cycles for the reset to occur properly.	0 = Do not perform a reset of the LEM block (Default) 1 = Reset the LEM block **Note:** Software must allow two DSP clock cycles for the reset to occur properly
enable Bit 4	**Enable processing —** Start processing of the LEM block. **Note:** This must be the last bit written in the programming sequence. See Section 13.5, "Programming Model," on page 13-20 for the correct programming sequence.	0 = Do not start processing data (Default) 1 = Start processing data
clk_en Bit 3	**LEM clock enable —** Enables or disables the master clock to the LEM. If this bit is not set, the LEM has no clock.	0 = Disable the master clock to the LEM (Default) 1 = Enable the master clock to the LEM

Table 13-22. CNTRL Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>mode[2:0]
Bits 2-0</td><td>Processing mode select — Determines what mode of operation the LEM is to use when processing the input data.</td><td>
<table>
<tr><th>mode[2:0]</th><th>Mode Select</th></tr>
<tr><td>$0</td><td>A5/1 block generation</td></tr>
<tr><td>$1</td><td>A5/2 block generation</td></tr>
<tr><td>$2</td><td>FIRE encode</td></tr>
<tr><td>$3</td><td>FIRE decode</td></tr>
<tr><td>$4 .. $7</td><td>RESERVED</td></tr>
</table>
</td></tr>
</table>

STAT LEM Status Register **Addr Y:$FFA1**

BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
													CE	UE	Done
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 13-23. STAT Description

Name	Description	Settings
Bits 15-3	Reserved	N/A
CE Bit 2	**Correctable Error —** Set when FIRE decode has detected a correctable error	0 = No correctable error detected (Default) 1 = A correctable error was detected
UE Bit 1	**Uncorrectable Error —** Set when FIRE decode has detected that uncorrectable errors exist in the data	0 = No uncorrectable errors detected (Default) 1 = Uncorrectable errors detected in input data
Done Bit 0	**Done processing —** The done bit is set whenever the LEM is not processing data. **Note:** After setting the enable bit in the CNTRL register, software must wait 2 DSP clock cycles before polling on the done bit.	0 = LEM is currently processing data. 1 = LEM is not processing data (Default)

DATA0 - DATA15 LEM Generic Data Registers **Addr Y:$FFA2-Y:$FFB1**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Data [15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-24. DATA Description

Name	Description	Settings
Data[15:0] Bits 15-0	**Data registers —** The data registers hold both the input and output data. The input data is used by the encryption engine, and the output data is the result of processing the input data. The reset value will be read from the data registers until the enable bit is set. When the LEM is enabled, the data registers cannot be written, and should not be read until the gem is done processing the data. **Note:** Reading the data registers while the LEM is processing is not supported and will result in erratic results.	N/A

Chapter 14
Base Band Port (BBP)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01	kamlesh kumar	Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
1.0	04/18/02	kamlesh kumar	
1.1	07/19/02	Shannon Osgood, Mark Babcock	Updated per DDTS# DSPhl14835.
1.2	05/07/03		Updated for LTE specification release.

14.1 BBP changes from RSI

The Base Band Port (BBP) is basically an RSI with an addition of a receive and transmit frame sync counters and special glue logic for the MAGIC serial communication protocol. For more details see Section 6.5, “MAGIC Serial Communication Protocol Enable on BBP pins.”

- Table 14-7. CRBB Description TCE Bit 4
- Table 14-7. CRBB Description RCE Bit 5
- Table 14-7. CRBB Description TCIE Bit 6
- Table 14-7. CRBB Description RCIE Bit 7
- Table 14-5. TCRB Description
- Table 14-4. RCRB Description
- Section 14.5.3, “BBP Exceptions,” : list 4 and 8.
- Section 14.5.4.5, “Counters Operating Modes.”
- Section 6.2.4, “Detailed Register Descriptions for DSP Core.”
- Section 6.5, “MAGIC Serial Communication Protocol Enable on BBP pins.”

14.2 Introduction

The Base Band Port (BBP) provides a full-duplex serial port for serial communication with a variety of serial devices including one or more industry-standard codecs, other DSPs, microprocessors, and peripherals which implement the Motorola SPI. The BBP consists of independent transmitter and receiver sections, receiver and transmitter counters and a common BBP clock generator.Following is the pin-out and some figures explaining the functioning of BBP.

Note: Kindly refer para 14.7 for neptune specific requirement.

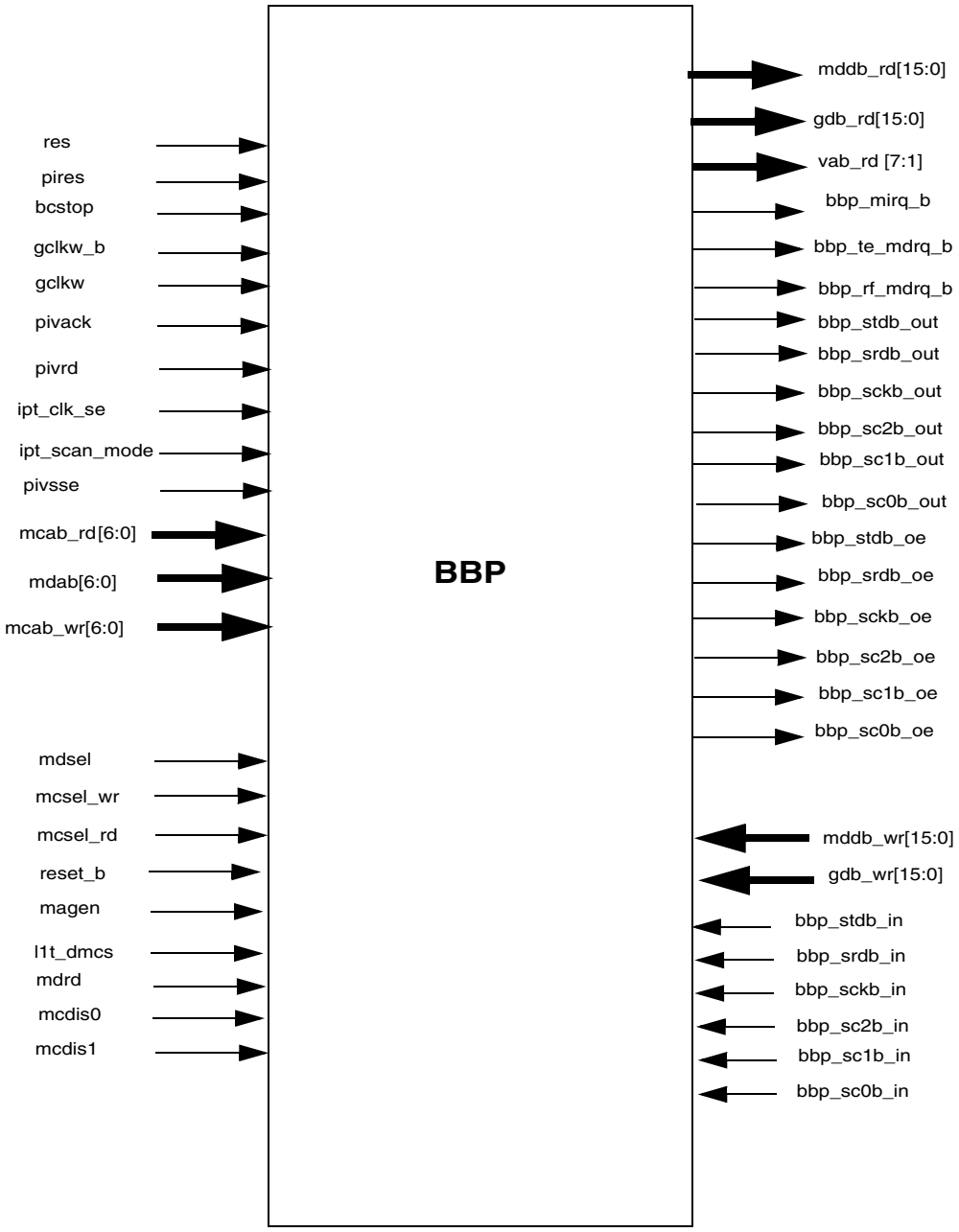


Figure 14-1. BBP Pin Out

14.2.1 BBP Module Pin List

Table 14-1 lists the pins in the BBP module.

Table 14-1. BBP Module Pin List

Pin Name	Direction	Description
reset_b	input	mcu reset
l1t_dmcs	input	L1 timer DMCS signal
magen	input	glue logic enable signal
ipt_clk_se	input	Scan enable signal
ipt_scan_mode	input	Scan Mode
mcab_rd[6:0]	input	Core Peripheral Read Address Bus
mcab_wr [6:0]	input	Core Peripheral Write Address Bus
mcdis0	input	Core Access T0 Disable
mcdis1	input	Core Access T1 Disable
mcsel_rd	input	Core IO Mapped Peripheral Read Address Bus Select
mcsel_wr	input	Core IO Mapped Peripheral Write Address Bus Select
gdb_rd[15:0]	output	Core Global Data Bus for read data
gdb_wr[15:0]	input	Core Global Data Bus for writedata
mdab[6:0]	input	DMA Peripheral Address Bus
mdrd	input	DMA Read/Write
mdsel	input	DMA IO Mapped Peripheral Address Bus Select
mddb_rd[15:0]	output	DMA Peripheral Data Buses for read data.
mddb_wr [15:0]	input	DMA Peripheral Data Buses for write data.
pivack	input	Peripheral Interrupt Acknowledge
pivrd	input	Interrupt Vector Read Strobe
pivsse	input	Interrupt Vector Source Select
gclkw	input	dsp clockClock
gclkw_b	input	inverted dsp clock
res	input	Peripheral hardware RESET
pires	input	Peripheral software RESET
bcstop	input	STOP RESET
bbp_rf_mdrq_b	output	DMA Peripheral Request (Receiver full)
bbp_te_mdrq_b	output	DMA Peripheral Request (Transmitter empty)
bbp_mirq_b	output	Peripheral Module Interrupt Request
vab_rd [7:1]	output	Vector address bus

Table 14-1. BBP Module Pin List (Continued)

Pin Name	Direction	Description
bbp_stdb_out	output	Serial transmit data out
bbp_srdb_out	output	Serial data receive out
bbp_sckb_out	output	bbp bit clock out
bbp_sc2b_out	output	sc2b port data out
bbp_sc1b_out	output	sc1b port data out
bbp_sc0b_out	output	sc0b port data out
bbp_stdb_in	input	stdb port input data
bbp_srdb_in	input	srdb output data
bbp_sckb_in	input	sckb port data
bbp_sc2b_in	input	sc2b port data
bbp_sc1b_in	input	sc1b port data
bbp_sc0b_in	input	sc0b port data
bbp_stdb_oe	output	stdb outpout enable
bbp_srdb_oe	output	srdb outpout enable
bbp_sckb_oe	output	sckb outpout enable
bbp_sc2b_oe	output	sc2b outpout enable
bbp_sc1b_oe	output	sc1b outpout enable
bbp_sc0b_oe	output	sc0b outpout enable

The Block diagram of BBP is given as follows.

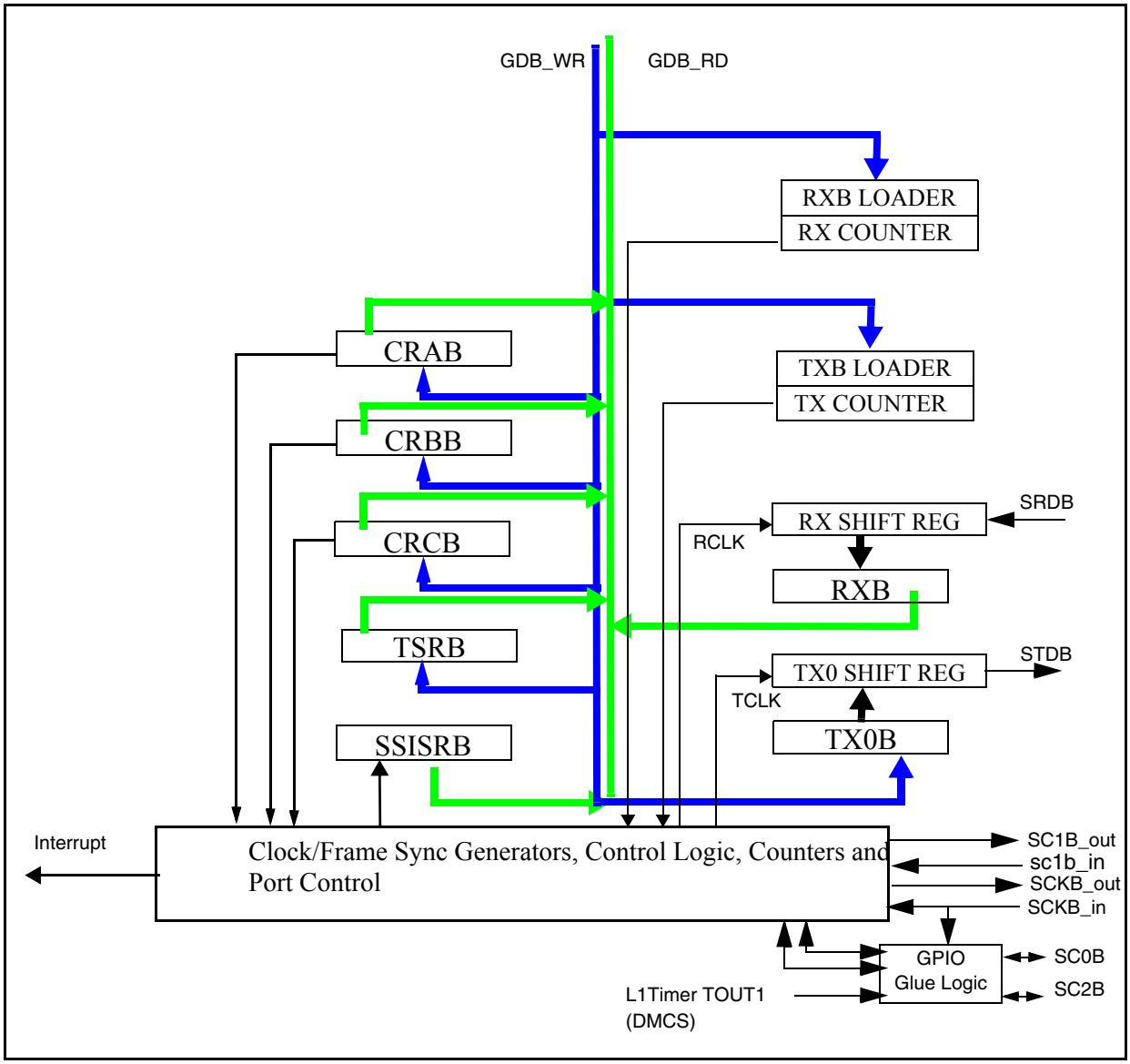


Figure 14-2. BBP Block Diagram

The Functional Block Diagram for BBP is as follows.

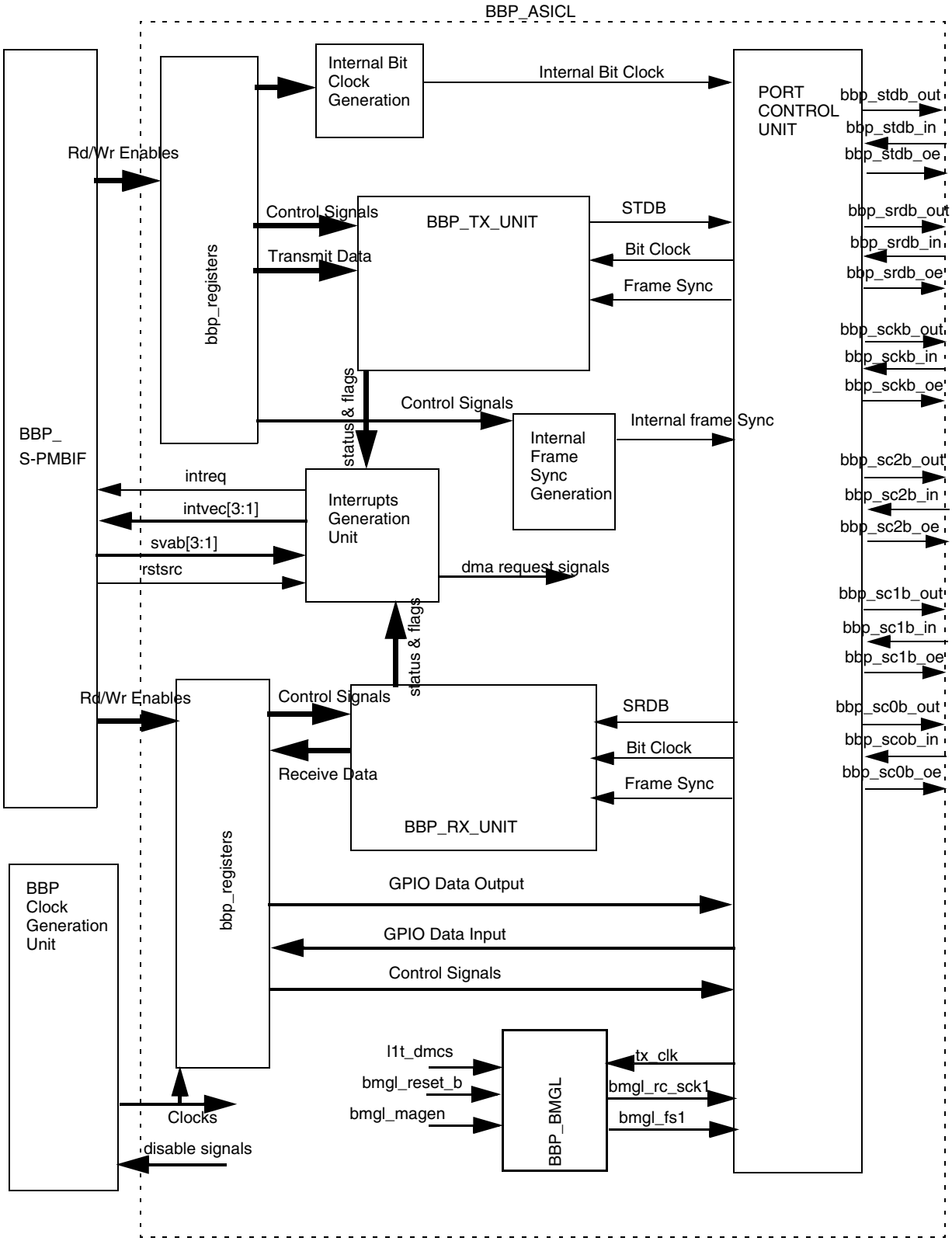

BBP_ASICL
Internal Bit Clock Generation
Internal Bit Clock
PORT CONTROL UNIT
bbp_stdb_out
bbp_stdb_in
bbp_stdb_oe
Rd/Wr Enables
Control Signals
BBP_TX_UNIT
STDB
bbp_srdb_out
bbp_srdb_in
bbp_srdb_oe
Transmit Data
Bit Clock
Frame Sync
bbp_registers
bbp_sckb_out
bbp_sckb_in
bbp_sckb_oe
status & flags
Control Signals
Internal frame Sync
BBP_ S-PMBIF
Internal Frame Sync Generation
bbp_sc2b_out
bbp_sc2b_in
bbp_sc2b_oe
intreq
intvec[3:1]
Interrupts Generation Unit
svab[3:1]
rstsrc
dma request signals
bbp_sc1b_out
bbp_sc1b_in
bbp_sc1b_oe
status & flags
Rd/Wr Enables
Control Signals
SRDB
bbp_sc0b_out
bbp_scob_in
bbp_sc0b_oe
Bit Clock
Receive Data
Frame Sync
BBP_RX_UNIT
BBP Clock Generation Unit
bbp_registers
GPIO Data Output
GPIO Data Input
Control Signals
l1t_dmcs
tx_clk
bmgl_reset_b
bmgl_rc_sck1
BBP_BMGL
bmgl_magen
bmgl_fs1
Clocks
disable signals

Figure 14-3.

14.3 BBP Data and Control Pins

14.3.1 Serial Transmit Data Pin (STDB)

STDB is used for transmitting data from TXB serial transmit shift register. When bbp_stdb_oe is high i.e.STDB is configured as output, data present in TXB shift register is being transmitted from bbp_stdb_out line. With an internally generated bit clock, the bbp_stdb_out pin will have last transmitted data and bbp_stdb_oe will be low after the last data bit has been transmitted for a full clock period, assuming another data word does not follow immediately. If a data word follows immediately, then bbp_stdb_oe will remain high and it will not go to low state. STDB may be programmed as a general-purpose pin when the BBP STDB function is not being used.

14.3.2 Serial Receive Data Pin (SRDB)

SRDB receives serial data and transfers the data to the BBP receive shift register. SRDB may be programmed as a general-purpose I/O pin when the BBP SRDB function is not being used.

14.3.3 Serial Clock (SCKB)

SCKB has one input only pin(bbp_sckb_in) and one output only pin(bbp_sckb_out) for providing the serial bit rate clock for the BBP interface. The SCKB is a clock input or output used by the transmitter and receiver in synchronous modes or by the transmitter only in asynchronous modes (see Table 14-2. BBP Clock Sources). If the SCKB pin configured as output pin, the Internal Bit Clock frequency will be:

$$f_{INT\_BIT\_CLK}=f_{DSP\_CLK} /[(2\text{-}PSR)^3 *2*(PM+1)]$$

In all cases, care must be taken to have always the Internal Bit Clock frequency lower than one third of the DSP_CLK frequency (see the note in Table 14-6, CRAB Description, PSR Bit 15).

The Frame Sync Frequency is based on the Internal Bit Clock frequency and will be:

$$f_{FRAME\_SYNC\_CLK}=f_{INT\_BIT\_CLK} / [WL*(DC+1)]$$

NOTE:

Setting DC=0 in network mode is a special case - on demand mode. There is no meaning for the FRAME_SYNC_CLK period in this mode, since the frame sync is not periodic.

SCKB may be programmed as a general-purpose I/O pin when the BBP SCKB function is not being used.

NOTE:

Although an external serial clock can be independent of and asynchronous to the DSP system clock, it must exceed the minimum clock cycle time of 6T (i.e., the system clock frequency must be at least three times the external BBP clock frequency). The BBP needs at least three DSP phases (DSP phase = T) inside each half of the serial clock.

14.3.4 Serial Control Pin (SC0B)

The function of this pin is determined by the selection of either synchronous or asynchronous mode (see Table 14-8. Mode and Pin Definition Table). For asynchronous mode, this pin will be used for the receive clock I/O. For synchronous mode, this pin is used for serial flag I/O. A typical application of flag I/O would be multiple device selection for addressing in codec systems. If this pin is configured as serial flag pin its direction is determined by the SCD0 bit in the CRCB If SCD0 is high then data will come on input line(bbp_sc0b_in) and when it is configured as output data will be send out from the outline (bbp_sc0b_out) . When configured as an output, this pin will be either serial output flag 0, based on control bit OF0 in CRBB, or a receive shift register clock output. When configured as an input, this pin may be used either as serial input flag 0, which will control status bit IF0 in the SSISRB, or as a receive shift register clock input.
SC0B may be programmed as a general-purpose I/O pin (P0) when the BBP SC0B function is not being used.

14.3.5 Serial Control Pin (SC1B)

The function of this pin is determined by the selection of either synchronous or asynchronous mode (see Table 14-8. Mode and Pin Definition Table). In asynchronous mode (such as a single codec with asynchronous transmit and receive), this pin is the receiver frame sync I/O. For synchronous mode, this pin is used for serial flag I/O and operates like the previously described SC0B. SC0B and SC1B are independent serial I/O flags but may be used together for multiple serial device selection. SC0B and SC1B can be used unencoded to select up to two codecs or may be decoded externally to select up to four codecs. If this pin is configured as serial flag pin its direction is determined by the SCD1 bit in the CRCB. When configured as an output, this pin will be either a serial output flag, based on control bit OF1, or it will make the receive frame sync signal available. When configured as an input, this pin may be used as a serial input flag, which will control status bit IF1 in the BBP status register, or as a receive frame sync from an external source.
SC1B may be programmed as a general-purpose I/O pin (P1) when the BBP SC1B function is not being used.

Table 14-2. BBP Clock Sources

SYN	SCKD	SCD0	R Clock Source	RXB Clock Out	T Clock Source	TXB Clock Out
Asynchronous						
0	0	0	EXT, SC0B	–	EXT, SCKB	–
0	0	1	INT	SC0B	EXT, SCKB	–
0	1	0	EXT, SC0B	–	INT	SCKB
0	1	1	INT	SC0B	INT	SCKB
Synchronous						
1	0	d.c.	EXT, SCKB	–	EXT, SCKB	–
1	1	d.c.	INT	SCKB	INT	SCKB

14.3.6 Serial Control Pin (SC2B)

This pin is used for frame sync I/O. SC2B is the frame sync for both the transmitter and receiver in synchronous mode and for the transmitter only in asynchronous mode (see Table 14-8. Mode and Pin Definition Table). The direction of this pin is determined by the SCD2 bit in CRCB. When configured as an output, this pin is the internally generated frame sync signal. When configured as an input, this pin receives an external frame sync signal for the transmitter (and the receiver in synchronous operation). SC2B may be programmed as a general-purpose I/O pin (P2) when the BBP SC2B function is not being used.

14.4 BBP PROGRAMMING MODEL

The BBP can be viewed as three control registers, one status register, a transmit data register, a receive data register, special-purpose time slot register, two frame sync counters (receive and transmit) and three port registers, control, direction and data. The following paragraphs give detailed descriptions and operations of each of the bits in the BBP registers.

14.4.1 Register Summary

The Register Summary lists all registers of the module by ascending address. The absolute address of each register is given, and the value of each bit for reads and writes is given using the conventions in the Legend.

Note that the DSP register address indicates if the register is in X or Y memory

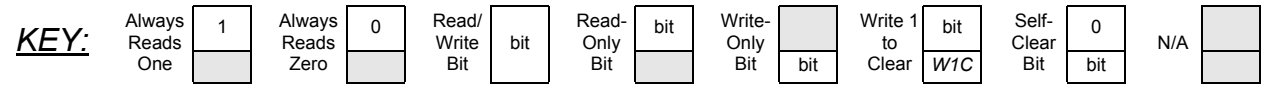


Table 14-3. BBP Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCRB (X:$FFA4)	R	RLV[15:0]															
	W																
TCRB (X:$FFA5)	R	TLV[15:0]															
	W																
CRAB (X:$FFA6)	R	PSR	WL[1:0]		DC[4:0]					PM[7:0]							
	W																
CRBB (X:$FFA7)	R	REIE	TEIE	RLIE	TLIE	RIE	TIE	RE	TE	RCIE	TCIE	RCE	TCE	0	0	OF[1:0]	
	W																
CRCB (X:$FFA8)	R	FSP	FSR	FSL[1:0]		0	0	0	0	SHFD	CKP	SCKD	SCD[2:0]			MOD	SYN
	W																
SSISRB (X:$FFA9)	R	0	0	0	0	0	0	0	0	RDF	TDE	ROE	TUE	RFS	TFS	IF[1:0]	
	W																
RXB (X:$FFAA)	R	RXB[15:0]															
	W																
TSRB (X:$FFAB)	R																
	W	TSRB[15:0]															
TXB (X:$FFAC)	R																
	W	TXB[15:0]															
PDRB (X:$FFAD)	R	0	0	0	0	0	0	0	0	0	0	PD[5:0]					
	W																

Table 14-3. BBP Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRRB (X:$FFAE)	R	0	0	0	0	0	0	0	0	0	0	PDC[5:0]					
	W																
PCRB (X:$FFAF)	R	0	0	0	0	0	0	0	0	PEN	0	PC[5:0]					
	W																

14.4.2 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various BBP registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

RCRB BBP Receive Counter Load Register **Addr X:$FFA4**

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
	RLV[15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-4. RCRB Description

Name	Description	Settings
RLV[15-0] Bits15-0	**Receive Load Value** — RCRB is a 16 bit read - write register. When the counter is enabled by setting the RCE bit, the receive counter is loaded with the RCRB data (RLV0-15 bits) and counting down starts. Upon a roll over of the counter, it is reloaded with RCRB data and count down continues. All RCRB bits are cleared by hardware and software reset.	N/A

TCRB BBP Transmit Counter Load Register **Addr X:$FFA5**

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
	TLV[15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-5. TCRB Description

Name	Description	Settings
TLV[15-0] Bits15-0	**Transmit Load Value** — TCRB is a 16 bit read - write register. When the counter is enabled by setting the TCE bit, the transmit counter is loaded with the TCRB data (TLV0-15 bits) and counting down starts. Upon a roll over of the counter, it is reloaded with TCRB data and count down continues. All TCRB bits are cleared by hardware and software reset.	N/A

CRAB is one of three 16-bit read/write control registers used to direct the operation of the BBP. The CRAB controls the BBP clock generator bit and frame sync rates, word length, and number of words per frame for the serial data. The CRAB control bits are described in the following paragraphs (see Table 14-6)

CRAB BBP Control Register A **Addr X:$FFA6**

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
	PSR	WL[1:0]		DC[4:0]					PM[7:0]							
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-6. CRAB Description

Name	Description	Settings
PSR Bit 15	**Prescaler Range** —The PSR controls a fixed divide-by-eight prescaler in series with the variable prescaler. This bit is used to extend the range of the prescaler for those cases where a slower bit clock is desired. When PSR is set, the fixed prescaler is bypassed. When PSR is cleared, the fixed divide-by-eight prescaler is operational. The maximum internally generated bit clock frequency is BRGCLK/4 ((BRGCLK is the input clock to the baud rate generator, i.e. DSP_CLK), the minimum internally generated bit clock frequency is BRGCLK/2/8/256=BRGCLK/4096. Hardware and software reset clear PSR. **Note:** The combination PSR = 1 and PM7-PM0 = $00 is forbidden and may cause synchronization problems if used.	0 = Prescaler enabled (default). 1 = Prescaler disabled.
WL[1:0] Bits 14–13	**Word Length Control** —The WL1 and WL0 bits are used to select the length of the data words being transferred via the BBP. Word lengths of 8, 12 or 16 bits may be selected according to the assignment described in the settings column. Hardware and software reset clear WL1 and WL0	**WL1** / **WL0** / **Number of Bits/Word** 0 / 0 / 8 (default) 0 / 1 / 12 1 / 0 / 16 1 / 1 / Reserved

Table 14-6. CRAB Description

Name	Description	Settings
DC[4:0] Bits 12–8	The DC4–DC0 bits control the divide ratio for the programmable frame rate dividers used to generate the frame clocks. In network mode, this ratio may be interpreted as the number of words per frame minus one. In normal mode, this ratio determines the word transfer rate. The divide ratio may range from 1 to 32 (DC=00000 to 11111) for normal mode and 2 to 32 (DC=00001 to 11111) for network mode. A divide ratio of one (DC=00000) in network mode is a special case (on demand mode). In normal mode, a divide ratio of one (DC=00000) provides continuous periodic data word transfers. A bit-length sync must be used in this case. Hardware and software reset clear DC4–DC0.	N/A
PM[7:0] Bits 7–0	The PM7–PM0 bits specify the divide ratio of the prescale divider in the BBP clock generator. A divide ratio from 1 to 256 (PM=0 to $FF) may be selected. The bit clock output is available at the transmit clock (SCKB) and/or the receive clock (SC0B) pins of the DSP. The bit clock output is also available internally for use as the bit clock to shift the transmit and receive shift registers. Careful choice of the crystal oscillator frequency and the prescaler modulus will allow the industry-standard codec master clock frequencies of 2.048 MHz, 1.544 MHz, and 1.536 MHz to be generated. Hardware and software reset clear PM0–PM7.	N/A

The CRBB is one of three 16-bit read/write control registers used to direct the operation of the BBP (see Table 14-7). CRBB controls the serial output flag, the BBP interrupts enables and transmitter and receiver enable. The BBP CRBB bits are described in the following paragraphs. Hardware and software reset clear all the bits in the CRBB.

CRBB BBP Control Register B **Addr X:$FFA7**

	Bit15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
	REIE	TEIE	RLIE	TLIE	RIE	TIE	RE	TE	RCIE	TCIE	RCE	TCE			OF1	OF0
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-7. CRBB Description

Name	Description	Settings
REIE Bit 15	**Receive Exception Interrupt Enable**-- When REIE is set, the DSP will be interrupted when both RDF and ROE in the BBP status register are set. When REIE is cleared, this interrupt is disabled. Reading the status register followed by reading the receive data register will clear ROE, thus clearing the pending interrupt. Hardware and software reset clear REIE.	0 = Interrupt disabled (default). 1 = Interrupt enabled.
TEIE Bit 14	**Transmit Exception Interrupt Enable**—When TEIE is set, the DSP will be interrupted when both TDE and TUE in the BBP status register are set. When TEIE is cleared, this interrupt is disabled. Reading the status register followed by writing to the transmitter data registers will clear TUE, thus clearing the pending interrupt. Hardware and software reset clear TEIE.	0 = Interrupt disabled (default). 1 = Interrupt enabled.
RLIE Bit 13	**Receive Last Slot Interrupt Enable—**RLIE enables an interrupt after the last slot of a frame ended in network mode only. When RLIE is set the DSP will be interrupted after the last slot in a frame ended. When RLIE is cleared the receive last slot interrupt is disabled. Hardware and software reset clear RLIE. RLIE is disabled when DC = $0 (on demand mode). The use of the Receive last slot interrupt is described in Section 14.5.3, “BBP Exceptions.” .	0 = Interrupt disabled (default). 1 = Interrupt enabled.

Table 14-7. CRBB Description (Continued)

Name	Description	Settings
TLIE Bit 12	**Transmit Last Slot Interrupt Enable—**TLIE enables an interrupt at the beginning of last slot of a frame in network mode. When TLIE is set the DSP will be interrupted at the start of the last slot in a frame in network mode. When TLIE is cleared the transmit last slot interrupt is disabled. Hardware and software reset clear TLIE. TLIE is disabled when DC= $0 (on demand mode). The use of the transmit last slot interrupt is described in paragraph 14.5.3 BBP Exceptions.	0 = Interrupt disabled (default). 1 = Interrupt enabled.
RIE Bit 11	**Receive Interrupt Enable—**When RIE is set, the DSP will be interrupted when RDF in the BBP status register is set. When RIE is cleared, this interrupt is disabled. Reading the receive data register will clear RDF, thus clearing the pending interrupt. Receive interrupts with exception have higher priority than normal receive data interrupts, therefore if exception occurs (ROE is set) and REIE is set the BBP will request an BBP receive data with exception interrupt from the interrupt controller. Hardware and software reset clear RIE.	0 = Interrupt disabled (default). 1 = Interrupt enabled.
TIE Bit 10	**Transmit Interrupt Enable—**The DSP will be interrupted when TIE and the TDE flag in the BBP status register are set. When TIE is cleared, this interrupt is disabled. Writing data to the data register of the transmitter or to TSR will clear TDE, thus clearing the interrupt. Transmit interrupts with exception have higher priority than normal transmit data interrupts, therefore if exception occurs (TUE is set) and TEIE is set the BBP will request an BBP transmit data with exception interrupt from the interrupt controller. Hardware and software reset clear TIE.	0 = Interrupt disabled (default). 1 = Interrupt enabled.
RE Bit 9	**Receive Enable**—When RE is set, the receive portion of the BBP is enabled. When this bit is cleared, the receiver will be disabled by inhibiting data transfer into RXB. If data is being received while this bit is cleared, the remainder of the word will be shifted in and transferred to the BBP receive data register. RE must be set in the normal mode and on-demand mode to receive data. In network mode, the operation of clearing RE and setting it again will disable the receiver after reception of the current data word until the beginning of the next data frame. Hardware and software reset clear RE. **Note:** RE does not affect the generation of a frame sync.	0 = Receiver disabled (default). 1 = Receiver enabled.

Table 14-7. CRBB Description (Continued)

Name	Description	Settings
TE Bit 8	**Transmit Enable—**TE enables the transfer of data from TXB to the transmit shift register. When TE is set and a frame sync is detected, the transmit portion of the BBP is enabled for that frame. When TE is cleared, the transmitter will be disabled after completing transmission of data currently in the BBP transmit shift register. The STDB output is three-stated, and any data present in TXB will not be transmitted (i.e. data can be written to TXB with TE cleared; but data will not be transferred to the transmit shift register). The normal mode transmit enable sequence is to write data to the transmit data registers (or TSR) before setting TE. The normal transmit disable sequence is to clear TE, TIE and TEIE after TDE equals one. In the network mode, the operation of clearing TE and setting it again will disable the transmitter after completing transmission of the current data word until the beginning of the next frame. During that time period, the STDB pin will remain in the high-impedance state. Hardware reset and software reset clear TE. The on-demand mode transmit enable sequence can be the same as the normal mode, or TE can be left enabled. TE does not affect the generation of frame sync or output flags **Note:** The TE bit does not affect the generation of frame sync or output flags.	0 = Transmitter disabled (default). 1 = Transmitter enabled.
RCIE Bit 7	**Receive Counter Interrupt Enable**— When RCIE is set, a DSP interrupt will occur when the receive counter rolls over. When RCIE is cleared the interrupt is disabled. Hardware and software reset clear RCIE. The receive counter interrupt is described in Section 14.5.3, “BBP Exceptions.” : item 4.	0 = Interrupt disabled (default). 1 = Interrupt enabled.
TCIE Bit 6	**Transmit Counter Interrupt Enable—** When TCIE is set, a DSP interrupt will occur when the transmit counter rolls over. When TCIE is cleared the interrupt is disabled. Hardware and software reset clear TCIE. The transmit counter interrupt is described in Section 14.5.3, “BBP Exceptions.” : item 8.	0 = Interrupt disabled (default). 1 = Interrupt enabled.
RCE Bit 5	**Receive Counter Enable**—RCE enables the 16 bits receiver frame sync counter. When RCE bit is set, the counter is loaded with the RCRB data and starts counting down. When the RCE bit is cleared the receive counter is disabled. Hardware and software reset clear RCE bit.	0 = Counter disabled (default). 1 = Counter enabled.

Table 14-7. CRBB Description (Continued)

Name	Description	Settings
TCE Bit 4	**Transmit Counter Enable** — TCE enables the 16 bits transmitter frame sync counter. When TCE bit is set, the counter is loaded with the TCRB data and starts counting down. When the TCE bit is cleared the transmit counter is disabled. Hardware and software reset clear TCE bit.	0 = Counter disabled (default). 1 = Counter enabled.
Reserved Bits Bits 2:3	**Reserved Bits** — Bits 2-3 in the CRBB are reserved bits for future compatibility. They read as zero and must be written with zero.	N/A
OF1 Bit 1	**Serial Output Flag 1**— When the BBP is in the synchronous clock mode SC1B pin is configured as BBP flag 1. The serial control direction one bit (SCD1) when set, indicates that the SC1B pin is an output, then data present in OF1 will be written to SC1B at the beginning of the frame in normal mode or at the beginning of the next time slot in network mode. Hardware and software reset clear OF1. The normal sequence for setting output flags when transmitting data is: wait for TDE (TXB empty) to be set, first write the flags, and then write the transmit data to the TXB register. OF0 and OF1 are double buffered so that the flag states appear on the pins when the TXB data is transferred to the transmit shift register (i.e. the flags are synchronous with the data). **Note:** The optional serial output pins timing (SC0B and SC1B) are controlled by the frame timing and are not affected by TE or RE.	N/A
OF0 Bit 0	**Serial Output Flag 0**—When the BBP is in the synchronous clock mode SC0B pin is configured as BBP flag 0. The serial control direction zero bit (SCD0) when set, indicates that the SC0B pin is an output, then data present in OF0 will be written to SC0B at the beginning of the frame in normal mode or at the beginning of the next time slot in network mode. Hardware and software reset clear OF0.	N/A

Table 14-8. Mode and Pin Definition Table

Control Bits			BBP PINS					
SYN	TE	RE	SC0B	SC1B	SC2B	SCKB	STDB	SRDB
0	0	0	U	U	U	U	U	U
0	0	1	RXC	FSR	U	U	U	RD
0	1	0	U	U	FST	TXC	TD	U
0	1	1	RXC	FSR	FST	TXC	TD	RD
1	0	0	F0/U	F1/U	FS	XC	U	U
1	0	1	F0/U	F1/U	FS	XC	U	RD
1	1	0	F0/U	F1/U	FS	XC	TD	U
1	1	1	F0/U	F1/U	FS	XC	TD	RD

- TXC - Transmitter Clock
- RXC - Receiver Clock
- XC - Transmitter/Receiver Clock (Synchronous Operation)
- FST - Transmitter Frame Sync
- FSR - Receiver Frame Sync
- FS - Transmitter/Receiver Frame Sync (Synchronous Operation)
- TD - Transmit Data
- RD - Receive Data
- F0/U - Flag 0 / Unused
- F1/U - Flag 1 / Unused
- U - Unused (may be used as GPIO pin)

NOTE:

A pin may be used as General Purpose IO pins (GPIO) if its corresponding bit in the Port Control Register (PCRB, bits PC5-PC0) is zero. See Table 14-16, PCRB Description for more details.

The CRCB is one of three 16-bit read/write control registers used to direct the operation of the BBP (see Table 14-9). CRCB controls the BBP multifunction pins, SC2B, SC1B, and SC0B, which can be used as clock inputs or outputs, frame synchronization pins or serial I/O flag pins. The direction control bits for the serial control pins are in the BBP CRCB. Operating modes are also selected in this register. Hardware and software reset clear all the bits in the CRCB. The relationships between the BBP pins and some of the CRCB bits are summarized in Table 14-8. Mode and Pin Definition Table. The BBP CRCB bits are described in the following paragraphs.

CRCB BBP Control Register C **Addr X:$FFA8**

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
	FSP	FSR	FSL[1:0]						SHFD	CKP	SCKD	SCD2	SCD1	SCD0	MOD	SYN
TYPE	rw	rw	rw	rw	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-9. CRCB Description

Name	Description	Settings
FSP Bit 15	**Frame Sync Polarity—**FSP determines the polarity of the receive and transmit frame sync signals. When FSP is cleared the frame sync signal polarity is positive (i.e. the frame start is signed by the high level of the frame sync pin). When FSP is set the frame sync signal polarity is negative (i.e. the frame start is signed by the low level of the frame sync pin). Hardware reset and software reset clear FSP.	0 = Frame Sync Polarity Active-high (default). 1 = Frame Sync Polarity Active-low.
FSR Bit 14	**Frame Sync Relative Timing—**FSR determines the relative timing of the receive and transmit frame sync signal as referred to the serial data lines, for a word length frame sync only. When FSR is cleared the word length frame sync occurs together with the first bit of the data word of the first slot. When FSR is set the word length frame sync occurs one serial clock cycle earlier (i.e. together with the last bit of the previous data word). Hardware reset and software reset clear FSR.	0 = Frame Sync occurs with First bit of current frame (default). 1 = Frame Sync occurs with Last bit of previous frame.

Table 14-9. CRCB Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>FSL[1:0]
Bits
13–12</td><td>Frame Sync Length—These bits select the length of frame sync to be generated or recognized. If FSL1 equals to zero and FSL0 equals to zero, a word-length frame sync is selected for both TXB and RXB that is the length of the data word defined by bits WL1 and WL0. If FSL1 equals to one and FSL0 equals to zero, a 1-bit clock period frame sync is selected for both TXB and RXB. When FSL0 equals to one, the TXB and RXB frame syncs are different lengths. The encoding of FSL1 and FSL0 is described in the following table.
Hardware reset and software reset clear FSL0 and FSL1</td><td><table><tr><th>FSL 1</th><th>FSL0</th><th>Frame Sync Length</th></tr><tr><td>0</td><td>0</td><td>Word length Frame Sync for both TXB/RXB</td></tr><tr><td>0</td><td>1</td><td>Bit length Frame Sync for TXB and word length Frame Sync for RXB</td></tr><tr><td>1</td><td>0</td><td>Bit length Frame Sync for both TXB/RXB</td></tr><tr><td>1</td><td>1</td><td>Bit length Frame Sync for RXB and word length frame sync for TXB</td></tr></table></td></tr>
<tr><td>Bits
8 -11</td><td>Reserved Bits—Bits 8-11 in the CRCB are reserved bits they read as zeros and must be written with zero for future compatibility.</td><td>N/A</td></tr>
<tr><td>SHFD
Bit 7</td><td>Shift Direction—This bit causes the transmit shift register to shift data out MSB first when SHFD equals zero or LSB first when SHFD equals one. Received data is shifted in MSB first when SHFD equals zero or LSB first when SHFD equals one. Hardware reset and software reset clear SHFD.</td><td>0 = Shift Data MSB first (default).
1 = Shift Data LSB first.</td></tr>
<tr><td>CKP
Bit 6</td><td>Clock Polarity—The clock polarity bit controls on which bit clock edge data and frame sync are clocked out and latched in. If CKP is cleared the data and the frame sync are clocked out on the rising edge of the transmit bit clock and latched in on the falling edge of the receive bit clock. If CKP is set the falling edge of the transmit clock is used to clock the data out and frame sync and the rising edge of the receive clock is used to latch the data and frame sync in. Hardware reset and software reset clear CKP.</td><td>0 = Data and frame sync clocked out on the rising edge of the transmit bit clock and latched in on the falling edge of the receive bit clock
1 = Data and frame sync clocked out on the falling edge of the transmit bit clock and latched in on the rising edge of the receive bit clock</td></tr>
<tr><td>SCKD
Bit 5</td><td>Clock Source Direction—SCKD selects the source of the clock signal used to clock the transmit shift register in the asynchronous mode and the transmit shift register and the receive shift register in the synchronous mode. In asynchronous mode when SCKD is set, the internal clock source becomes the bit clock for the transmit shift register and word length divider and is the output on the SCKB pin. When SCKD is cleared, the clock source is external; the internal clock generator is disconnected from the SCKB pin, and an external clock source may drive this pin. Hardware and software reset clear SCKD.</td><td>0 = External Clock (default).
1 = Internal Clock</td></tr>
</table>

Table 14-9. CRCB Description (Continued)

Name	Description	Settings
SCD2 Bit 4	**Serial Control 2 Direction**— SCD2 controls the direction of the SC2B I/O pin. When SCD2 is cleared, SC2B is an input; when SCD2 is set, SC2B is an output. Hardware and software reset clear SCD2.	0 = SC2B pin is an Input (default). 1 = SC2B pin is an Output.
SCD1 **Bit 3**	**Serial Control 1 Direction**—SCD1 controls the direction of the SC1B I/O pin. When SCD1 is cleared, SC1B is an input; when SCD1 is set, SC1B is an output. Hardware and software reset clear SCD1.	0 = SC1B pin is an Input (default). 1 = SC1B pin is an Output.
SCD0 **Bit 2**	**Serial Control 0 Direction** — SCD0 controls the direction of the SC0B I/O pin. When SCD0 is cleared, SC0B is an input; when SCD0 is set, SC0B is an output. Hardware and software reset clear SCD0.	0 = SC0B pin is an Input (default). 1 = SC0B pin is an Output.
MOD **Bit 1**	**Mode Select** — MOD selects the operational mode of the BBP. When MOD is cleared, the normal mode is selected; when MOD is set, the network mode is selected. In the normal mode, the frame rate divider determines the word transfer rate – one word is transferred per frame sync during the frame sync time slot. In network mode, a word is (possibly) transferred every time slot. Hardware and software reset clear MOD.	0 = Normal mode (default). 1 = Network mode.
SYN Bit 0	**Synchronous/Asynchronous —** SYN controls whether the receive and transmit functions of the BBP occur synchronously or asynchronously with respect to each other. When SYN is set, synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals. When SYN is cleared, asynchronous mode is chosen and separate clock and frame sync signals are used for the transmit and receive sections. Hardware reset and software reset clear SYN.	0 = Asynchronous mode (default). 1 = Synchronous mode.

The SSISRB (see Table 14-10) is an 8-bit read-only status register used by the DSP to read the status and serial input flags of the BBP. When the SSISRB is read to the internal data bus, the register contents occupy the low-order byte of the data bus, and the remaining bits are read as zeros. The status bits are described in the following paragraphs.

SSISRB BBP Status Register **Addr X:$FFA9**

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
									RDF	TDE	ROE	TUE	RFS	TFS	IF1	IF0
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Table 14-10. SSISRB Description

Name	Description	Settings
RDF Bit 7	**Receive Data Register Full** — RDF is set when the contents of the receive shift register are transferred to the receive data register. RDF is cleared when the DSP reads the receive data register or cleared by hardware, software, BBP individual, or STOP reset. If RIE is set, a DSP receive data interrupt request will be issued when RDF is set.	0 = Receive Data Register not full (default). 1 = Receive Data Register Full
TDE Bit 6	**Transmit Data Register Empty** — This flag is set when the contents of the transmit data register is transferred to the transmit shift register; it is also set for a TSR disabled time slot period in network mode (as if data were being transmitted after the TSR was written). When set, TDE indicates that data should be written to the TXB register or to the time slot register (TSR). TDE is cleared when the DSP writes to the transmit data register, or when the DSP writes to the TSR to disable transmission of the next time slot. If TIE is set, a DSP transmit data interrupt request will be issued when TDE is set. Hardware, software, BBP individual, and STOP reset set TDE.	0 = Transmit Data Register not empty 1 = Transmit Data Register Empty (default).
ROE Bit 5	**Receiver Overrun Error Flag** — This flag is set when the serial receive shift register is filled and ready to transfer to the receiver data register (RXB) and RXB is already full (i.e., RDF=1). If REIE is set, a DSP receiver overrun error interrupt request will be issued when ROE is set. Hardware, software, BBP individual, and STOP reset clear ROE. ROE is also cleared by reading the SSISRB with ROE set, followed by reading the RXB.	0 = No receive error (default). 1 = Receiver overrun error has occurred.

Table 14-10. SSISRB Description (Continued)

Name	Description	Settings
TUE Bit 4	**Transmitter Underrun Error Flag** — TUE is set when the transmit shift registers is empty (no new data to be transmitted) and a transmit time slot occurs. When a transmit underrun error occurs, the previous data (which is still present in the TXB register that was not written) will be retransmitted. In the normal mode, there is only one transmit time slot per frame. In the network mode, there can be up to 32 transmit time slots per frame. If TEIE is set, a DSP transmit underrun error interrupt request will be issued when TUE is set. Hardware, software, BBP individual, and STOP reset clear TUE. TUE is also cleared by reading the SSISRB with TUE set, followed by writing to the transmit data registers or to TSR.	0 = No transmit error (default). 1 = Transmitter underrun error has occurred.
RFS Bit 3	**Receive Frame Sync Flag** — When set, RFS indicates that a receive frame sync occurred during reception of the word in the serial receive data register. This indicates that the data word is from the first time slot in the frame. When RFS is clear and a word is received, it indicates (only in the network mode) that the frame sync did not occur during reception of that word. RFS is cleared by hardware, software, BBP individual, or STOP reset. RFS is valid only if the receiver is enabled (RE=1). **Note:** In normal mode, RFS will always read as a one when reading data because there is only one time slot per frame – the “frame sync” time slot.	0 = Receive Frame Sync did not occur (default). 1 = Receive Frame Sync occurred.
TFS Bit 2	**Transmit Frame Sync Flag** — When set, TFS indicates that a transmit frame sync occurred in the current time slot. TFS is set at the start of the first time slot in the frame and cleared during all other time slots. Data written to a transmit data register during the time slot when TFS is set will be transmitted (in network mode), if the transmitter is enabled, during the second time slot in the frame. TFS is useful in network mode to identify the start of a frame. TFS is cleared by hardware, software, BBP individual, or STOP reset. TFS is valid only if the transmitter is enabled (TE equal 1). **Note:** In normal mode, TFS will always be read as a one when transmitting data because there is only one time slot per frame – the “frame sync” time slot.	0 = Transmit Frame Sync did not occur (default). 1 = Transmit Frame Sync occurred.

Table 14-10. SSISRB Description (Continued)

Name	Description	Settings
IF1 Bit 1	**Serial Input Flag 1**— The BBP latches data present on the SC1B pin during reception of the first received bit after frame sync is detected. IF1 bit is updated with this data when the receiver shift register is transferred into the receive data register. The IF1 bit is enabled only when SC1B is programmed as BBP in the Port Control Register, SYN is set, and SCD1 is cleared, indicating that SC1B is an input flag and the synchronous mode is selected; otherwise, IF1 reads as a zero when it is not enabled. Hardware, software, BBP individual, and STOP reset clear IF1.	N/A
IFO Bit 0	**Serial Input Flag 0** — The BBP latches data present on the SC0B pin during reception of the first received bit after frame sync is detected. IF0 bit is updated with this data when the receive shift register is transferred into the receive data register. The IF0 bit is enabled only when SC0B is programmed as BBP in the Port Control Register, SYN is set, and SCD0 is cleared, indicating that SC0B is an input flag and the synchronous mode is selected; otherwise, IF0 reads as a zero when it is not enabled. Hardware, software, BBP individual, and STOP reset clear IF0.	N/A

RXB BBP Receive Data Register **Addr X:$FFAA**

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
	RXB[15:0]															
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 14-11. RXB Description

Name	Description	Settings
RXB[15:0] Bits15-0	**BBP Receive Data** — RXB is a 16-bit read-only register that accepts data from the receive shift register as it becomes full. The data read will occupy the most significant portion of the receive data register. The unused bits (least significant portion) will read as zeros. The DSP is interrupted whenever RXB becomes full if the associated interrupt is enabled.	N/A

TSRB BBP Time Slot Register **Addr X:$FFAB**

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
	TSRB[15:0]															
TYPE:	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
RESET:	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 14-12. TSRB Description

Name	Description	Settings
TSRB[15:0] Bits15-0	**BBP Time Slot Data** — TSR is effectively a null data register that is used when the data is not to be transmitted in the available transmit time slot. For the purposes of timing, TSR is a write-only register that behaves like an alternative transmit data register, except that, rather than transmitting data, the transmit data pin is in the high-impedance state for that time slot.	N/A

TXB BBP Transmit Data Register **Addr X:$FFAC**

	BIT15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	TXB[15:0]															
TYPE:	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
RESET:	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 14-13. TXB Description

Name	Description	Settings
TXB[15:0] Bits15-0	**BBP Transmit Data** — TXB is 16-bit write-only register. Data to be transmitted is written into this register and is automatically transferred to the transmit shift registers. The data written (8, 12 or 16 bits) should occupy the most significant portion of the TXB. The unused bits (least significant portion) of the TXB are don't care bits. The DSP is interrupted whenever TXB becomes empty if the transmit data register empty interrupt has been enabled.	N/A

The read/write 16 bit Port Data Register is used to read or write data to/from BBP GPIO pins.

PDRB BBP Port Data Register **Addr X:$FFAD**

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
											PD5	PD4	PD3	PD2	PD1	PD0
TYPE:	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-14. PDRB Description

Name	Description	Settings
PD [5:0] Bits 5-0	**Port Data Register bits** — PD(5:0) are used to read or write data from/to the corresponding port pins if they are configured as GPIO (by PC(5:0) bits in PCRB). If a port pin [i] is configured as a GPIO input, then the corresponding PD[i] bit will reflect the value present on this pin. If a port pin [i] is configured as a GPIO output, then the value written into the corresponding PD[i] bit will be reflected on the this pin. Hardware and software reset clear all PDRB bits. The mapping between the PDRB bits and the actual BBP ports is: PD[5] - STDB PD[4] - SRDB PD[3] - SCKB PD[2] - SC2B PD[1] - SC1B PD[0] - SC0B	N/A

The read/write 16 bit Port Direction Register controls the direction of BBP GPIO pins.

PRRB BBP Port Direction Register **Addr X:$FFAE**

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
											PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
TYPE:	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-15. PDRB Description

Name	Description	Settings
PDC Bits [5:0]	**Port Direction Register bits** — When port pin[i] is configured as GPIO, PDC[i] controls the port pin direction. When PDC[i] is set, the GPIO port pin[i] is configured as output. When PDC[i] is cleared the GPIO port pin[i] is configured as input. Hardware and software reset clear all PRRB bits. The following table describe the port pin configurations. The mapping between the PRRB bits and the actual BBP ports is: PDC[5] - STDB PDC[4] - SRDB PDC[3] - SCKB PDC[2] - SC2B PDC[1] - SC1B PDC[0] - SC0B	

PC[i]	PDC[i]	Port Pin[i] Function
1	X	BBP
0	0	GPIO input
0	1	GPIO output

The read/write 16 bit Port Control Register controls the functionality of BBP GPIO pins.

PCRB BBP Port Control Register **Addr X:$FFAF**

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
5									PEN		PC5	PC4	PC3	PC2	PC1	BPC0
TYPE:	r	r	r	r	r	r	r	r	rw	r	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-16. PCRB Description

Name	Description	Settings
PEN Bit 7	**Port Enable bit**— is the port enable bit, when cleared all BBP pins will be three-stated, ignoring any other settings. When set the pins will be active as defined by all other settings. Hardware and software reset clear al PC [i] bits and the PEN bit.	0 = BBP Pins three-stated (default). 1 = BBP Pins active
PC [5:0] Bit 5-0	**Port Control Register bits** — Each of PC(5:0) bits controls the functionality of the corresponding port pin. When a PC[i] bit is set, the corresponding port pin is configured as a BBP pin. When a PC [i] bit is cleared, the corresponding port pin is configured as GPIO pin. The mapping between the PCRB bits and the actual BBP ports is: PC[5] - STDB PC[4] - SRDB PC[3] - SCKB PC[2] - SC2B PC[1] - SC1B PC[0] - SC0B	0 = Corresponding Port Pin is a GPIO pin (default). 1 = Corresponding Port Pin is a BBP pin

14.4.3 BBP Receive Shift Register

This 16-bit shift register receives the incoming data from the serial receive data pin. Data is shifted in by the selected (internal/external) bit clock when the associated frame sync I/O is asserted. Data is assumed to be received MSB first if SHFD equals zero and LSB first if SHFD equals one. Data is transferred to the BBP receive data register after 8, 12 or 16 serial clock cycles were counted, depending on the word-length control bits in the CRAB. The BBP Receive Shift Register is not user accessible.

14.4.4 BBP Transmit Shift Register

This 16-bit shift registers contain the data being transmitted. Data is shifted out to the serial transmit data pins by the selected (internal/external) bit clock when the associated frame sync I/O is asserted. The number of bits shifted out before the shift register is considered empty and may be written to again can be 8, 12 or 16 bits (determined by the word-length control bits in CRAB). The data to be transmitted occupies the most significant portion of the shift register. The unused portion of the register is ignored. Data is shifted out of this register MSB first if SHFD equals zero and LSB first if SHFD equals one. The BBP Transmit Shift Register is not user accessible.

14.5 Operating Modes

BBP operating mode are selected by the BBP control registers (CRAB, CRBB and CRCB). The main operating mode are described in the following paragraphs.

14.5.1 BBP After Reset

Hardware or software reset clears the port control register bits and the port direction control register bits, which configure all I/O as general-purpose input and all pins at High Impedance. The BBP is reset while all BBP pins are programmed as general-purpose I/O and is active only if at least one of the BBP I/O pins is programmed as BBP pin.

14.5.2 BBP Initialization

The correct way to initialize the BBP is as follows:

1. Hardware, software, BBP individual, or STOP reset
2. Program BBP control according to the desired functionality using BBP registers.
3. Program the appropriate GPIO registers for each multiplexed BBP pin configured as an input or an output:
   a) For outputs, the appropriate GPIO Port Data Direction Register bit should be set to 1, and the appropriate GPIO Port Configuration Register bits should be set to select the BBP pin functionality.
   b) For inputs, the appropriate GPIO Port Data Direction Register bit should be set to 0, and the appropriate GPIO Port Alternate Input Register bits should be set to select the BBP pin functionality.

Refer to Chapter 23, “General Purpose Input/Output (GPIO),” and the IO Multiplexing description in Section 23.5, “GPIO Register Summary,” for more information on which BBP pins are multiplexed, and how the GPIO Port registers should be programmed.

During program execution, PC5-PC0 bits in the GPIO port control register (PCRB) may be cleared causing the BBP to stop serial activity and enter the individual reset state. All status bits of the interface will be set to their reset state; however, the contents of CRAB, CRBB and CRCB are not affected. This procedure allows the DSP program to reset each interface separately from the other internal peripherals.

To ensure proper operation of the interface, the DSP program must reset the BBP before changing any of its control registers (except for CRBB bits, TEIE, REIE, TLIE, RLIE, TIE, RIE, TE, RE, TCE, RCE, TCIE, RCIE, OF1 or OF0).

14.5.3 BBP Exceptions

The BBP can generate the following different exceptions, ordered from the highest to the lowest priority:

1. BBP Receive Data with Exception Status – occurs when the receive exception interrupt is enabled, the receive data register is full, and a receiver overrun error has occurred. ROE is cleared by first reading the SSISRB and then reading RXB.
2. BBP Receive Data – occurs when the receive interrupt is enabled, the receive data register is full, and no receive error conditions exist. Reading RXB clears the pending interrupt. This error-free interrupt can use a fast interrupt service routine for minimum overhead.
3. BBP Receive Last slot interrupt occurs after the last slot of the frame ended (in network mode only). Using the Receive last slot interrupt guarantees that the previous frame was serviced with the previous setting and the new frame will be serviced with the new setting without synchronization problems. Note that the maximum Receive last slot interrupt service time, should not exceed N-1 BBP bits service time (Where N is the number of bits in a slot).
4. BBP Receive Count Interrupt – occurs when the receive count interrupt is enabled, and the receive counter is rolls-over. Using the receive count interrupt enables the user to service a specified number of frames.
5. BBP Transmit Data with Exception Status – occurs when the transmit exception interrupt is enabled, the transmit data register is empty, and a transmitter underrun error has occurred. TUE is cleared by first reading the SSISRB and then writing to the transmit data register, or to the TSR to clear the pending interrupt.
6. BBP Transmit Last slot interrupt occurs at the start of the last slot of the frame in network mode. Using the Transmit last slot interrupt guarantees that the previous frame was serviced with the previous setting and the new frame will be serviced with the new setting without synchronization problems. Note that the maximum Transmit last slot interrupt service time, should not exceed N-1 BBP bits service time (Where N is the number of bits in a slot).
7. BBP Transmit Data – occurs when the transmit interrupt is enabled, and the transmit data register is empty, and no transmitter error conditions exist. Writing to the TXB registers, or to the TSR will clear this interrupt. This error-free interrupt may use a fast interrupt service routine for minimum overhead.
8. BBP Transmit Count Interrupt – occurs when the transmit count interrupt is enabled, and the transmit counter rolls-over. Using the transmit count interrupt enables the user to service a specified number of frames.

14.5.4 Operating Modes – Normal, Network, and On-Demand

The BBP has three basic operating modes and many data/operation formats. These modes can be programmed by several bits in the BBP control registers. The data/operation formats available to the BBP are selected by setting or clearing control bits in the CRAB and CRCB. These control bits are DC4–DC0, WL1, WL0, MOD, SYN, FSL1, FSL0, FSR, FSP, CKP and SHFD.

14.5.4.1 Normal/Network/On-Demand Mode Selection

Selecting between the normal mode and network mode is accomplished by clearing or setting the MOD bit in the CRCB. For normal mode, the BBP functions with one data word of I/O per frame. For the network mode, 2 to 32 time slots per frame may be selected. During each frame, 0 to 32 data words of I/O may be received/transmitted. In either case, the transfers are periodic. The normal mode is typically used to transfer data to/from a single device. Network mode is typically used in time division multiplexed (TDM) networks of codecs or DSPs with multiple words per frame.

Setting the MOD bit in the CRCB, as for network mode, and setting the frame rate divider to zero (DC=00000) selects the on demand mode. This special case will not generate a periodic frame sync. A frame sync pulse will be generated only when data is available to transmit. The frame sync signal indicates the first time slot in the frame. The on-demand mode requires that the transmit frame sync be internal (output) and the receive frame sync be external (input). Therefore, for simplex operation, the synchronous mode could be used; however, for full-duplex operation, the asynchronous mode must be used. Data transmission that is data driven is enabled by writing data into the TXB. Although the BBP is double buffered, only one word can be written to the TXB, even if the transmit shift register is empty. The receive and transmit interrupts, function as usual using TDE and RDF; however, transmit underruns are impossible for on-demand transmission and are disabled. This mode is useful for interfacing to codecs requiring a continuous clock.

14.5.4.2 Synchronous/Asynchronous Operating Modes

The transmit and receive sections of this interface may be synchronous or asynchronous – i.e., the transmitter and receiver may use common clock and synchronization signals or they may have their own separate clock and sync signals (asynchronous operating mode). The SYN bit in CRCB selects synchronous or asynchronous operation. Since the BBP is designed to operate either synchronously or asynchronously, separate receive and transmit interrupts are provided.
When SYN equals zero, the BBP TXB and RXB clocks and frame sync sources are independent. If SYN equals one, the BBP TXB and RXB clocks and frame sync come from the same source (either external or internal).
Data clock and frame sync signals can be generated internally by the DSP or may be obtained from external sources. If internally generated, the BBP clock generator is used to derive bit clock and frame sync signals from the DSP internal system clock. The BBP clock generator consists of a selectable fixed prescaler and a programmable prescaler for bit rate clock generation and also a programmable frame-rate divider and a word-length divider for frame-rate sync-signal generation.

14.5.4.3 Frame Sync Selection

The transmitter and receiver can operate totally independent of each other. The transmitter can have either a bit-long or word-long frame-sync signal format, and the receiver can have the same or opposite format. The selection is made by programming FSL0 and FSL1 in the CRCB.

1. If FSL1 equals zero, the RXB frame sync is asserted during the entire data transfer period. This frame sync length is compatible with Motorola codecs, SPI serial peripherals, serial A/D and D/A converters, shift registers, and telecommunication PCM serial I/O.
2. If FSL1 equals one, the RXB frame sync pulses active for one bit clock immediately before the data transfer period. This frame sync length is compatible with Intel and National components, codecs, and telecommunication PCM serial I/O.

The ability to mix frame sync lengths is useful in configuring systems in which data is received from one type device (e.g., codec) and transmitted to a different type device.
FSL0 controls whether RXB and TXB have the same frame sync length. If FSL0 equals zero, RXB and TXB have the same frame sync length, which is selected by FSL1. If FSL0 equals one, RXB and TXB have different frame sync lengths, which are selected by FSL1. FSL0 is ignored when the SYN bit is set
FSR controls the relative timing of the word length frame sync as referred to the data word. When FSR is cleared the word length frame sync is generated (or expected) with the first bit of the data word. When FSR is set the word length frame sync is generated (or expected) with the last bit of the previous word. FSR is ignored when a bit length frame sync is selected.

The BBP does not support a bit length late frame sync. If the BBP receiver is configured for external frame sync, and is connected to a device that provides a bit length late frame sync, the BBP receiver may be configured for word length late frame sync to work around this issue, as long as the BBP divide ratio is not zero (DC[4:0] = 00000 with word length frame sync is not allowed)

FSP controls the polarity of the frame sync. When FSP is cleared the polarity of the frame sync is positive i.e. the frame sync signal is asserted high. When FSP is set the polarity of the frame sync is negative i.e. the frame sync is asserted low.
The BBP receiver looks for a receive frame sync leading edge (trailing edge if FSP is set) only when the previous frame is completed. If the frame sync goes high before the frame is completed (or before the last bit of the frame is received in the case of a bit frame sync or a word length frame sync with FSR set), the current frame sync will not be recognized, and the receiver will be internally disabled until the next frame sync. Frames do not have to be adjacent – i.e., a new frame sync does not have to immediately follow the previous frame. Gaps of arbitrary periods can occur between frames. The transmitter will be three-stated during these gaps.

14.5.4.4 Shift Direction Selection

Some data formats, such as those used by codecs, specify MSB first other data formats, such as the AES-EBU digital audio, specify LSB first. To interface with devices from both systems, the shift registers in the BBP are bidirectional. The MSB/LSB selection is made by programming SHFD in the CRCB.
If SHFD equals zero, data is shifted into the receive shift register MSB first and shifted out of the transmit shift register MSB first. If SHFD equals one, data is shifted into the receive shift register LSB first and shifted out of the transmit shift register LSB first.

14.5.4.5 Counters Operating Modes

The transmit and receive counters may be used as an individuals frame sync counters (both transmitter and receiver). Both are enabled by the TCE or RCE (transmitter or receiver counter enables), they are loaded with the TCRB/RCRB data. The counters will count frames and will use the frame sync signals as their source clock. When the counter rolls-over a DSP interrupt occurs when the interrupt enable bit (TCIE/RCIE) is set.

14.5.5 Flags

Two BBP pins (SC1B and SC0B) are available as serial I/O flags. Their operation can be controlled by SYN, SCD0 and SCD1 bits in the CRCB. The control bits (OF1 and OF0) and status bits (IF1 and IF0) are double buffered to/from SC1B and SC0B. Double buffering the flags keeps them in sync with TXB and RXB.

The flags are available in the synchronous mode only (SYN=1). Each flag can be separately programmed. When flag 0 is enabled, its direction is selected by SCD0, SCD0=1 as output and SCD0=0 as input. In the same way when flag1 is enabled, its direction is selected by SCD1, SCD1=1 as output and SCD1=0 as input.
When programmed as input, SC0B and SC1B value, respectively, are latched at the same time as the first bit is sampled of the receive data word. Since the input was latched, the signal on the input flag pin (SC0B and SC1B), can change without affecting the input flag until the first bit of the next receive data word. When received data word is latched by RXB, the latched values are then latched by the SSISRB IF0 and IF1 bits respectively and can be read by software.
When programed as output, SC0B and SC1B value, is driven by the value, from the CRBB OF0 and OF1 bits respectively, latched when the TXB is transferred to the transmit shift register. The value on SC0B or SC1B, will be stable from the same time the first bit of the transmit data word is transmitted until the first bit of the next transmit data word is transmitted. Software can change the CRBB OF0 and OF1 values and thus controlling the SC0B and SC1B pin values for each transmitted word.

14.6 BBP MAGIC Communication.

BBP will communicate to magic through A2DIGL. Kindly refer Fig 6.12 for BBP connectivity. STDB is connected to GPIO as well STDB0 pin of A2DIGL . Data to SCKB, SRDB, SC0Band SC1B. are, multiplexed inputs from A2DIGL

For BBP MAGIC serial communication protocol kindly refer 6.6 Magic serial communication protocol enable on bbp pins.

14.7 Neptune LTE Specific Requirement

Even though bbp_epcpd[5:0] pins are bidirectional pins (each pin can be configured as input/output in gpio mode), in Neptune the functionality of these pins is restricted.

STDB (bbp_epcpd[5]) is always configured as output pin and does not support configuration as an input.

SC2B has been configured to support bidirectional functionality in Neptune LTE.

SCKB configuration depend on A2DIGL and BBP settings. BBP can configure SCKB as an output. If the BBP configures SCKB as an input, A2DIGL determines whether to drive SCKB directly or to route SCKB from the external SCKB input. The BBP uses MUXCTL bit TXSEL3 to select the input source to the BBP SCKB. See the description of TXSEL3 in the A2DIGL MUXCTL register and Figure 6-15 in the Mixed Signals Interconnect Section 6.4.

The rest of the pins will always be configured as input only pins as A2DIGL is driving them. The programmer has to take care while programming these pins. Even if we try to program the bbp_epcpd[4:0] pins as not all pins will not get configured into output mode. Programmer has to configure it in the input mode only. Kindly Refer Fig14-4.

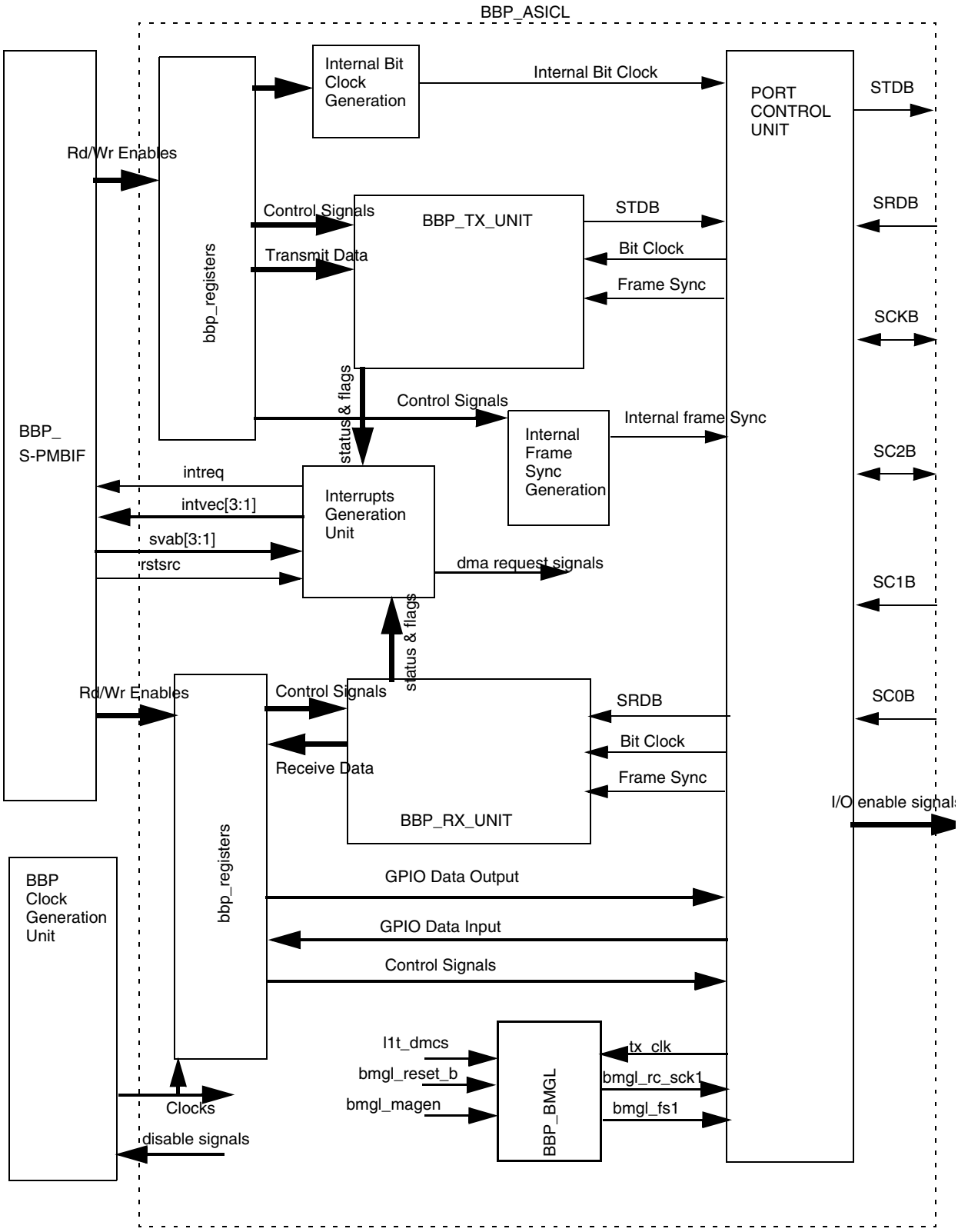

BBP_ASICL
Internal Bit Clock Generation
Internal Bit Clock
PORT CONTROL UNIT
STDB
SRDB
SCKB
SC2B
SC1B
SC0B
I/O enable signals
Rd/Wr Enables
Control Signals
Transmit Data
BBP_TX_UNIT
STDB
Bit Clock
Frame Sync
bbp_registers
status & flags
Control Signals
Internal frame Sync
Internal Frame Sync Generation
BBP_ S-PMBIF
intreq
intvec[3:1]
svab[3:1]
rstsrc
Interrupts Generation Unit
dma request signals
status & flags
Rd/Wr Enables
Control Signals
Receive Data
SRDB
Bit Clock
Frame Sync
BBP_RX_UNIT
bbp_registers
BBP Clock Generation Unit
GPIO Data Output
GPIO Data Input
Control Signals
Clocks
disable signals
l1t_dmcs
bmgl_reset_b
bmgl_magen
BBP_BMGL
tx_clk
bmgl_rc_sck1
bmgl_fs1

Figure 14-4.

Chapter 15
DSP Timer Module (Dtimer)

Revision History Table

Revision	Date	Author	Changes
1.0	10/8/99		Initial Release.
2.0	17/10/2001		Initial doc has been taken from Neptune spec.
3.0	17/10/2001	Akanksha Mehta	1)In Pin diagram scan related ports have been removed except scan_en , scan_mode ports. 2)New port gclkw_b added according to Patriot Indy project requirements. 3) Renaming of signal bdis0 to mcdis0, bdis1 to mcdis1, vab to vab_rd, mirq_ to mirq_b.
4.0	20/03/2002	Girjesh K. Soni	Initial version for Neptune LTS
5.0	23/7/2002	Mark Babcock, Shannon Osgood	Updated per DDTS# DSPhl14797.
	05/07/03		Updated for LTE specification release.

15.1 Introduction

This section describes the DSP Timer module, composed of a common 14-bit clock prescaler and three independent and identical general purpose 16-bit timer/event counters, each with its own register set.

Each timer can use internal or external clocking and can interrupt the DSP after a specified number of events (clocks) or can signal an external device after counting internal events. Each timer connects to the external world through one bidirectional pin TIO. When TIO is configured as input, the timer functions as an external event counter or can measure external pulse width/signal period. When TIO is used as output the timer is functioning as either a timer, a watchdog or a Pulse Width Modulator. When the TIO pin is not used by the timer it can be used as a General Purpose Input/Output Pin.

15.2 Architecture

15.2.1 DSP Timer Module Block Diagram

Figure 15-1 shows a block diagram of the DSP Timer module. It includes a 16-bit Timer Prescaler Load Register (TPLR), a 16-bit Timer Prescaler Count Register (TPCR), a 14-bit counter and three timers. (Each of the three timers may use the Prescaled Clock as its clock source).

The14-bit Prescaled Clock counter decrements on each rising edge of the prescaler input clock. The counter is enabled when at least one of the three timers is both enabled ($TE_i = 1$) and using the prescaler output as its source ($PCE_i = 1$).

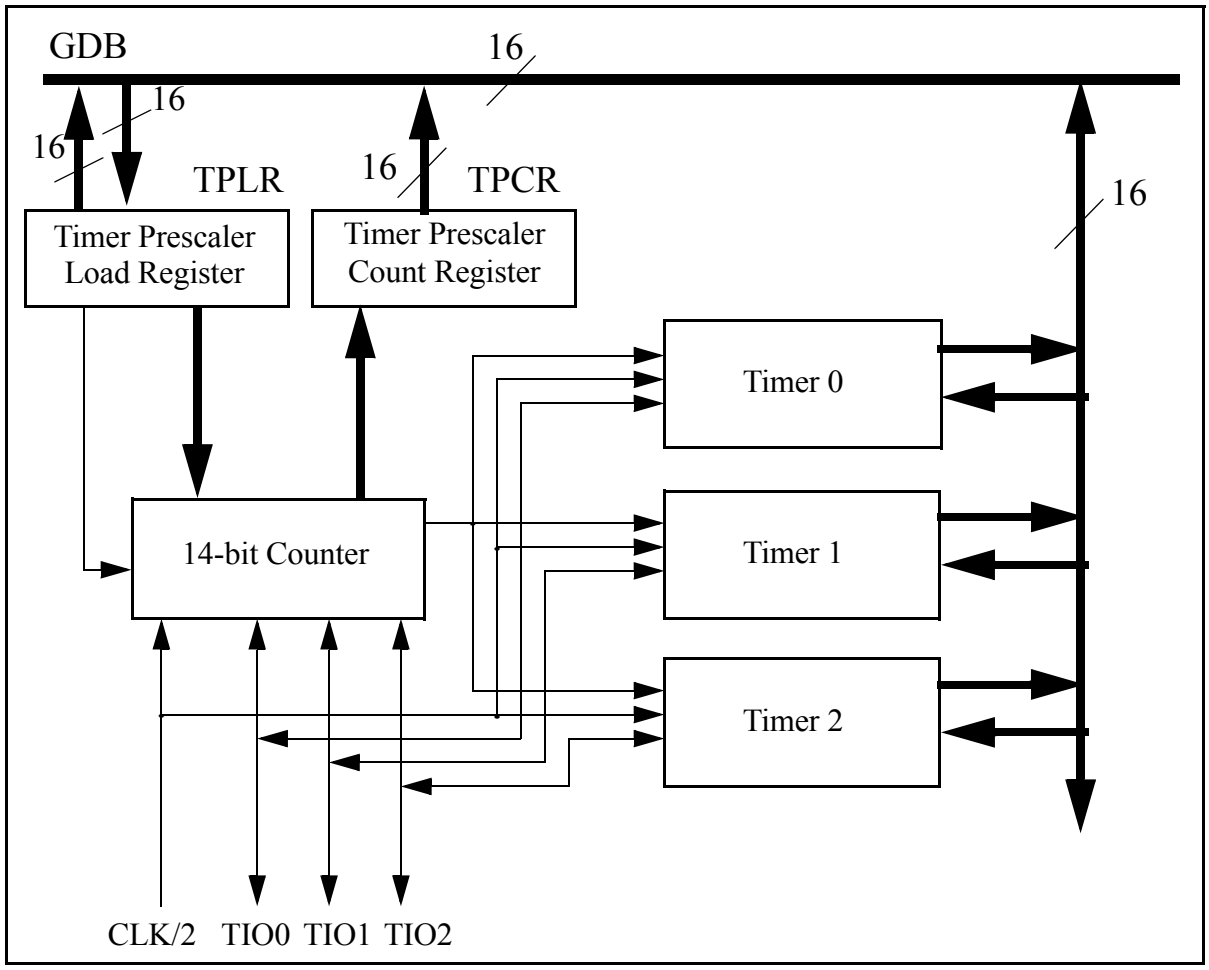


Figure 15-1. DSP Timer Module Block Diagram

15.2.2 Timer Block Diagram

Figure 15-2 shows a block diagram of a single timer. It includes a 16-bit counter, a 16-bit read-write Timer Control and Status Register (TCSR), a 16-bit read only Timer Count Register (TCR), a 16-bit write only Timer Load Register (TLR), a 16-bit read-write Timer Compare Register (TCPR), and logic for clock selection and interrupt trigger generation. The DSP views each timer as a memory mapped peripheral occupying four 16-bit words in the X data memory space. The user may use standard polled or interrupt programming techniques.

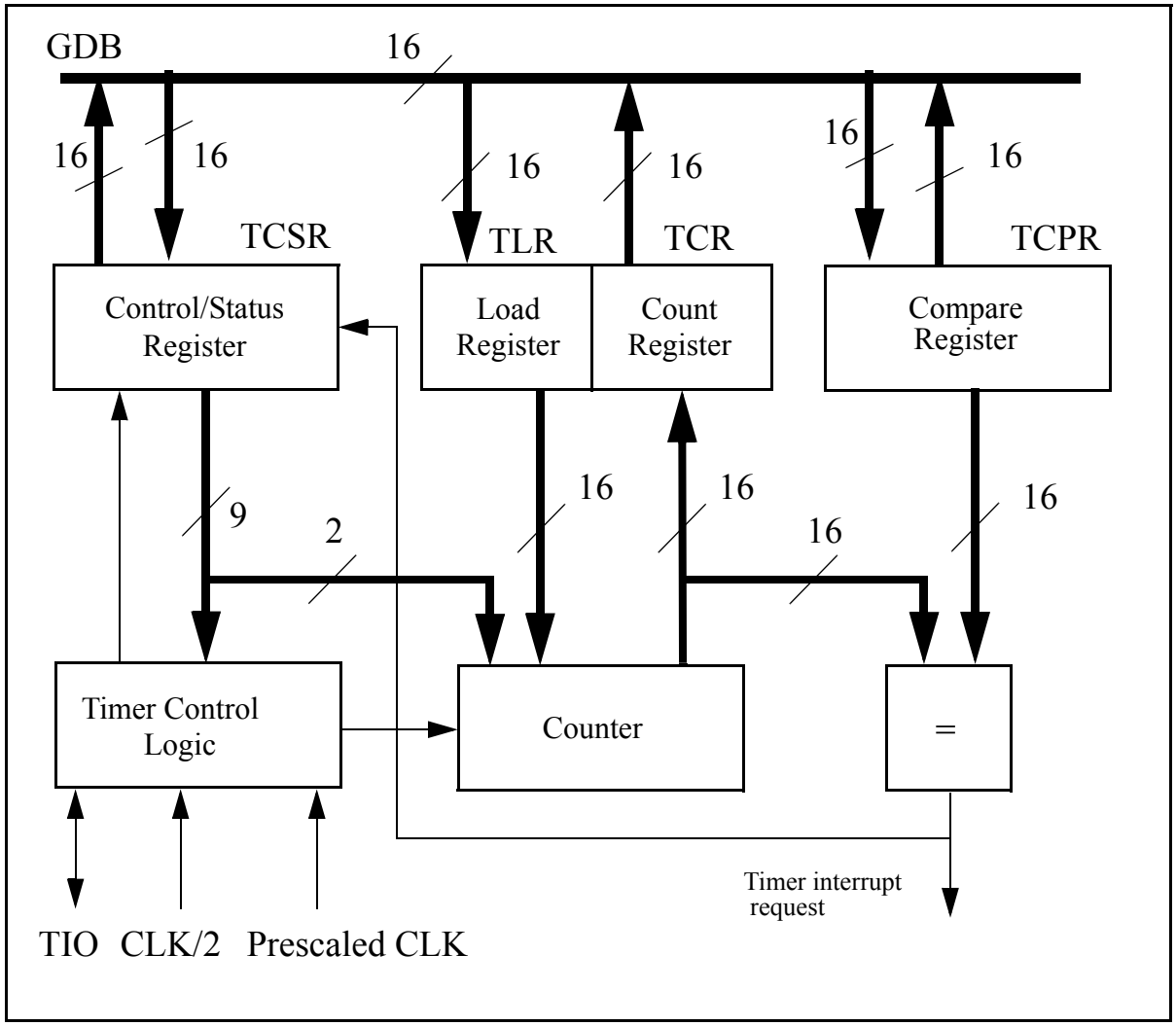


Figure 15-2. 16-bit Timer Module Block Diagram

The notation for a register in a specific timer block contains the generic register name (TCSR, TLR, TCR, or TCPR) appended with the block index (0, 1, or 2).

15.2.3 Pin Diagram

Table 15-1. DTimer Pin Diagram

LOCAL SIGNAL NAME	REMARKS
ndbgctl	indicates that core is in debug mode
mcab_rd[6:0]	DSP read address
mcab_wr[6:0]	DSP write address
mcdis0	core pipeline stall indication
mcdis1	core pipeline stall indication
pivack	peripheral module interrupt acknowledge

Table 15-1. DTimer Pin Diagram

LOCAL SIGNAL NAME	REMARKS
pivrd	vector read strobe
mcsel_wr	write select
mcsel_rd	read select
pivsse	interrupt vector source select
gclkw	DSP clock
gclkw_b	Inverted DSP clock
ipt_clk_se	scan enable source signal
ipt_scan_mode	scan mode
gdb_wr[15:0]	core gobal write data
gdb_rd[15:0]	core gobal read data
pires	peripheral module software reset
res	peripheral module hardware reset
bcstop	indicates that core is in stop mode
vab_rd[7:1]	interrupt vector
mirq_b	peripheral module interrupt request
pdtimer_fiod2_in	connected to TIO2 of GPIO
pdtimer_fiod1_in	connected toTIO1 of GPIO
pdtimer_fiod0_in	connected toTIO0 of GPIO
pdtimer_fiod2_out	connected to TIO2 of GPIO
pdtimer_fiod1_out	connected toTIO1 of GPIO
pdtimer_fiod0_out	connected toTIO0 of GPIO
pdtimer_fios2	output enable for pdtimer_fiod2
pdtimer_fios1	output enable for pdtimer_fiod1
pdtimer_fios0	output enable for pdtimer_fiod0
pdtimer_fbie2	input enable for pdtimer_fiod2
pdtimer_fbie1	input enable for pdtimer_fiod1
pdtimer_fbie0	input enable for pdtimer_fiod0

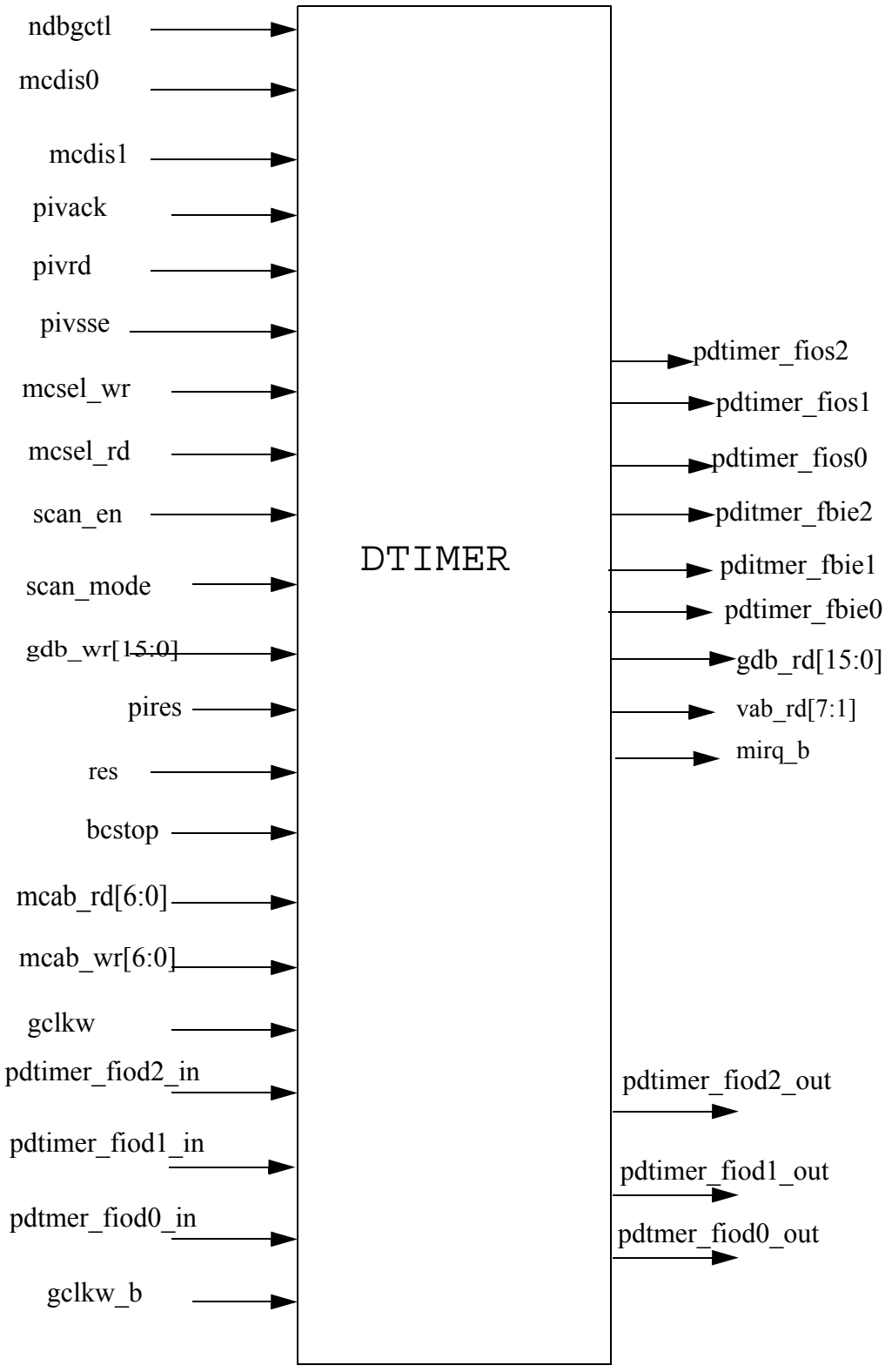


Figure 15-3. DTimer Pin Diagram

15.3 DSP Timer Module Register Summary

The registers comprising the DSP Timer Module are shown in Table 15-2 below.

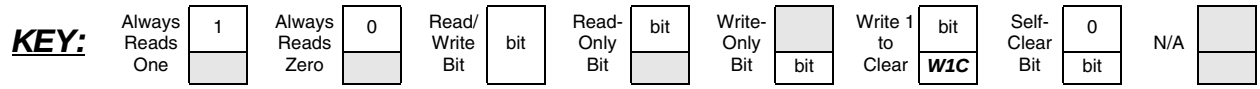


Table 15-2. DSP Timer Module Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPCR (X:$FFD2)	R	0	0	PC[13:0]													
	W																
TPLR (X:$FFD3)	R	PS[1:0]		PL[13:0]													
	W																
TCR2 (X:$FFD4)	R	TCR2[15:0]															
	W																
TCPR2 (X:$FFD5)	R	TCPR2[15:0]															
	W																
TLR2 (X:$FFD6)	R																
	W	TLR2[15:0]															
TCSR2 (X:$FFD7)	R	PCE	TCF	TOF	DO	DI	DIR	TRM	INV	TC[3:0]				0	TCIE	TOIE	TE
	W		*W1C*	*W1C*													
TCR1 (X:$FFD8)	R	TCR1[15:0]															
	W																
TCPR1 (X:$FFD9)	R	TCPR1[15:0]															
	W																
TLR1 (X:$FFDA)	R																
	W	TLR1[15:0]															
TCSR1 (X:$FFDB)	R	PCE	TCF	TOF	DO	DI	DIR	TRM	INV	TC[3:0]				0	TCIE	TOIE	TE
	W		*W1C*	*W1C*													
TCR0 (X:$FFDC)	R	TCR0[15:0]															
	W																
TCPR0 (X:$FFDD)	R	TCPR0[15:0]															
	W																
TLR0 (X:$FFDE)	R																
	W	TLR0[15:0]															
TCSR0 (X:$FFDF)	R	PCE	TCF	TOF	DO	DI	DIR	TRM	INV	TC[3:0]				0	TCIE	TOIE	TE
	W		*W1C*	*W1C*													

15.3.1 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various DSP Timer registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

The Timer Prescaler Load Register is a 16-bit read/write register that controls the prescaler divide factor and the source for the prescaler input clock. The control bits are described in Table 15-3.

TPLR Timer Prescaler Load Register **Addr X:$FFD3**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	PS[1:0]		PL[13:0]													
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-3. TPLR Description

Name	Description	Settings
PS[1:0] Bits 15-14	**Prescaler Clock Source --** Selects the source clock for the the prescaler counter from one of the Internal CLK/2 and the three TIO signals. The Internal CLK/2 signal is the DSP clock divided by two. The TIO signals may be selected regardless of the operating mode of the appropriate timer. **Note:** If the prescaler clock is external, the prescaler counter will be incremented by the transitions on the TIO pin. An external clock supplying a TIO signal is synchronized to the internal clock, so its frequency should be lower than the internal operating frequency divided by 4 (CLK/4). **Note:** The PS1-PS0 bits should be changed only when the prescaler counter is disabled to ensure proper functionality.	**PS[1:0]** / **PRESCALER CLOCK SOURCE** 00 — Internal CLK/2 01 — TIO0 10 — TIO1 11 — TIO2
PL[13:0] Bits 13-0	**Prescaler Preload Value --** 14-bit value to be loaded into the prescaler counter when the counter reaches zero or changes to from disabled to enabled state. If PL0-PL13 = N then the prescaler will count N+1 source clock cycles before generating a prescaled clock pulse. Therefore the prescaler divide factor is preload value + 1. PL0-PL13 bits are cleared by hardware RESET and software RESET (RESET instruction)..	N/A

The Timer Prescaler Count Register is a 16-bit read only register that reflects the current value in the prescaler counter.

TPCR Timer Prescaler Count Register **Addr X:$FFD2**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
			PC[13:0]													
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-4. TPCR Description

Name	Description	Settings
Bits 15-14	**Reserved Bits --** These bits are read as 0 and should be written as 0 for future compatability	N/A
PC[13:0] Bits 13-0	**Prescaler Count Value --** 14-bit read-only value reflecting the current value in the prescaler counter.	N/A

TCR Timer Count Register **Addr (TCR0)X:$FFDC (TCR1)X:$FFD8 (TCR2)X:$FFD4**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	TCR[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-5. TCR Description

Name	Description	Settings
TCR[15:0] Bits 15-0	**Count Value --** The Timer Count Register is a 16-bit read-only register. In Timer and Watchdog Modes the counter contents can be read at any time by reading the TCR register. In Measurement Modes the TCR will be loaded with the current value of the counter on the appropriate edge of the input signal and its value can be read to determine the width, period or delay of the leading edge of the input signal (incoming on the TIO pin)	N/A

TCPR Timer Compare Register **Addr (TCPR0)X:$FFD D (TCPR1)X:$FFD9 (TCPR2)X:$FFD5**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	CMP[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 15-6. TCPR Description

Name	Description	Settings
CMP[15:0] Bits 15-0	**Compare Value --** The compare register is a 16-bit read-write register which contains the value to be compared to the counter value. The counter value is compared against the value contained by the TCPR on every timer clock after the Timer Enable bit is set (TE=1). When the compare matches, the TCF bit is set and, if interrupts are enabled (TCIE=1), an interrupt is generated. In Measurement Modes the TCPR is ignored.	N/A

TLR Timer Load Register **Addr (TLR0)X:$FFDE (TLR1)X:$FFDA (TLR2)X:$FFD6**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	TLR[15:0]															
TYPE	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
RESET	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Name	Description	Settings
TLR[15:0] Bits 15-0	**Load Value --** The Timer Load Register is a 16-bit write-only register. In all modes the counter is preloaded with the TLR value after the Timer Enable bit is set (TE=1) and a first event occurs. In Timer Modes, if timer reload mode is set (TRM=1), the counter is reloaded each time after it has reached the value contained by the Timer Compare Register and the new event occurs. In Measurement Modes, if timer reload mode is set (TRM=1), the counter is reloaded with the TLR value on each appropriate edge of the input signal, after the Timer Enable bit is set (TE=1). In PWM Modes, if timer reload mode is set (TRM=1), the counter is reloaded each time after it has overflowed and the new event occurs. In Watchdog Modes, if timer reload mode is set (TRM=1), the counter is reloaded each time after it has reached the value contained by the Timer Compare Register and the new event occurs. In this mode, the counter is also reloaded whenever the TLR is written with a new value while the Timer Enable bit is set (TE = 1). In all modes, if TRM is cleared (TRM=0), the counter operates as free running counter.	N/A

The control/status register is a 16-bit read/write register that controls the timer and reflects its status. The control and status bits are described in Table 15-7.

TCSR Timer Control/Status Register **Addr (TCSR0)X:$FFDF (TCSR1)X:$FFDB (TCSR2)X:$FFD7**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	PCE	TCF	TOF	DO	DI	DIR	TRM	INV	TC[3:0]					TCIE	TOIE	TE
TYPE	rw	w1c	w1c	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-7. TCSR Description

Name	Description	Settings
PCE Bit 15	**Prescaled Clock Enable --** The prescaled clock enable bit is used to select the prescaled clock as the timer source clock. Note: The PCE bit should be changed only when TE=0 (i.e. timer disabled) to ensure proper functionality. Note: The source clock for the prescaler is determined only by the Prescaler Source bits (PS0-PS1) of the Timer Prescaler Load Register (TPLR). Therefore a timer may be clocked by prescaled clock derived from the TIO of another timer. PCE is cleared by hardware $\overline{\text{RESET}}$ and software RESET (RESET instruction).	0 = the timer uses either internal (CLK/2) or external (TIO) source clock as determined by the timer operating mode. 1 = the prescaler output is used as the timer source clock for the counter regardless of the timer operating mode.

Table 15-7. TCSR Description

Name	Description	Settings
TCF Bit 14	**Timer Compare Flag --** In the Timer PWM and Watchdog Modes, this bit is set when (N-M+1) events are counted, where N is the value in the compare register and M is TLR value. In the Measurement Modes this bit is set when the measurement has been completed. This bit is cleared by writing a one into the TCF bit, and is also cleared when the timer compare interrupt is serviced (timer compare interrupt acknowledge), by the STOP instruction, or by disabling the timer (TE=0), in addition to hardware or software reset. Writing 0 has no effect. **Note:** Writing a zero in the TOF or TCF bit can be done with the Bit Test and Clear (BCLR) instruction. The state of the tested bit will be stored in the carry bit of the status register (SR). **Note:** To ensure that only the desired bit is cleared, the programmer should not use the BSET command. The proper way to clear these bits is to write (using the MOVE(P) instruction) logic one (‘1’) to the flag to be cleared and zero to the other flag.	0 = No compare has occurred 1 = Sucessful compare has occurred since bit was last cleared
TOF Bit 13	**Timer Overflow Flag--** Indicates whether counter wraparound has occurred. This bit is cleared by writing a one into the TOF bit, and is also cleared when the timer overflow interrupt is serviced (timer overflow interrupt acknowledge), by the STOP instruction, or by disabling the timer (TE=0), in addition to hardware or software reset. Writing 0 has no effect. **Note:** Writing a zero in the TOF or TCF bit can be done with the Bit Test and Clear (BCLR) instruction. The state of the tested bit will be stored in the carry bit of the status register (SR). **Note:** To ensure that only the desired bit is cleared, the programmer should not use the BSET command. The proper way to clear these bits is to write (using the MOVE(P) instruction) logic one (‘1’) to the flag to be cleared and zero to the other flag.	0 = Counter wraparound has not occurred 1 = Counter wraparound has occurred since bit was last cleared

Table 15-7. TCSR Description

Name	Description	Settings
DO Bit 12	**Data Output --** If GPIO mode is enabled (TC[3:0] = '0000') and TIO is an output (DIR=1), writing the DO bit will write the data to the TIO pin. If the INV bit is set the data on the TIO pin will be inverted. That is: TIO = DO ⊕ INV, where TIO is the TIO pin,and INV is TCSR[8]. DO is cleared by hardware $\overline{\text{RESET}}$ and software RESET (RESET instruction). **Note:** When not in GPIO mode, writing the DO bit will have no effect.	N/A
DI Bit 11	**Data Input --** The DI bit reflects the value of TIO pin according to the INV bit. Reading the DI bit will read the TIO pin if INV=0, or the inverted TIO pin if INV=1. That is, the DI bit reads as TIO ⊕ INV, where TIO is the TIO pin, and INV is TCSR[8].	N/A
DIR Bit 10	**Direction --** Set the direction of the TIO pin when used as a General Purpose Input/Output (GPIO) pin. If not in GPIO mode (TC[3:0] != '0000'), DIR has no effect.	0 = TIO pin is an input 1 = TIO pin is an output

Table 15-7. TCSR Description

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>TRM
Bit 9</td><td>Timer Reload Mode -- Controls the counter reload operation according to the table to the right. TRM is cleared by hardware RESET and software RESET (RESET instruction)</td><td>
<table>
<tr><th>TRM</th><th>Timer Mode</th><th>When Counter is Reloaded with TLR</th></tr>
<tr><td>1</td><td>Timer</td><td>The counter is preloaded with the TLR value after the Timer Enable bit is set (TE=1) and a first event occurs. The counter is reloaded each time after it has reached the value contained by the Timer Compare Register and the new event occurs.</td></tr>
<tr><td></td><td>Measure</td><td>The counter is reloaded with the TLR value on each appropriate edge of the input signal, after the Timer Enable bit is set (TE=1)</td></tr>
<tr><td></td><td>PWM</td><td>The counter is reloaded each time after it has overflowed and the new event occurs</td></tr>
<tr><td></td><td>Watch-dog</td><td>The counter is preloaded with the TLR value after the Timer Enable bit is set (TE=1) and a first event occurs. The counter is reloaded after it has reached the value contained by the Timer Compare Register and the new event occurs. Also, whenever the TLR is written with a new value while the Timer Enable bit is set (TE = 1)</td></tr>
<tr><td>0</td><td>All</td><td>Never -- the counter is free- running, incrementing on each incoming event.</td></tr>
</table>
</td></tr>
</table>

Table 15-7. TCSR Description

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>INV
Bit 8</td><td>Inverter -- Affects the polarity of the external incoming signal on the TIO pin (when TIO is an input) and affects the polarity of the pulse generated on the TIO pin (when TIO is an output).

Note: The INV bit affects both the timer and the GPIO modes of operation.

Note: The INV bit should be changed only when TE=0 (i.e. timer disabled), or in GPIO mode of operation, to ensure proper functionality.

Note: The polarity of the prescaler source clock, when the TIO is used as input to the prescaler, is not affected by the corresponding INV bit.</td><td>
<table>
<tr><th>DIR</th><th>Mode</th><th>INV</th><th>Effect</th></tr>
<tr><td rowspan="6">0</td><td rowspan="2">Timer</td><td>0</td><td>0 to 1 transitions on the TIO input pin increments the counter</td></tr>
<tr><td>1</td><td>1 to 0 transitions on the TIO input pin increments the counter</td></tr>
<tr><td rowspan="2">Input Width Mode</td><td>0</td><td>Measure high pulse</td></tr>
<tr><td>1</td><td>Measure low pulse</td></tr>
<tr><td rowspan="2">Input Period Mode</td><td>0</td><td>Measure period between rising edges</td></tr>
<tr><td>1</td><td>Measure period between falling edges</td></tr>
<tr><td rowspan="2">1</td><td rowspan="2">any</td><td>0</td><td>Generate pulses of positive polarity.</td></tr>
<tr><td>1</td><td>Generate pulses of negative polarity.</td></tr>
</table>
</td></tr>
</table>

Table 15-7. TCSR Description

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>TC[3:0]
Bit 7-4</td><td>Timer Control -- 4-bit value controlling the source of the timer clock, the behavior of the TIO pin and the timer mode of operation. The table to the right summarizes the functionality of the TC bits. A detailed description of the timer operating modes is given in Section 15.4, “Timer Functional Description (Modes of Operation).” . TC0-TC3 bits are cleared by hardware RESET and software RESET (RESET instruction).
Note: If the clock is external, the counter will be incremented by the transitions on the TIO pin. The external clock is internally synchronized to the internal clock and its frequency should be lower than the internal operating frequency divided by 4 (CLK/4).
Note: Note that the TC bits should only be changed when TE = 0 to ensure proper functionality.
Note: Writing an value that corresponds to the reserved mode encoding WILL place the timer in a illegal or undefine state.</td><td>
<table>
<tr><th>TC [3:0]</th><th>TIO</th><th>CLOCK</th><th>MODE</th></tr>
<tr><td>0000</td><td>GPIO</td><td>Internal</td><td>Timer GPIO</td></tr>
<tr><td>0001</td><td>Output</td><td>Internal</td><td>Timer Pulse</td></tr>
<tr><td>0010</td><td>Output</td><td>Internal</td><td>Timer Toggle</td></tr>
<tr><td>0011</td><td>Input</td><td>External</td><td>Event Counter</td></tr>
<tr><td>0100</td><td>Input</td><td>Internal</td><td>Input Width</td></tr>
<tr><td>0101</td><td>Input</td><td>Internal</td><td>Input Period</td></tr>
<tr><td>0110</td><td>Input</td><td>Internal</td><td>Capture</td></tr>
<tr><td>0111</td><td>Output</td><td>Internal</td><td>Pulse Width Modulation (PWM)</td></tr>
<tr><td>1000</td><td>-</td><td>-</td><td>Reserved</td></tr>
<tr><td>1001</td><td>Output</td><td>Internal</td><td>Watchdog Pulse</td></tr>
<tr><td>1010</td><td>Output</td><td>Internal</td><td>Watchdog Toggle</td></tr>
<tr><td>1011-1111</td><td>-</td><td>-</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>Bit 3</td><td>Reserved Bit -- Alreads reads as zero, and should be written as 0 for future compatability.</td><td>N/A</td></tr>
<tr><td>TCIE
Bit 2</td><td>Timer Compare Interrupt Enable -- Enables the timer compare interrupts. The compare interrupt will be generated after the counter matches the compare register (in the Timer, PWM or Watchdog Modes). If TCPR is loaded with N, an interrupt will occur after (N-M+1) events, where M is TLR value. TCIE is cleared by hardware RESET and software RESET (RESET instruction).</td><td>1 = enable the compare interrupts
0 = disabled the compare interrupts</td></tr>
<tr><td>TOIE
Bit 1</td><td>Timer Overflow Interrupt Enable -- Enables the timer overflow interrupt which is generated when the associated timer wraps (i.e. the counter value changes from $FFFF to $0000 when a new event occurs). TOIE is cleared by hardware RESET and software RESET (RESET instruction).</td><td>1 = enable the overflow interrupt
0 = disable the overflow interrupt</td></tr>
</table>

Table 15-7. TCSR Description

Name	Description	Settings
TE Bit 0	**Timer Enable --** Used to enable or disable the timer. Enabling the timer will start the timer in a mode according to the TC[3:0] bitfield. TE is cleared by hardware $\overline{\text{RESET}}$ and software RESET (RESET instruction). **Note:** When all the three timers are disabled and not in GPIO mode, all three TIO pins becomes tri-state. In order to prevent undesired spikes on the TIO pins (when switching from tri-state into active state) it is recommended to use external pull-ups or pull downs resistors tied to TIO pins.	1 = enable the timer and clear the timer counter 0 = disable the timer

15.4 Timer Functional Description (Modes of Operation)

This section describes the various modes of operation of the timer module.

15.4.1 Timer Modes

15.4.1.1 Timer mode, internal clock, no output (mode 0)

This mode is defined by TC3-TC0 equal 0000. In this mode the counter is cleared after TE is set and loaded with the TLR value on the first timer clock derived either from the DSP clock divided by two (CLK/2) or from the prescaled clock input. The following timer clocks will increment the counter. When the counter matches the value contained by the TCPR, the TCF bit in TCSR is set and, if the TCIE is set, a compare interrupt is generated. At the next timer clock the counter is loaded with TLR value (if TRM is set) and the count is resumed. If TRM is cleared, the counter continues to be incremented on each timer clock. If counter wraparound has occurred the TOF bit is set and, if the TOIE is set, an overflow interrupt is generated. This process is repeated until the timer is disabled (TE=0). The counter contents can be read at any time by reading the TCR register.

15.4.1.2 Timer mode, internal clock, output pulse enable (mode 1)

This mode is defined by TC3-TC0 equal 0001. In this mode the counter is cleared after TE is set and loaded with the TLR value on the first timer clock derived either from the DSP clock divided by two (CLK/2) or from the prescaled clock input. The following timer clocks will increment the counter. When the counter matches the value contained by the TCPR, the TCF bit in TCSR is set and, if the TCIE is set, a compare interrupt is generated. At the next timer clock the counter is loaded with TLR value (if TRM is set) and the count is resumed. If TRM is cleared, the counter continues to be incremented on each timer clock. This process is repeated until the timer is disabled (TE=0). Each time the counter matches the TCPR value a pulse will be output on the TIO pin with the width equal to timer clock period. The pulse polarity is determined by the INV bit. If counter wraparound has occurred the TOF bit is set and, if the TOIE is set, an overflow interrupt is generated. The counter contents can be read at any time by reading the TCR register.

Note: After TE=1 the TIO pin output value is set equal to INV bit, to guarantee the correct first pin transition.

15.4.1.3 Timer mode, internal clock, output toggle enable (mode 2)

This mode is defined by TC3-TC0 equal 0010. In this mode the counter is cleared after TE is set and loaded with the TLR value on the first timer clock derived either from the DSP clock divided by two (CLK/2) or from the prescaled clock input. The following timer clocks will increment the counter. When the counter matches the value of the TCPR, the TIO output pin will be toggled, the TCF bit in TCSR is set and, if the TCIE is set, a compare interrupt is generated. At the next timer clock the counter is loaded with TLR value (if TRM is set) and the count is resumed. If TRM is cleared, the counter continues to be incremented on each timer clock. This process is repeated until the timer is disabled (TE=0). The TIO polarity is determined by the INV bit. On the first match the TIO output will be set if INV=0 or cleared if INV=1. If counter wraparound has occurred the TOF bit is set and, if the TOIE is set, an overflow interrupt is generated. The counter contents can be read at any time by reading the TCR register.

Note: After TE=1 the TIO pin output value is set equal to INV bit, to guarantee the correct first pin transition.

15.4.1.4 Timer mode, external clock, event counter (mode 3)

This mode is defined by TC3-TC0 equal 0011. In this mode the counter is cleared after TE is set and loaded with the TLR value on the first transitions on the source clock which can be either the TIO input pin or the prescaled clock input. The following transitions will increment the counter. When the counter matches the value contained by TCPR, the TCF bit in TCSR is set and, if the TCIE is set, a compare interrupt is generated. At the next transitions the counter is loaded with TLR value (if TRM is set) and the count is resumed. If TRM is cleared, the counter continues to be incremented with each transitions on the source clock. This process is repeated until the timer is disabled (TE=0). The INV bit determines whether 0 to 1 transitions (INV=0) or 1 to 0 transitions (INV=1) will increment the counter. If counter wraparound has occurred the TOF bit is set and, if the TOIE is set, an overflow interrupt is generated. The counter contents can be read at any time by reading the TCR register. The external clock is internally synchronized to the internal clock and its frequency should be lower than the internal operating frequency divided by 4 (CLK/4).

15.4.2 Measurement Modes

15.4.2.1 Pulse width measurement mode (mode 4)

This mode is defined by TC3-TC0 equal 0100. In this mode the counter is cleared after TE is set and, after the first appropriate transition (as determined by the INV bit) occurring on TIO input pin, it is loaded with the TLR value on the first timer clock derived either from the DSP clock divided by two (CLK/2) or from the prescaled clock input. The following timer clocks will increment the counter. When the first edge of opposite polarity occurs on TIO the counter stops, the TCF bit in TCSR is set and, if the TCIE is set, a compare interrupt is generated. The contents of the counter is loaded into the TCR and the user's program can read its value that represents the widths of the TIO pulse. On the first timer clock, following the next transition that occurs on TIO input pin, the counter is loaded with TLR value (if TRM is set) and the count is resumed. If TRM is cleared, the counter continues to be incremented on each timer clock, accumulating measurements results. This process is repeated until the timer is disabled (TE=0). If counter wraparound has occurred the TOF bit is set and, if the TOIE is set, an overflow interrupt is generated. In this mode TIO acts as a gating signal for the internal timer clock. The INV bit determines whether the counting is enabled when TIO is high (INV=0) or when TIO is low (INV=1).

15.4.2.2 Period measurement mode (mode 5)

This mode is defined by TC3-TC0 equal 0101. In this mode the counter is cleared after TE is set and, after the first appropriate transition (as determined by the INV bit) occurring on TIO input pin, it is loaded with the TLR value on the first timer clock derived either from the DSP clock divided by two (CLK/2) or from the prescaled clock input. The following timer clocks will increment the counter. On each following transition of the same polarity that occurs on TIO the TCF bit in TCSR is set and, if the TCIE is set, a compare interrupt is generated. The contents of the counter is loaded in the TCR and the user's program can read its value and the user's program can read the TCR to determine the distance between TIO edges. On the next timer clock the counter is loaded with TLR value (if TRM is set) and the count is resumed. If TRM is cleared, the counter continues to be incremented on each timer clock, accumulating measurements results. This process is repeated until the timer is disabled (TE=0). If counter wraparound has occurred the TOF bit is set and, if the TOIE is set, an overflow interrupt is generated. The INV bit determines whether the period is measured between consecutive 0 to 1 transitions of TIO (INV=0) or between consecutive 1 to 0 transitions of TIO (INV=1).

15.4.2.3 Capture mode (mode 6)

This mode is defined by TC3-TC0 equal 0110. In this mode the counter is cleared after TE is set and loaded with the TLR value on the first timer clock derived either from the DSP clock divided by two (CLK/2) or from the prescaled clock input. The following timer clocks will increment the counter. If counter wraparound has occurred the TOF bit is set and, if the TOIE is set, an overflow interrupt is generated. At the first transition of external clock the TCF bit in TCSR is set and, if the TCIE is set, a compare interrupt is generated. The contents of the counter is loaded into the TCR and the user's program can read its value that represents the delay of the leading detected edge in relation to the setting of the TE bit. The counting is stopped, when timer is disabled. The INV bit determines whether the period is measured between the setting of TE bit and the transitions of TIO from 0 to 1 (INV=0) or from 1 to 0 (INV=1).

15.4.2.4 Measurement modes exactness

Since the measurement modes use the internal clock to increment the counter, but use the external signal for gating the count, synchronization is needed. The synchronization process may effect the measurement exactness up to a single selected (internal or prescaled) clock cycle.

15.4.3 PWM Modes

15.4.3.1 Pulse Width Modulation mode, internal clock, output toggle enable (mode 7)

This mode is defined by TC3-TC0 equal 0111. In this mode the counter is cleared after TE is set and loaded with the TLR value on the first timer clock derived either from the DSP clock divided by two (CLK/2) or from the prescaled clock input. The following timer clocks will increment the counter. When the counter matches the value of the TCPR, the TIO output pin will be toggled, the TCF bit in TCSR is set and, if the TCIE is set, a compare interrupt is generated and the count is continued. When counter wraparound has occurred the TIO output pin will be toggled, the TOF bit in TCSR is set and, if the TOIE is set, an overflow interrupt is generated. At the next timer clock the counter is loaded with TLR value (if TRM is set) and the count is resumed. If TRM is cleared, the counter continues to be incremented on each timer clock. This process is repeated until the timer is disabled (TE=0). The TIO polarity is determined by the INV bit. On the first transaction the TIO output will be set if INV=0 or cleared if INV=1. The counter contents can be read at any time by reading the TCR register.

The value in the TLR determines the output period ($FFFF - TLR). The value in the TCPR determines the duty cycle of the output signal (TLR - TCPR vs. $FFFF - TCPR). Therefore, the ensure correct functionality, the values in TLR and TCPR should not be the same.

Note: After TE=1 the TIO pin output value is set equal to INV bit, to guarantee the correct first pin transition.

15.4.4 Watchdog Modes

15.4.4.1 Watchdog mode, internal clock, output pulse enable (mode 9)

This mode is defined by TC3-TC0 equal 1001. In this mode the counter is cleared after TE is set and loaded with the TLR value on the first timer clock derived either from the DSP clock divided by two (CLK/2) or from the prescaled clock input. The following timer clocks will increment the counter. When the counter matches the value of the TCPR, the TCF bit in TCSR is set and, if the TCIE is set, a compare interrupt is generated and the count is continued. At the next timer clock the counter is loaded with TLR value (if TRM is set) and the count is resumed. If TRM is cleared, the counter continues to be incremented on each timer clock. This process is repeated until the timer is disabled (TE=0). The counter will be reloaded whenever the TLR is written with a new value while the Timer Enable bit is set (TE = 1). When counter wraparound has occurred the TOF bit in TCSR is set and, if the TOIE is set, an overflow interrupt is generated. At the same time a pulse will be output on the TIO pin with the width equal to timer clock period. The pulse polarity is determined by the INV bit. The counter contents can be read at any time by reading the TCR register.

Note: In this mode, the internal hardware will preserve the TIO value and direction for additional 2.5 internal clock cycles after reset was activated. This will ensure a valid length reset when the TIO is used as input to the RESET pin.

15.4.4.2 Watchdog mode, internal clock, output toggle enable (mode 10)

This mode is defined by TC3-TC0 equal 1010. In this mode the counter is cleared after TE is set and loaded with the TLR value on the first timer clock derived either from the DSP clock divided by two (CLK/2) or from the prescaled clock input. The following timer clocks will increment the counter. When the counter matches the value of the TCPR, the TCF bit in TCSR is set and, if the TCIE is set, a compare interrupt is generated and the count is continued. At the next timer clock the counter is loaded with TLR value (if TRM is set) and the count is resumed. If TRM is cleared, the counter continues to be incremented on each timer clock. This process is repeated until the timer is disabled (TE=0). The counter will be reloaded whenever the TLR is written with a new value while the Timer Enable bit is set (TE = 1). When counter wraparound has occurred the TIO output pin will be toggled, the TOF bit in TCSR is set and, if the TOIE is set, an overflow interrupt is generated. The TIO polarity is determined by the INV bit. On the first transaction the TIO output will be set if INV=0 or cleared if INV=1. The counter contents can be read at any time by reading the TCR register.

Note: After TE=1 the TIO pin output value is set equal to INV bit, to guarantee the correct first pin transition.

Note: In this mode, the internal hardware will preserve the TIO value and direction for additional 2.5 internal clock cycles after reset was activated. This will ensure a valid length reset when the TIO is used as input to the RESET pin.

15.4.5 Reserved Modes

Modes 8,11,12,13,14 and 15 are reserved.

15.4.6 Special Cases

15.4.6.1 Timer behavior during WAIT and STOP

During the execution of the WAIT instruction the timer clocks are active thus the timer activity will continue undisturbed. Upon reaching the final event, if the timer interrupt is enabled, an interrupt will be generated, and the processor will leave the wait state and service the interrupt.

During the execution of the STOP instruction the timer clocks are disabled, the timer activity is stopped and the TIO pins are disconnected. If, for example, the TIO pin is used as input the changes that occur while in STOP will be ignored. In order to ensure correct behavior the timer should be disabled before executing the STOP instruction.

Chapter 16
Serial Audio Codec Port (SAP)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	03/18/02	Anand Gupta	Scan related pins deleted, scan_en replaced by ipt_test_clk_se.
0.2	04/18/02	Anand Gupta	All bidirectional pins removed. They have been replaced by dedicated inputs and dedicated outputs. Changed the names of scan_en and scan_mode
0.3	09/06/02	Anand Gupta, Shannon Osgood	Updated per DDTS# DSPhl15233. Replaced BCARA and BCBRA reset values with $05A7 and $FF4E.
0.4	09/09/02	Anand Gupta	Updated pin list and also the block, connectivity, and pin diagrams.
0.5	11/09/02	Anand Gupta	Modified pinlist, Table 16.1, SAP Pin diagram 16.2, SAP connectivity diagram 16.3
0.6	05/07/03		Updated for LTE specification release.

16.1 SAP changes from RSI

The Serial Audio Codec Port (SAP) is basically a RSI with the addition of a Binary Rate Multiplier (BRM) and a General Purpose Timer.

16.1.1 Binary Rate Multiplier (BRM)

The Binary Rate Multiplier is a Clock divider which generates a BRM clock from the reference clock input. The BRM, when used with the SAP dividers described later, allows the generation of standard codec master clock frequencies. There is a mux which selects between the DSP clock and the BRM clock.
For more details see:

- Figure 16-1. SAP Block Diagram
- Table 16-12. CRCA Description, BRM Bit 8

16.1.2 General Purpose Timer

The General Purpose Timer counts down with the DSP clock divided by 2048. It has a Timer Count Load Register (TCLR) which is used to load the counter upon timer enable or when the counter rolls over. Upon a roll over, an interrupt to the DSP is issued. For more details see:

- Table 16-10. CRBA Description, TCE Bit 2
- Paragraph 16.6.3 SAP Exceptions, list: 4.
- Paragraph 16.6.4.5 Timer Operating Mode

16.2 Introduction

The Serial Audio Codec Port (SAP) provides a full-duplex serial port for serial communication with Integrated Audio DC. The SAP consists of independent transmitter and receiver sections, a SAP binary rate multiplier, and a general purpose timer (which may use for debugging).

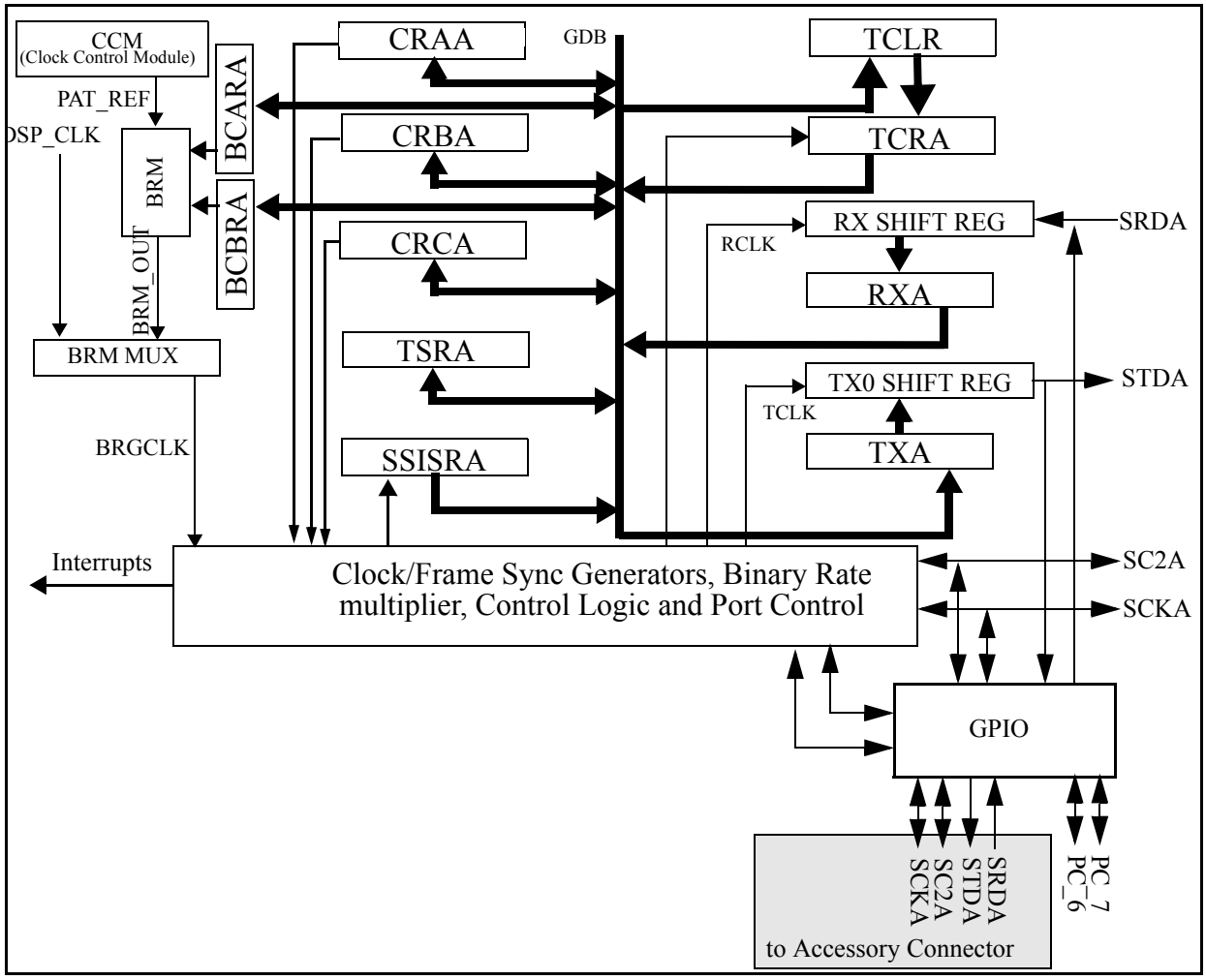


Figure 16-1. SAP Block Diagram

16.3 SAP Module Pin List

Table 16-1 is a list of all the SAP module pins.

Table 16-1. SAP Module Pin List

Pin Name	Direction	Description
Core Access Signals		
mcab_wr[6:0]	Input	Core Write Address Bus
mcab_rd[6:0]	Input	Core Read Address Bus
mcsel_wr	Input	Core select write
mcsel_rd	Input	X or Y I/O Read Select. Connected at the module instantiation level to one of the MCSELX_RD MCSELY_RD
mcdis0	Input	Core pipeline stall indication (used in SPMBIF only. hidden for the user)

Table 16-1. SAP Module Pin List (Continued)

Pin Name	Direction	Description
mcdis1	Input	Core pipeline stall indication (used in SPMBIF only. hidden for the user)
gdb_wr[15:0]	Input	Core global write data bus
gdb_rd[15:0]	Output	Core global read data bus
DMA Channel Access Signals		
mdab[6:0]	Input	DMA Address Bus
mdsel	Input	DMA-XIO Select. Connected at the module instantiation level to one of the four following PMB signals: MDSELX, MDSELY.
mdrd	Input	DMA Read
mddb_wr[15:0]	Input	DMA Write Data bus
mddb_rd[15:0]	Output	DMA Read Data bus
sap_te_mdrq_b	Output	DMA request when transmitter empty
sap_rf_mdrq_b	Output	DMA request when receiver full
Interrupt Request Signals		
pivrd	Input	Peripheral Interrupt vector read signal
pivsse	Input	Interrupt Vector Source Select. Connected at the module instantiation level to one of the 12 interrupt Vector Source Select line pivsse[11:0].
pivack	Input	Peripheral Module Interrupt Acknowledge
vab_rd[7:1]	Output	Interrupt Vector Bus
mirq_b	Output	Peripheral Module Interrupt Request. Connected at the module instantiation level to one of the 12 interrupt request line mirqn[11:0]. Active low.
Scan Signals		
ipt_scan_mode	Input	Scan Mode
ipt_test_clk_se	Input	Scan Enable
spmbif_scan_tristate_enable	Input	Scan tristate enable
Other Signals		
gclkw	Input	Raw clk from DSP clk-generator
gclkw_b	Input	Raw inverted clk from DSP clk-generator
pires	Input	Peripheral Module's Software reset.
res	Input	Peripheral Module's Hardware reset.

Table 16-1. SAP Module Pin List (Continued)

Pin Name	Direction	Description
bcstop	Input	This signal indicates that the core is in stop mode. It is asserted four cycles before the gclkw goes idle.
Signals for SAP ASIC		
ccm_sap_ref_clk	Input	Reference clock from CCM
sap_stda_o	Output	Output data pin STDA
sap_srda_o	Output	Output data pin SRDA
sap_scka_o	Output	Output data pin SCKA
sap_sc2a_o	Output	Output data pin SC2A
sap_sc1a_o	Output	Output data pin SC1A/Flag1
sap_sc0a_o	Output	Output data pin SC0A/Flag0
sap_stda_i	Input	Input data pin STDA
sap_srda_i	Input	Input data pin SRDA
sap_scka_i	Input	Input data pin SCKA
sap_sc2a_i	Input	Input data pin SC2A
sap_sc1a_i	Input	Input data pin SC1A
sap_sc0a_i	Input	Input data pin SC0A
sap_stda_oe	Output	Port control output enable for STDA
sap_srda_oe	Output	Port control output enable for SRDA
sap_scka_oe	Output	Port control output enable for SCKA
sap_sc2a_oe	Output	Port control output enable for SC2A
sap_sc1a_oe	Output	Port control output enable for SC1A/Flag1
sap_sc0a_oe	Output	Port control output enable for SC0A/Flag0

16.3.1 SAP Pin Diagram

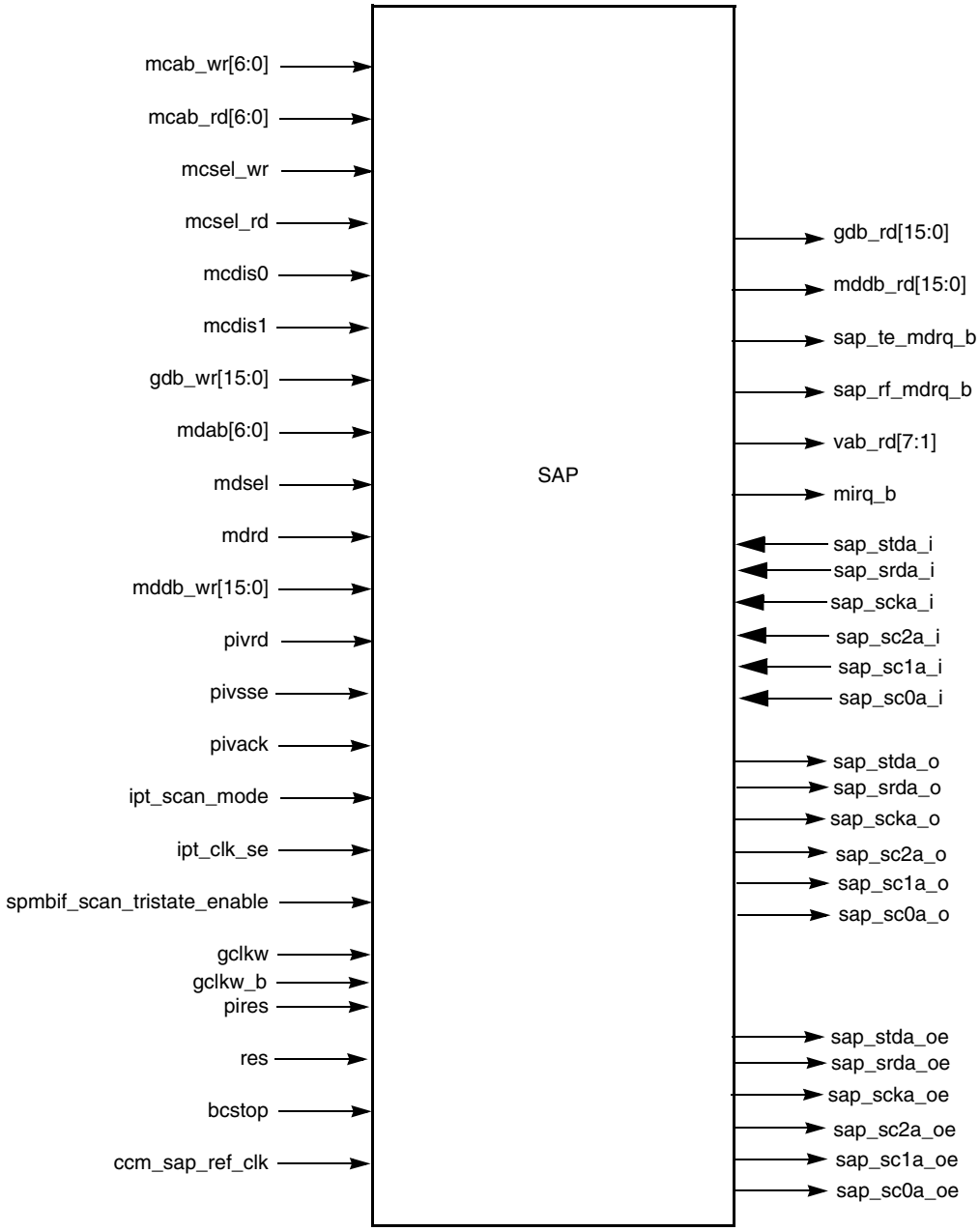


Figure 16-2. SAP Pin Diagram

16.3.1.1 SAP Connectivity Diagram

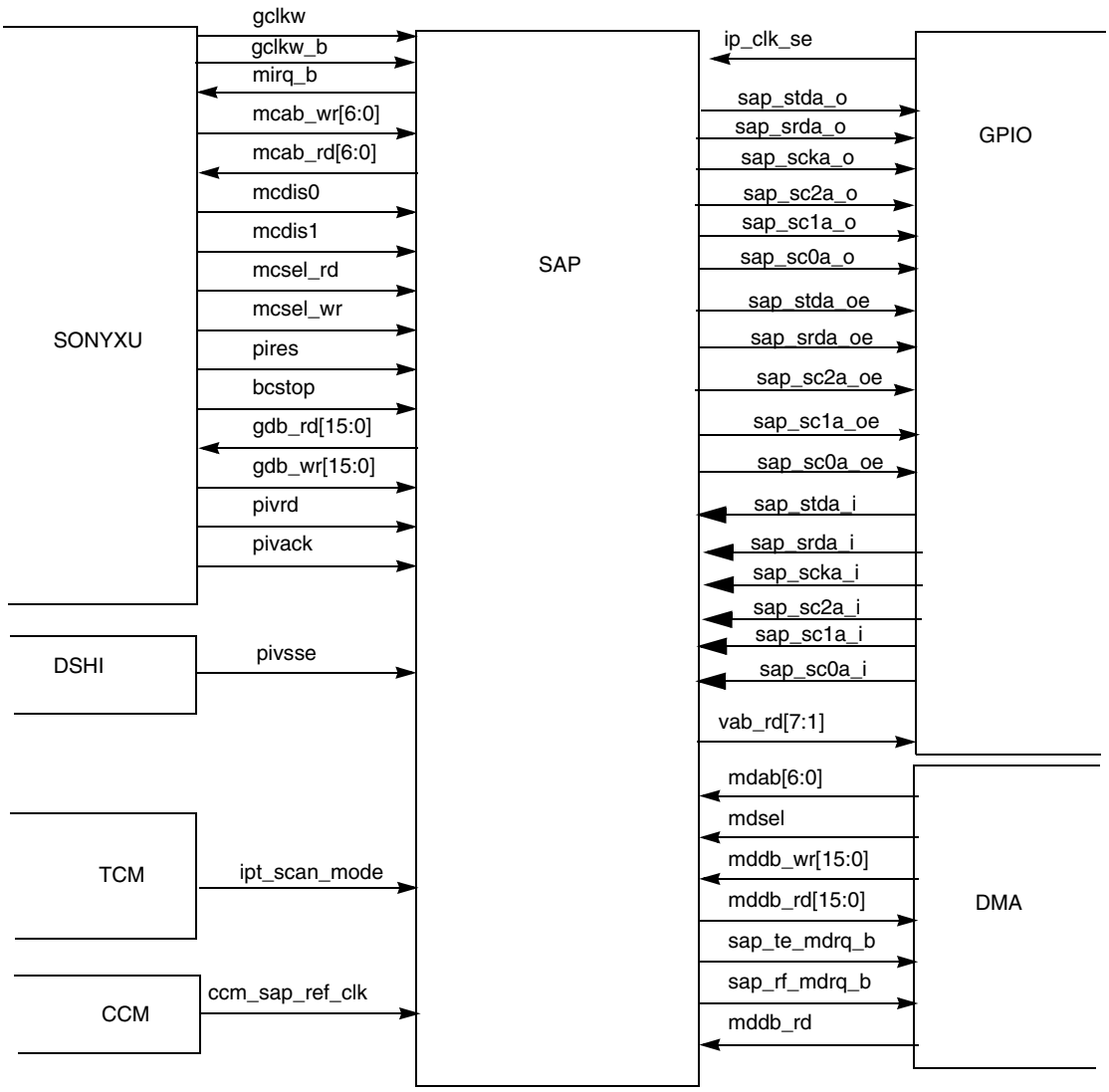


Figure 16-3. SAP Connectivity Diagram

16.4 SAP Data and Control Pins

16.4.1 Serial Transmit Data Pin (STDA)

STDA is used for transmitting data from TXA serial transmit shift register. STDA is an output when data is being transmitted from TXA shift register. With an internally generated bit clock, the STDA pin becomes high impedance after the last data bit has been transmitted for a full clock period, assuming another data word does not follow immediately. If a data word follows immediately, there will not be a high-impedance interval. STDA may be programmed as a general-purpose pin.

16.4.2 Serial Receive Data Pin (SRDA)

SRDA receives serial data and transfers the data to the SAP receive shift register. SRDA may be programmed as a general-purpose I/O pin when the SAP SRDA function is not being used.

16.4.3 Serial Clock (SCKA)

SCKA is a bidirectional pin providing the serial bit rate clock for the SAP interface. The SCKA is a clock input or output used by the transmitter and receiver in synchronous modes or by the transmitter only in asynchronous modes (see Table 16-2, SAP Clock Sources).
If the SCKA pin configured as output pin, the clock output source come from the Baud Rate Generator which derived from the Binary Rate Multiplier or the DSP_CLK based on BRM bit in CRCA. If the BRM bit is set (BRM=1), then the Internal Bit Clock frequency will be:

$$f_{INT\_BIT\_CLK}=f_{BRM\_OUT} / [(2\text{-}PSR)^3 *2*(PM+1)]$$

where PM is the Prescaler Divide Ratio (see Table 16-9, CRAA Description, PM[7:0]) and PSR is the value of the PSR bit (see Table 16-9, CRAA Description, PSR Bit 15).

If the BRM bit is cleared (BRM=0), then the Internal Bit Clock frequency will be:

$$f_{INT\_BIT\_CLK}=f_{DSP\_CLK} /[(2\text{-}PSR)^3 *2*(PM+1)]$$

In all cases, care must be taken to have always the Internal Bit Clock frequency lower than one third of the DSP_CLK frequency (see the note in Table 16-9, CRAA Description, PSR Bit 15).

The Frame Sync Frequency is based on the Internal Bit Clock frequency and will be:

$$f_{FRAME\_SYNC\_CLK}=f_{INT\_BIT\_CLK} / [WL*(DC+1)]$$

NOTE:

Setting DC=0 in network mode is a special case - on demand mode. There is no meaning for the FRAME_SYNC_CLK period in this mode, since the frame sync is not periodic.

SCKA may be programmed as a general-purpose I/O pin when the SAP SCKA function is not being used.

NOTE:

Although an external serial clock can be independent of and asynchronous to the DSP system clock, it must exceed the minimum clock cycle time of 6T (i.e., the system clock frequency must be at least three times the external SAP clock frequency). The SAP needs at least three DSP phases (DSP phase = T) inside each half of the serial clock. The same restriction is applicable to the Baud rate generator output clock when the Binary Rate Multiplier is select as the clock source.

16.4.4 Serial Control Pin (SC0A)

The function of this pin is determined by the selection of either synchronous or asynchronous mode (see Table 16-11, Mode and Pin Definition Table). For asynchronous mode, this pin will be used for the receive clock I/O. If an asynchronous mode was selected and the pin configured as output pin, its source comes from the Baud Rate Generator which is derived from the Binary Rate Multiplier or the DSP_CLK determined by the BRM bit in CRCA.
For synchronous mode, this pin is used for serial flag I/O. A typical application of flag I/O would be multiple device selection for addressing in codec systems. If this pin is configured as serial flag pin its direction is determined by the SCD0 bit in the CRCA. When configured as an output, this pin will be either

serial output flag 0, based on control bit OF0 in CRBA, or a receive shift register clock output. When configured as an input, this pin may be used either as serial input flag 0, which will control status bit IF0 in the SSISRA, or as a receive shift register clock input.
SC0A may be programmed as a general-purpose I/O pin when the SAP SC0A function is not being used.

16.4.5 Serial Control Pin (SC1A)

The function of this pin is determined by the selection of either synchronous or asynchronous mode (see Table 16-11, Mode and Pin Definition Table). In asynchronous mode (such as a single codec with asynchronous transmit and receive), this pin is the receiver frame sync I/O. For synchronous mode, this pin is used for serial flag I/O and operates like the previously described SC0A. SC0A and SC1A are independent serial I/O flags but may be used together for multiple serial device selection. SC0A and SC1A can be used unencoded to select up to two codecs or may be decoded externally to select up to four codecs. If this pin is configured as serial flag pin its direction is determined by the SCD1 bit in the CRCA. When configured as an output, this pin will be either a serial output flag, based on control bit OF1, or it will make the receive frame sync signal available. When configured as an input, this pin may be used as a serial input flag, which will control status bit IF1 in the SAP status register, or as a receive frame sync from an external source.
SC1A may be programmed as a general-purpose I/O pin when the SAP SC1A function is not being used.

Table 16-2. SAP Clock Sources

SYN	SCKD	SCD0	R Clock Source	RXA Clock Out	T Clock Source	TXA Clock Out
Asynchronous						
0	0	0	EXT, SC0A	–	EXT, SCKA	–
0	0	1	INT	SC0A	EXT, SCKA	–
0	1	0	EXT, SC0A	–	INT	SCKA
0	1	1	INT	SC0A	INT	SCKA
Synchronous						
1	0	d.c.	EXT, SCKA	–	EXT, SCKA	–
1	1	d.c.	INT	SCKA	INT	SCKA

16.4.6 Serial Control Pin (SC2A)

This pin is used for frame sync I/O. SC2A is the frame sync for both the transmitter and receiver in synchronous mode and for the transmitter only in asynchronous mode (see Table 16-11, Mode and Pin Definition Table). The direction of this pin is determined by the SCD2 bit in CRCA. When configured as an output, this pin is the internally generated frame sync signal. When configured as an input, this pin receives an external frame sync signal for the transmitter (and the receiver in synchronous operation).
SC2A may be programmed as a general-purpose I/O pin when the SAP SC2A function is not being used.

16.5 SAP Programming Model

The SAP can be viewed as three control registers, one status register, a transmit data register, a receive data register, special-purpose time slot register and three port registers, control, direction and data. The following paragraphs give detailed descriptions and operations of each of the bits in the SAP registers.

16.5.1 Register Summary

The Register Summary lists all registers of the module by ascending address. The absolute address of each register is given, and the value of each bit for reads and writes is given using the conventions in the Legend.

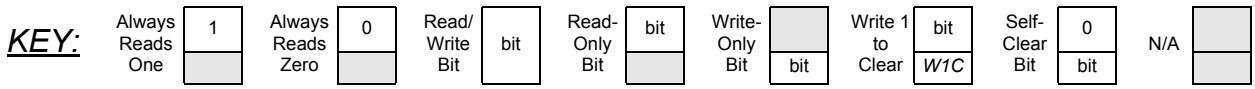


Table 16-3. Serial Audio Port Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCBRA (X:$FFC2)	R	BCB[15:0]															
	W																
BCARA (X:$FFC3)	R	BCA[15:0]															
	W																
TCRA (X:$FFC4)	R	CV[15:0]															
	W																
TCLR (X:$FFC5)	R	LV[15:0]															
	W																
CRAA (X:$FFC6)	R	PSR	WL[1:0]		DC[4:0]					PM[7:0]							
	W																
CRBA (X:$FFC7)	R	REIE	TEIE	RLIE	TLIE	RIE	TIE	RE	TE	0	0	0	0	0	TCE	OF1	OF0
	W																
CRCA (X:$FFC8)	R	FSP	FSR	FSL[1:0]		0	0	0	BRM	SHFD	CKP	SCKD	SCD[2:0]			MOD	SYN
	W																
SSISRA (X:$FFC9)	R	0	0	0	0	0	0	0	0	RDF	TDE	ROE	TUE	RFS	TFS	IF[1:0]	
	W																
RXA (X:$FFCA)	R	RXA[15:0]															
	W																
TSRA (X:$FFCB)	R																
	W	TSRA[15:0]															
TXA (X:$FFCC)	R																
	W	TXA[15:0]															
PDRA (X:$FFCD)	R	0	0	0	0	0	0	0	0	0	0	PD[5:0]					
	W																
PRRA (X:$FFCE)	R	0	0	0	0	0	0	0	0	0	0	PDC[5:0]					
	W																
PCRA (X:$FFCF)	R	0	0	0	0	0	0	0	0	PEN	0	PC[5:0]					
	W																

16.5.2 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various SAP registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

BCBRA Binary Constant B Register **Addr X:$FFC2**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	BCB[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	1	1	1	1	1	1	1	1	0	1	0	0	1	1	1	0

Table 16-4. BCBRA Description

Name	Description	Settings
BCB[15:0] Bits15-0	**Binary Constant B** — BCBRA is a 16 bit read/write register which determines in conjunction with BCARA the Binary Rate Multiplication factor. The BCBRA is initialized by hardware or software reset to the value of $FF4E.	N/A

Table specifies the values which must be written into BCARA and BCBRA in order to generate the required clock rates for different system clocks

Table 16-5. Recommended Programming of BCARA and BCBRA

PAT_REF (MHz)	BCARA	BCBRA	BRM_OUT (MHz)
13	$05A7	$FF4E	12.288
16.8	$01F3	$FFE6	16.384
19.44	$0341	$FE82	16.384

The formula for BRM_OUT is:

$$\text{BRM\_OUT} = \text{PAT\_REF} \times \frac{\text{BCARA} + 0.5\ (2\text{'s complement of BCBRA})}{\text{BCARA} + (2\text{'s complement of BCBRA})}$$

To determine BCARA and BCBRA, express BRM_OUT/PAT_REF as a ratio of two integers (X/Y) which have been reduced to their least common terms. Then,

BCARA = 2*X - Y

BCBRA = 2's complement of (2*(Y - X))

Example: PAT_REF=13MHz, BRM_OUT=12.228MHz. BRM_OUT/PAT_REF reduces to 1536/1625. Then,

BCARA = 2*1536 - 1625 = 1447 = $05A7

BCBRA = 2's comp (2*(1625-1536)) = 2's comp($b2) = $FF4E

If the BRM bit is set (BRM=1), then the Internal Bit Clock frequency will be:

$f_{INT\_BIT\_CLK}=f_{BRM\_OUT} / [(2\text{-}PSR)^3 *2*(PM+1)]$

where PM is the Prescaler Divide Ratio (see Table 16-9, CRAA Description, PM[7:0]) and PSR is the value of the PSR bit (see Table 16-9, CRAA Description, PSR Bit 15).

If the BRM bit is cleared (BRM=0), then the Internal Bit Clock frequency will be:

$f_{INT\_BIT\_CLK}=f_{DSP\_CLK} /[(2\text{-}PSR)^3 *2*(PM+1)]$

In all cases, care must be taken to have always the Internal Bit Clock frequency lower than one third of the DSP_CLK frequency (see the note in Table 16-9, CRAA Description, PSR Bit 15).

BCARA Binary Constant A Register **Addr X:$FFC3**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	BCA[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	1	0	1	1	0	1	0	0	1	1	1

Table 16-6. BCARA Description

Name	Description	Settings
BCA[15:0] Bits15-0	**Binary Constant A**— BCARA is a 16 bit read/write register which determines in conjunction with BCBRA the Binary Rate Multiplication factor. The BCARA is initialized by hardware or software reset to the value of $05A7	N/A

TCRA SAP Timer Counter Register **Addr X:$FFC4**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	CV[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-7. TCRA Description

Name	Description	Settings
CV[15:0] Bits15-0	**Count Value** — The TCRA is a 16 bit read-only register. The timer counter register (TCRA) is loaded with TCLR data (LV0-15 bits) and when the timer is enabled by setting TCE bit (of CRBA register) the counting down starts. Upon a roll over of the timer, the TCRA is reloaded with TCLR data and count down continues The TCRA bits are cleared by hardware reset	N/A

TCLR SAP Timer Counter Load Register **Addr X:$FFC5**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	LV[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-8. TCLR Description

Name	Description	Settings
LV[15:0] Bits15-0	**Load Value** — TCLR is a 16 bit read-write register.The timer count register (TCRA) is loaded with TCLR data (LV0-15 bits) and when counter is enabled by setting TCE bit the counting down starts. All TCLR bits are cleared by hardware and software reset.	N/A

CRAA SAP Control Register A **Addr X:$FFC6**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PSR	WL[1:0]		DC[4:0]					PM[7:0]							
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-9. CRAA Description

Name	Description	Settings
PSR Bit 15	**Prescaler Range—**The PSR controls a fixed divide-by-eight prescaler in series with the variable prescaler. This bit is used to extend the range of the prescaler for those cases where a slower bit clock is desired. When PSR is set, the fixed prescaler is bypassed. When PSR is cleared, the fixed divide-by-eight prescaler is operational. The maximum internally generated bit clock frequency is BRGCLK/4 (BRGCLK is the input clock to the baud rate generator which can be the DSP_CLK or the BRM output), the minimum internally generated bit clock frequency is BRGCLK/2/8/256=BRGCLK/4096. Hardware and software reset clear PSR. **Note:** The combination PSR = 1 and PM7-PM0 = $00 is forbidden and may cause synchronization problems if used. **Note:** It is forbidden to program the TCLOCK or RCLOCK (see Figure 16-4) to be faster than one-third of the DSP_CLK because it may cause synchronization problems if used.	0 = Prescaler applied (default). 1 = No prescaler.
WL[1:0] Bits 14–13	**Word Length Control—**The WL1 and WL0 bits are used to select the length of the data words being transferred via the SAP. Word lengths of 8, 12 or 16 bits may be selected according to the assignment described in settings column. Hardware and software reset clear WL1 and WL0	**WL1** / **WL0** / **Number of Bits/Word** 0 / 0 / 8 (default) 0 / 1 / 12 1 / 0 / 16 1 / 1 / Reserved

Table 16-9. CRAA Description

Name	Description	Settings
DC[4:0] Bits 12–8	**Frame Rate Divider Control—**The DC4–DC0 bits control the divide ratio for the programmable frame rate dividers used to generate the frame clocks. In network mode, this ratio may be interpreted as the number of words per frame minus one. In normal mode, this ratio determines the word transfer rate. The divide ratio may range from 1 to 32 (DC=00000 to 11111) for normal mode and 2 to 32 (DC=00001 to 11111) for network mode. A divide ratio of one (DC=00000) in network mode is a special case (on demand mode). In normal mode, a divide ratio of one (DC=00000) provides continuous periodic data word transfers. A bit-length sync must be used in this case. Hardware and software reset clear DC4–DC0.	N/A
PM[7:0] Bits 7–0	**Prescale Modulus Select—**The PM7–PM0 bits specify the divide ratio of the prescale divider in the SAP clock generator. A divide ratio from 1 to 256 (PM=0 to $FF) may be selected. The bit clock output is available at the transmit clock (SCKA) and/or the receive clock (SC0A) pins of the DSP. The bit clock output is also available internally for use as the bit clock to shift the transmit and receive shift registers. Careful choice of the crystal oscillator frequency and the prescaler modulus will allow the industry-standard codec master clock frequencies of 2.048 MHz, 1.544 MHz, and 1.536 MHz to be generated. Hardware and software reset clear PM0–PM7.	N/A

CRAA is one of three 16-bit read/write control registers used to direct the operation of the SAP. The CRAA controls the SAP clock generator bit and frame sync rates, word length, and number of words per frame for the serial data. The CRAA control bits are described in Table 16-9. The SAP clock generator functional diagram is shown in Figure 16-4.

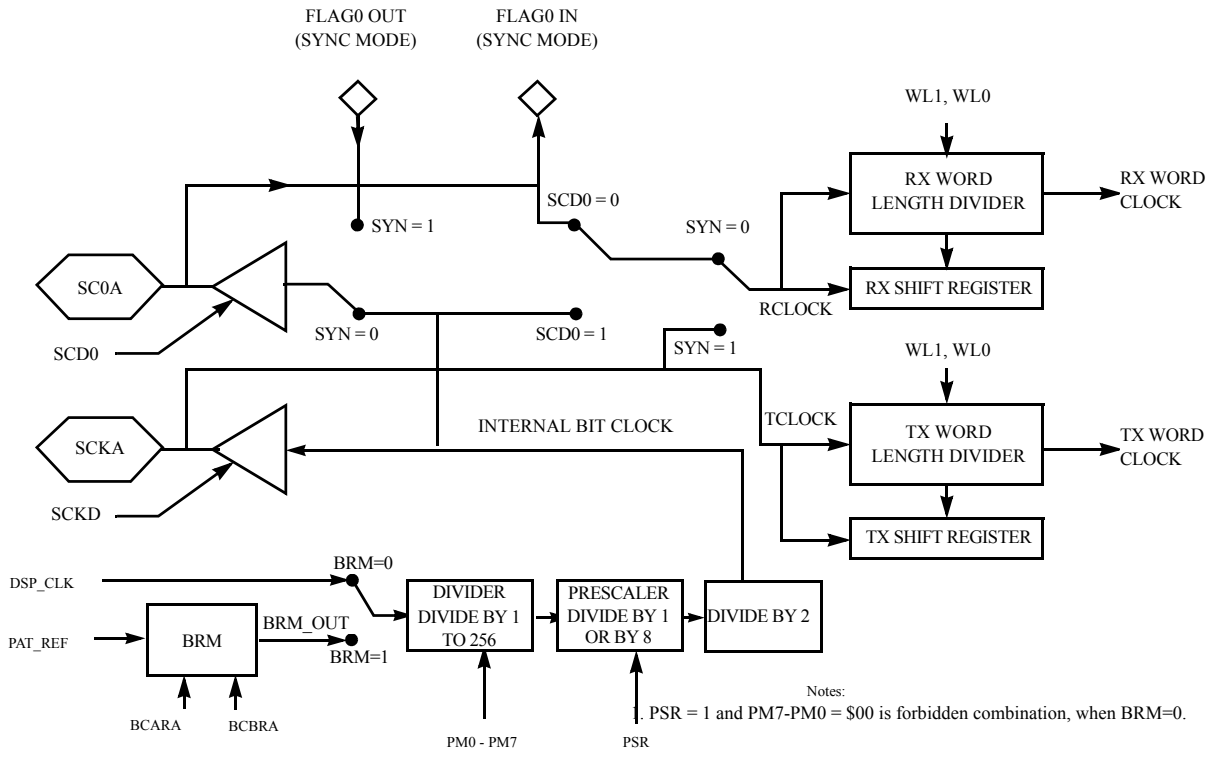


Figure 16-4. SAP Clock Generator Block Diagram

The CRBA is one of three 16-bit read/write control registers used to direct the operation of the SAP (see Table 16-10). CRBA controls the serial output flag, the SAP interrupts enables and transmitter and receiver enable. Hardware and software reset clear all the bits in the CRBA

CRBA SAP Control Register B **Addr X:$FFC7**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	REIE	TEIE	RLIE	TLIE	RIE	TIE	RE	TE						TCE	OF1	OF0
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-10. CRBA Description

Name	Description	Settings
REIE Bit 15	**Receive Exception Interrupt Enable—**When REIE is set, the DSP will be interrupted when both RDF and ROE in the SAP status register are set. When REIE is cleared, this interrupt is disabled. Reading the status register followed by reading the receive data register will clear ROE, thus clearing the pending interrupt. Hardware and software reset clear REIE.	0 = Interrupt disabled (default). 1 = Interrupt enabled.
TEIE Bit 14	**Transmit Exception Interrupt Enable—**When TEIE is set, the DSP will be interrupted when both TDE and TUE in the SAP status register are set. When TEIE is cleared, this interrupt is disabled. Reading the status register followed by writing to the transmitter data registers will clear TUE, thus clearing the pending interrupt. Hardware and software reset clear TEIE.	0 = Interrupt disabled (default). 1 = Interrupt enabled.
RLIE Bit 13	**Receive Last Slot Interrupt Enable—**RLIE enables an interrupt after the last slot of a frame ended in network mode only. When RLIE is set the DSP will be interrupted after the last slot in a frame ended. When RLIE is cleared the receive last slot interrupt is disabled. Hardware and software reset clear RLIE. RLIE is disabled when DC = $0 (on demand mode). The use of the Receive last slot interrupt is described in paragraph Table 16.6.3, SAP Exceptions.	0 = Interrupt disabled (default). 1 = Interrupt enabled.
TLIE Bit 12	**Transmit Last Slot Interrupt Enable—** LIE enables an interrupt at the beginning of last slot of a frame in network mode. When TLIE is set the DSP will be interrupted at the start of the last slot in a frame in network mode. When TLIE is cleared the transmit last slot interrupt is disabled. Hardware and software reset clear TLIE. TLIE is disabled when DC= $0 (on demand mode). The use of the transmit last slot interrupt is described in paragraph 16.6.3 SAP Exceptions.	0 = Interrupt disabled (default). 1 = Interrupt enabled.

Table 16-10. CRBA Description (Continued)

Name	Description	Settings
RIE Bit 11	**Receive Interrupt Enable—**When RIE is set, the DSP will be interrupted when RDF in the SAP status register is set. When RIE is cleared, this interrupt is disabled. Reading the receive data register will clear RDF, thus clearing the pending interrupt. Receive interrupts with exception have higher priority than normal receive data interrupts, therefore if exception occurs (ROE is set) and REIE is set the SAP will request an SAP receive data with exception interrupt from the interrupt controller. Hardware and software reset clear RIE.	0 = Interrupt disabled (default). 1 = Interrupt enabled.
TIE Bit 10	**Transmit Interrupt Enable—**The DSP will be interrupted when TIE and the TDE flag in the SAP status register are set. When TIE is cleared, this interrupt is disabled. Writing data to the data register of the transmitter or to TSRA will clear TDE, thus clearing the interrupt. Transmit interrupts with exception have higher priority than normal transmit data interrupts, therefore if exception occurs (TUE is set) and TEIE is set the SAP will request an SAP transmit data with exception interrupt from the interrupt controller. Hardware and software reset clear TIE.	0 = Interrupt disabled (default). 1 = Interrupt enabled.
RE Bit 9	**Receive Enable—**When RE is set, the receive portion of the SAP is enabled. When this bit is cleared, the receiver will be disabled by inhibiting data transfer into RXA. If data is being received while this bit is cleared, the remainder of the word will be shifted in and transferred to the SAP receive data register. RE must be set in the normal mode and on-demand mode to receive data. In network mode, the operation of clearing RE and setting it again will disable the receiver after reception of the current data word until the beginning of the next data frame. Hardware and software reset clear RE. **Note:** RE does not affect the generation of a frame sync.	0 = Receiver disabled (default). 1 = Receiver enabled.

Table 16-10. CRBA Description (Continued)

Name	Description	Settings
TE Bit 8	**Transmit Enable—** TE enables the transfer of data from TXA to the transmit shift register. When TE is set and a frame sync is detected, the transmit portion of the SAP is enabled for that frame. When TE is cleared, the transmitter will be disabled after completing transmission of data currently in the SAP transmit shift register. The STDA output is three-stated, and any data present in TXA will not be transmitted (i.e. data can be written to TXA with TE cleared; but data will not be transferred to the transmit shift register). The normal mode transmit enable sequence is to write data to the transmit data registers (or TSRA) before setting TE. The normal transmit disable sequence is to clear TE, TIE and TEIE after TDE equals one. In the network mode, the operation of clearing TE and setting it again will disable the transmitter after completing transmission of the current data word until the beginning of the next frame. During that time period, the STDA pin will remain in the high-impedance state. Hardware reset and software reset clear TE. The on-demand mode transmit enable sequence can be the same as the normal mode, or TE can be left enabled. **Note:** TE does not affect the generation of frame sync or output flags.	0 = Transmitter disabled (default). 1 = Transmitter enabled.
Bits 7-3	**Reserved Bits—**Bits 3-7 in the CRBA are reserved bits they read as zeros and must be written with zero for future compatibility.	N/A
TCE Bit 2	**Timer Count Enable—**TCE enables the 16 bits general purpose timer. Upon setting the TCE bit, the timer count register which is loaded with the TCLR data, counting down starts. When the TCE bit is cleared the TCRA timer is disabled. Hardware and software reset clear TCE bit.	0 = Timer disabled (default). 1 = Timer enabled.
OF1 Bit 1	**Serial Output Flag 1—**When the SAP is in the synchronous clock mode SC1A pin is configured as SAP flag 1. The serial control direction one bit (SCD1) when set, indicates that the SC1A pin is an output, then data present in OF1 will be written to SC1A at the beginning of the frame in normal mode or at the beginning of the next time slot in network mode. Hardware and software reset clear OF1. The normal sequence for setting output flags when transmitting data is: wait for TDE (TXA empty) to be set, first write the flags, and then write the transmit data to the TXA register. OF0 and OF1 are double buffered so that the flag states appear on the pins when the TXA data is transferred to the transmit shift register (i.e. the flags are synchronous with the data). **Note:** The optional serial output pins timing (SC0A and SC1A) are controlled by the frame timing and are not affected by TE or RE.	N/A

Table 16-10. CRBA Description (Continued)

Name	Description	Settings
OF0 Bit 0	**Serial Output Flag 0—**When the SAP is in the synchronous clock mode SC0A pin is configured as SAP flag 0. The serial control direction zero bit (SCD0) when set, indicates that the SC0A pin is an output, then data present in OF0 will be written to SC0A at the beginning of the frame in normal mode or at the beginning of the next time slot in network mode. Hardware and software reset clear OF0.	N/A

Table 16-11. Mode and Pin Definition Table

Control Bits			SAP Pins					
SYN	TE	RE	SC0A	SC1A	SC2A	SCKA	STDA	SRDA
0	0	0	U	U	U	U	U	U
0	0	1	RXC	FSR	U	U	U	RD
0	1	0	U	U	FST	TXC	TD	U
0	1	1	RXC	FSR	FST	TXC	TD	RD
1	0	0	F0/U	F1/U	FS	XC	U	U
1	0	1	F0/U	F1/U	FS	XC	U	RD
1	1	0	F0/U	F1/U	FS	XC	TD	U
1	1	1	F0/U	F1/U	FS	XC	TD	RD

- TXC - Transmitter Clock
- RXC - Receiver Clock
- XC - Transmitter/Receiver Clock (Synchronous Operation)
- FST - Transmitter Frame Sync
- FSR - Receiver Frame Sync
- FS - Transmitter/Receiver Frame Sync (Synchronous Operation)
- TD - Transmit Data
- RD - Receive Data
- F0/U - Flag 0 / Unused
- F1/U - Flag 1 / Unused
- U - Unused (may be used as GPIO pin)

NOTE:

A pin may be used as General Purpose IO pins (GPIO) if its corresponding bit in the Port Control Register (PCRA, bits PC5-PC0) is zero. See Table 16-19, PCRA Description for more details.

The CRCA is one of three 16-bit read/write control registers used to direct the operation of the SAP (see Table 16-12.). CRCA controls the SAP multifunction pins, SC2A, SC1A, and SC0A, which can be used as clock inputs or outputs, frame synchronization pins or serial I/O flag pins. The direction control bits for the serial control pins are in the SAP CRCA. Operating modes are also selected in this register. Hardware and software reset clear all the bits in the CRCA. The relationships between the SAP pins and some of the CRCA bits are summarized in Table 16-11, Mode and Pin Definition Table. The SAP CRCA bits are described in the following table.

CRCA SAP Control Register C **Addr X:$FFC8**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	FSP	FSR	FSL[1:0]					BRM	SHFD	CKP	SCKD	SCD2	SCD1	SCD0	MOD	SYN
TYPE:	rw	rw	rw	rw	r	r	r	rw	rw	rw	rw	rw	rw	wr	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-12. CRCA Description

Name	Description	Settings
FSP Bit 15	**Frame Sync Polarity—**FSP determines the polarity of the receive and transmit frame sync signals. When FSP is cleared the frame sync signal polarity is positive (i.e. the frame start is signed by the high level of the frame sync pin). When FSP is set the frame sync signal polarity is negative (i.e. the frame start is signed by the low level of the frame sync pin). Hardware reset and software reset clear FSP.	0 = Frame Sync Polarity Active-high (default). 1 = Frame Sync Polarity Active-low.
FSR Bit 14	**Frame Sync Relative Timing—**FSR determines the relative timing of the receive and transmit frame sync signal as referred to the serial data lines, for a word length frame sync only. When FSR is cleared the word length frame sync occurs together with the first bit of the data word of the first slot. When FSR is set the word length frame sync occurs one serial clock cycle earlier (i.e. together with the last bit of the previous data word). Hardware reset and software reset clear FSR.	0 = Frame Sync occurs with First bit of current frame (default). 1 = Frame Sync occurs with Last bit of previous frame.
FSL[1:0] Bits 13–12	**Frame Sync Length—T**hese bits select the length of frame sync to be generated or recognized. If FSL1 equals to zero and FSL0 equals to zero, a word-length frame sync is selected for both TXA and RXA that is the length of the data word defined by bits WL1 and WL0. If FSL1 equals to one and FSL0 equals to zero, a 1-bit clock period frame sync is selected for both TXA and RXA. When FSL0 equals to one, the TXA and RXA frame syncs are different lengths. The encoding of FSL1 and FSL0 is described in the settings column. Hardware reset and software reset clear FSL0 and FSL1	

FSL1	FSL0	Frame Sync Length
0	0	WL bit clock for both TXA/RXA
0	1	One-bit clock for TXA and WL bit clock for RXA
1	0	One-bit clock for both TXA/RXA
1	1	One-bit clock for RXA and WL bit clock for TXA

Table 16-12. CRCA Description (Continued)

Name	Description	Settings
Bits 11-9	**Reserved Bits—**Bits 9-11 in the CRCA are reserved bits they read as zeros and must be written with zero for future compatibility.	N/A
BRM Bit 8	**Binary Rate Multiplier—**BRM bit selects the Prescaler internal clock source. When the BRM bit is cleared the internal clock is derived from the DSP clock (DSP_CLK). When BRM bit is set, the clock source of the divider-prescaler chain (Figure 16-4) is the BRM_OUT clock which is created by the Binary Rate Multiplier from PAT_REF clock (e.g. 16.8MHz, 19.44Mhz, or 13MHz), and may be used by the Baud Rate Generator to create Codec master clocks of 256 kHz, 512 kHz, 2.048 Mhz, or 4.096 MHz. **Note:** The BRM bit can be set or cleared only when the SAP is in the individual RESET state (SAP I/O configured as GPIO). If SAP is not in the individual RESET state, changing the BRM bit may cause unexpected clocking.	0 = DSP_CLK (default). 1 = BRM_CLK.
SHFD Bit 7	**Shift Direction—**This bit causes the transmit shift register to shift data out MSB first when SHFD equals zero or LSB first when SHFD equals one. Received data is shifted in MSB first when SHFD equals zero or LSB first when SHFD equals zero. Hardware reset and software reset clear SHFD.	0 = Shift Data MSB first (default). 1 = Shift Data LSB first.
CKP Bit 6	**Clock Polarity—**The clock polarity bit controls on which bit clock edge data and frame sync are clocked out and latched in. If CKP is cleared the data and the frame sync are clocked out on the rising edge of the transmit bit clock and latched in on the falling edge of the receive bit clock. If CKP is set the falling edge of the transmit clock is used to clock the data out and frame sync and the rising edge of the receive clock is used to latch the data and frame sync in. Hardware reset and software reset clear CKP.	0 = Data and frame sync clocked out on the rising edge of the transmit bit clock and latched in on the falling edge of the receive bit clock 1 = Data and frame sync clocked out on the falling edge of the transmit bit clock and latched in on the rising edge of the receive bit clock
SCKD Bit 5	**Clock Source Direction—**SCKD selects the source of the clock signal used to clock the transmit shift register in the asynchronous mode and the transmit shift register and the receive shift register in the synchronous mode. In asynchronous mode when SCKD is set, the internal clock source becomes the bit clock for the transmit shift register and word length divider and is the output on the SCKA pin. When SCKD is cleared, the clock source is external; the internal clock generator is disconnected from the SCKA pin, and an external clock source may drive this pin. Hardware and software reset clear SCKD.	0 = External Clock (default). 1 = Internal Clock
SCD2 Bit 4	**Serial Control 2 Direction—**SCD2 controls the direction of the SC2A I/O pin. When SCD2 is cleared, SC2A is an input; when SCD2 is set, SC2A is an output. Hardware and software reset clear SCD2.	0 = SC2A pin is an Input (default). 1 = SC2A pin is an Output.

Table 16-12. CRCA Description (Continued)

Name	Description	Settings
SCD1 Bit 3	**Serial Control 1 Direction—**SCD1 controls the direction of the SC1A I/O pin. When SCD1 is cleared, SC1A is an input; when SCD1 is set, SC1A is an output. Hardware and software reset clear SCD1.	0 = SC1A pin is an Input (default). 1 = SC1A pin is an Output.
SCD0 Bit 2	**Serial Control 0 Direction—**SCD0 controls the direction of the SC0A I/O pin. When SCD0 is cleared, SC0A is an input; when SCD0 is set, SC0A is an output. Hardware and software reset clear SCD0.	0 = SC0A pin is an Input (default). 1 = SC0A pin is an Output.
MOD Bit 1	**SAP Mode Select—**The MOD bit selects the operational mode of the SAP. When the MOD bit is cleared, the normal mode is selected; when the MOD bit is set, the network mode is selected. In the normal mode, the frame rate divider determines the word transfer rate – one word is transferred per frame sync during the frame sync time slot. In network mode, multiple words (up to 32) can be transferred every frame. Hardware and software reset clear the MOD bit.	0 = Normal mode (default). 1 = Network mode.
SYN Bit 0	**Synchronous/Asynchronous Select—**SYN controls whether the receive and transmit functions of the SAP occur synchronously or asynchronously with respect to each other. When SYN is set, synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals. When SYN is cleared, asynchronous mode is chosen and separate clock and frame sync signals are used for the transmit and receive sections. Hardware reset and software reset clear SYN.	0 = Asynchronous mode (default). 1 = Synchronous mode.

The SSISRA (see Table 16-13) is a read-only status register used by the DSP to read the status and serial input flags of the SAP. When the SSISRA is read to the internal data bus, the register contents occupy the low-order byte of the data bus, and the remaining bits are read as zeros. The status bits are described in the following table.

SSISRA SAP Status Register **Addr X:$FFC9**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									RDF	TDE	ROE	TUE	RFS	TFS	IF1	IF0
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Table 16-13. SSISRA Description

Name	Description	Settings
RDF Bit 7	**Receive Data Register Full—**RDF is set when the contents of the receive shift register are transferred to the receive data register. RDF is cleared when the DSP reads the receive data register or cleared by hardware, software, SAP individual (SAP I/O configure as GPIO), or STOP reset. If RIE is set, a DSP receive data interrupt request will be issued when RDF is set.	0 = Receive Data Register not full (default). 1 = Receive Data Register Full
TDE Bit 6	**Transmit Data Register Empty—**This flag is set when the contents of the transmit data register is transferred to the transmit shift register; it is also set for a TSRA disabled time slot period in network mode (as if data were being transmitted after the TSRA was written). When set, TDE indicates that data should be written to the TXA register or to the time slot register (TSRA). TDE is cleared when the DSP writes to the transmit data register, or when the DSP writes to the TSRA to disable transmission of the next time slot. If TIE is set, a DSP transmit data interrupt request will be issued when TDE is set. Hardware, software, SAP individual (SAP I/O configure as GPIO), and STOP reset set TDE.	0 = Transmit Data Register not empty 1 = Transmit Data Register Empty (default).
ROE Bit 5	**Receiver Overrun Error Flag—**This flag is set when the serial receive shift register is filled and ready to transfer to the receiver data register (RXA) and RXA is already full (i.e., RDF=1). If REIE is set, a DSP receiver overrun error interrupt request will be issued when ROE is set. Hardware, software, SAP individual (SAP I/O configure as GPIO), and STOP reset clear ROE. ROE is also cleared by reading the SSISRA with ROE set, followed by reading the RXA.	0 = No receive error (default). 1 = Receiver overrun error has occurred.

Table 16-13. SSISRA Description (Continued)

Name	Description	Settings
TUE Bit 4	**Transmitter Underrun Error Flag—**TUE is set when the transmit shift registers is empty (no new data to be transmitted) and a transmit time slot occurs. When a transmit underrun error occurs, the previous data (which is still present in the TXA register that was not written) will be retransmitted. In the normal mode, there is only one transmit time slot per frame. In the network mode, there can be up to 32 transmit time slots per frame. If TEIE is set, a DSP transmit underrun error interrupt request will be issued when TUE is set. Hardware, software, SAP individual (SAP I/O configure as GPIO), and STOP reset clear TUE. TUE is also cleared by reading the SSISRA with TUE set, followed by writing to the transmit data registers or to TSRA.	0 = No transmit error (default). 1 = Transmitter underrun error has occurred.
RFS **Bit 3**	**Receive Frame Sync Flag—**When set, RFS indicates that a receive frame sync occurred during reception of the word in the serial receive data register. This indicates that the data word is from the first time slot in the frame. When RFS is clear and a word is received, it indicates (only in the network mode) that the frame sync did not occur during reception of that word. RFS is cleared by hardware, software, SAP individual (SAP I/O configure as GPIO), or STOP reset. RFS is valid only if the receiver is enabled (RE=1). **Note:** In normal mode, RFS will always read as a one when reading data because there is only one time slot per frame – the “frame sync” time slot.	0 = Receive Frame Sync did not occur (default). 1 = Receive Frame Sync occurred.
TFS Bit 2	**Transmit Frame Sync Flag—**When set, TFS indicates that a transmit frame sync occurred in the current time slot. TFS is set at the start of the first time slot in the frame and cleared during all other time slots. Data written to a transmit data register during the time slot when TFS is set will be transmitted (in network mode), if the transmitter is enabled, during the second time slot in the frame. TFS is useful in network mode to identify the start of a frame. TFS is cleared by hardware, software, SAP individual, or STOP reset. TFS is valid only if the transmitter is enabled (TE equal 1). **Note:** In normal mode, TFS will always be read as a one when transmitting data because there is only one time slot per frame – the “frame sync” time slot.	0 = Transmit Frame Sync did not occur (default). 1 = Transmit Frame Sync occurred.

Table 16-13. SSISRA Description (Continued)

Name	Description	Settings
IF1 Bit 1	**Serial Input Flag 1—**The SAP latches data present on the SC1A pin during reception of the first received bit after frame sync is detected. IF1 bit is updated with this data when the receiver shift register is transferred into the receive data register. The IF1 bit is enabled only when SC1A is programmed as SAP in the Port Control Register, SYN is set, and SCD1 is cleared, indicating that SC1A is an input flag and the synchronous mode is selected; otherwise, IF1 reads as a zero when it is not enabled. Hardware, software, SAP individual (SAP I/O configure as GPIO), and STOP reset clear IF1.	N/A
IF0 Bit 0	**Serial Input Flag 0—**The SAP latches data present on the SC0A pin during reception of the first received bit after frame sync is detected. IF0 bit is updated with this data when the receive shift register is transferred into the receive data register. The IF0 bit is enabled only when SC0A is programmed as SAP in the Port Control Register, SYN is set, and SCD0 is cleared, indicating that SC0A is an input flag and the synchronous mode is selected; otherwise, IF0 reads as a zero when it is not enabled. Hardware, software, SAP individual (SAP I/O configure as GPIO), and STOP reset clear IF0.	N/A

RXA SAP Receive Data Register **Addr X:$FFCA**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	RXA[15:0]															
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-14. RXA Description

Name	Description	Settings
RXA[15:0] Bits15-0	**SAP Receive Data** — RXA is a 16-bit read-only register that accepts data from the receive shift register as it becomes full. The data read will occupy the most significant portion of the receive data register. The unused bits (least significant portion) will read as zeros. The DSP is interrupted whenever RXA becomes full if the associated interrupt is enabled. RXA register is cleared by hardware reset.	N/A

TSRA SAP Time Slot Register **Addr X:$FFCB**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	TSRA[15:0]															
TYPE:	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
RESET	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 16-15. TSRA Description

Name	Description	Settings
TSRA[15:0] Bits15-0	**SAP Time Slot Data** — TSRA is effectively a null data register that is used when the data is not to be transmitted in the available transmit time slot. For the purposes of timing, TSRA is a write-only register that behaves like an alternative transmit data register, except that, rather than transmitting data, the transmit data pin is in the high-impedance state for that time slot	N/A

TXA SAP Transmit Data Register **Addr X:$FFCC**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	TXA[15:0]															
TYPE:	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-16. TXA Description

Name	Description	Settings
TXA[15:0] Bits15-0	**SAP Transmit Data** — TXA is 16-bit write-only registers. Data to be transmitted is written into this register and is automatically transferred to the transmit shift registers. The data written (8, 12 or 16 bits) should occupy the most significant portion of the TXA. The unused bits (least significant portion) of the TXA are don't care bits. The DSP is interrupted whenever TXA becomes empty if the transmit data register empty interrupt has been enabled. TXA register is cleared with hardware reset.	N/A

The read/write 16 bit Port Data Register is used to read or write data to/from SAP GPIO pins.

PDRA SAP Port Data Register **Addr X:$FFCD**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
											PD5	PD4	PD3	PD2	PD1	PD0
TYPE:	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-17. PDRA Description

Name	Description	Settings
PD[5-0] Bits 5-0	**Port Data Register bits—**Bits PD(5:0) are used to read or write data from/to the corresponding port pins if they are configured as GPIO (by PC(5:0) bits in PCRA). If a port pin [i] is configured as a GPIO input, then the corresponding PD[i] bit will reflect the value present on this pin. If a port pin [i] is configured as a GPIO output, then the value written into the corresponding PD[i] bit will be reflected on the this pin. Hardware and software reset clear all PDRA bits. The mapping between the PDRA bits and the actual SAP ports is: PD[5] - STDA PD[4] - SRDA PD[3] - SCKA PD[2] - SC2A PD[1] - SC1A PD[0] - SC0A	N/A

The read/write 16 bit Port Direction Register controls the direction of SAP GPIO pins. The following table describe the port pin configurations.

PRRA SAP Port Direction Register **Addr X:$FFCE**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
											PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
TYPE:	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-18. PRRA Description

Name	Description	Settings
PDC[5-0] Bits 5-0	**Port Direction Register bits—**When port pin[i] is configured as GPIO, PDC[i] controls the port pin direction. When PDC[i] is set, the GPIO port pin[i] is configured as output. When PDC[i] is cleared the GPIO port pin[i] is configured as input. Hardware and software reset clear all PRRA bits. The mapping between the PRRA bits and the actual SAP ports is: PDC[5] - STDA PDC[4] - SRDA PDC[3] - SCKA PDC[2] - SC2A PDC[1] - SC1A PDC[0] - SC0A	**PC[i]** / **PDC[i]** / **Port Pin[i] Function** 1 / X / SAP 0 / 0 / GPIO input 0 / 1 / GPIO output

The read/write 16 bit Port Control Register controls the functionality of SAP GPIO pins. Hardware and software reset clear all PC[i] bits and the PEN bit.

PCRA SAP Port Control Register **Addr X:$FFCF**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									PEN		PC5	PC4	PC3	PC2	PC1	PC0
TYPE:	r	r	r	r	r	r	r	r	rw	r	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-19. PCRA Description

Name	Description	Settings
PC[5-0] Bits 5-0	**Port Control bits—**each bit controls the functionality of the corresponding port pin. When a PC[i] bit is set, the corresponding port pin is configured as a SAP pin. When a PC[i] bit is cleared, the corresponding port pin is configured as GPIO pin. The mapping between the PCRA bits and the actual SAP ports is: PC[5] - STDA PC[4] - SRDA PC[3] - SCKA PC[2] - SC2A PC[1] - SC1A PC[0] - SC0A	0 = Corresponding Port Pin is a GPIO pin (default). 1 = Corresponding Port Pin is a SAP pin
PEN Bit 7	**Port Enable bit—** when cleared all SAP pins will be three-stated, ignoring any other settings, when set the pins will be active as defined by all other settings.	0 = SAP Pins three-stated (default). 1 = SAP Pins active

16.5.3 SAP Receive Shift Register

This 16-bit shift register receives the incoming data from the serial receive data pin. Data is shifted in by the selected (internal/external) bit clock when the associated frame sync I/O is asserted. Data is assumed to be received MSB first if SHFD equals zero and LSB first if SHFD equals one. Data is transferred to the SAP receive data register after 8, 12 or 16 serial clock cycles were counted, depending on the word-length control bits in the CRAA. The SAP Receive Shift Register is not user accessible.

16.5.4 SAP Transmit Shift Register

This 16-bit shift registers contain the data being transmitted. Data is shifted out to the serial transmit data pins by the selected (internal/external) bit clock when the associated frame sync I/O is asserted. The number of bits shifted out before the shift register is considered empty and may be written to again can be 8, 12 or 16 bits (determined by the word-length control bits in CRAA). The data to be transmitted occupies the most significant portion of the shift register. The unused portion of the register is ignored. Data is shifted out of this register MSB first if SHFD equals zero and LSB first if SHFD equals one. The SAP Transmit Shift Register is not user accessible.

16.6 Operating Modes

SAP operating mode are selected by the SAP control registers (CRAA, CRBA and CRCA). The main operating mode are described in the following paragraphs.

16.6.1 SAP After Reset

Hardware or software reset clears the port control register bits and the port direction control register bits, which configure all I/O as general-purpose input and all pins at High Impedance. The SAP is held in reset while all SAP pins are programmed as general-purpose I/O and is active only if at least one of the SAP I/O pins is programmed as SAP pin. This condition is referred to as “individual reset” throughout this specification. Software may perform a hardware reset of this block by configuring all SAP I/O pins as GPIO.

16.6.2 SAP Initialization

The correct way to initialize the SAP is as follows:

1. Hardware, software, SAP individual, or STOP reset
2. Program SAP control according to the desired functionality using SAP registers.
3. Program the appropriate GPIO registers for each multiplexed SAP pin configured as an input or an output:
    a) For outputs, the appropriate GPIO Port Data Direction Register bit should be set to 1, and the appropriate GPIO Port Configuration Register bits should be set to select the SAP pin functionality.
    b) For inputs, the appropriate GPIO Port Data Direction Register bit should be set to 0, and the appropriate GPIO Port Alternate Input Register bits should be set to select the SAP pin functionality.

Refer to the GPIO Chapter and the Neptune Multiplexing description in Chapter 23, “General Purpose Input/Output (GPIO),” for more information on which SAP pins are multiplexed, and how the GPIO Port registers should be programmed.

During program execution, PC5-PC0 bits in the GPIO port control register (PCRA) may be cleared causing the SAP to stop serial activity and enter the individual reset state. All status bits of the interface will be set to their reset state; however, the contents of CRAA, CRBA and CRCA are not affected. This procedure allows the DSP program to reset each interface separately from the other internal peripherals. During individual reset internal DMA accesses to the data registers of the SAP are not valid and data read will be unexpected.

To ensure proper operation of the interface, the DSP program must reset the SAP before changing any of its control registers (except for CRBA bits, TEIE, REIE, TLIE, RLIE, TIE, RIE, TE, RE, OF1 or OF0).

16.6.3 SAP Exceptions

The SAP can generate seven different exceptions (ordered from the highest to the lowest priority):

1. SAP Receive Data with Exception Status – occurs when the receive exception interrupt is enabled, the receive data register is full, and a receiver overrun error has occurred. ROE is cleared by first reading the SSISRA and then reading RXA.
2. SAP Receive Data – occurs when the receive interrupt is enabled, the receive data register is full, and no receive error conditions exist. Reading RXA clears the pending interrupt. This error-free interrupt can use a fast interrupt service routine for minimum overhead.
3. SAP Receive Last slot interrupt occurs after the last slot of the frame ended (in network mode only). The Receive Last slot interrupt may be used to reconfigure the DMA channels and reassignment of data memory pointers. Using the Receive last slot interrupt guarantees that the previous frame was serviced with the previous setting and the new frame will be serviced with the new setting without synchronization problems. Note that the maximum Receive last slot interrupt service time, should not exceed N-1 SAP bits service time (Where N is the number of bits in a slot).
4. SAP Transmit Data with Exception Status – occurs when the transmit exception interrupt is enabled, the transmit data register is empty, and a transmitter underrun error has occurred. TUE is cleared by first reading the SSISRA and then writing to the transmit data register, or to the TSRA to clear the pending interrupt.
5. SAP Transmit Last slot interrupt occurs at the start of the last slot of the frame in network mode. The Transmit Last slot interrupt may be used to reconfigure the DMA channels and reassignment of data memory pointers. Using the Transmit last slot interrupt guarantees that the previous frame was serviced with the previous setting and the new frame will be serviced with the new setting without synchronization problems. Note that the maximum Transmit last slot interrupt service time, should not exceed N-1 SAP bits service time (Where N is the number of bits in a slot).
6. SAP Transmit Data – occurs when the transmit interrupt is enabled, and the transmit data register is empty, and no transmitter error conditions exist. Writing to the TXA registers, or to the TSRA will clear this interrupt. This error-free interrupt may use a fast interrupt service routine for minimum overhead.
7. Timer Counter rolls-over interrupt – occurs when the timer counter reaches a zero value. Using the timer counter interrupt enables the user to service a specified number of events.

16.6.4 Operating Modes – Normal, Network, and On-Demand

The SAP has three basic operating modes and many data/operation formats. These modes can be programmed by several bits in the SAP control registers.
The data/operation formats available to the SAP are selected by setting or clearing control bits in the CRAA and CRCA. These control bits are DC4–DC0, WL1, WL0, MOD, SYN, FSL1, FSL0, FSR, FSP, CKP and SHFD.

16.6.4.1 Normal/Network/On-Demand Mode Selection

Selecting between the normal mode and network mode is accomplished by clearing or setting the MOD bit in the CRCA. For normal mode, the SAP functions with one data word of I/O per frame. For the network mode, 2 to 32 time slots per frame may be selected. During each frame, 0 to 32 data words of I/O may be received/transmitted. In either case, the transfers are periodic. The normal mode is typically used to

transfer data to/from a single device. Network mode is typically used in time division multiplexed (TDM) networks of codecs or DSPs with multiple words per frame.
Setting the MOD bit in the CRCA, as for network mode, and setting the frame rate divider to zero (DC=00000) selects the on demand mode. This special case will not generate a periodic frame sync. A frame sync pulse will be generated only when data is available to transmit. The frame sync signal indicates the first time slot in the frame. The on-demand mode requires that the transmit frame sync be internal (output) and the receive frame sync be external (input). Therefore, for simplex operation, the synchronous mode could be used; however, for full-duplex operation, the asynchronous mode must be used. Data transmission that is data driven is enabled by writing data into the TXA. Although the SAP is double buffered, only one word can be written to the TXA, even if the transmit shift register is empty. The receive and transmit interrupts function as usual using TDE and RDF; however, transmit underruns are impossible for on-demand transmission and are disabled. This mode is useful for interfacing to codecs requiring a continuous clock.

16.6.4.2 Synchronous/Asynchronous Operating Modes

The transmit and receive sections of this interface may be synchronous or asynchronous – i.e., the transmitter and receiver may use common clock and synchronization signals or they may have their own separate clock and sync signals (asynchronous operating mode). The SYN bit in CRCA selects synchronous or asynchronous operation. Since the SAP is designed to operate either synchronously or asynchronously, separate receive and transmit interrupts are provided.
When SYN equals zero, the SAP TXA and RXA clocks and frame sync sources are independent. If SYN equals one, the SAP TXA and RXA clocks and frame sync come from the same source (either external or internal).
Data clock and frame sync signals can be generated internally by the DSP or may be obtained from external sources. If internally generated, the SAP clock generator is used to derive bit clock and frame sync signals from the DSP internal system clock. The SAP clock generator consists of a selectable fixed prescaler and a programmable prescaler for bit rate clock generation and also a programmable frame-rate divider and a word-length divider for frame-rate sync-signal generation.

16.6.4.3 Frame Sync Selection

The transmitter and receiver can operate totally independent of each other. The transmitter can have either a bit-length-early or word-length-late frame-sync signal format, and the receiver can have the same or opposite format. The selection is made by programming FSL0 and FSL1 in the CRCA.

1. If FSL1 equals zero, the RXA frame sync is asserted during the entire data transfer period. This frame sync length is compatible with Motorola codecs, SPI serial peripherals, serial A/D and D/A converters, shift registers, and telecommunication PCM serial I/O.
2. If FSL1 equals one, the RXA frame sync pulses active for one bit clock immediately before the data transfer period. This frame sync length is compatible with Intel and National components, codecs, and telecommunication PCM serial I/O.

The ability to mix frame sync lengths is useful in configuring systems in which data is received from one type device (e.g., codec) and transmitted to a different type device.
FSL0 controls whether RXA and TXA have the same frame sync length. If FSL0 equals zero, RXA and TXA have the same frame sync length, which is selected by FSL1. If FSL0 equals one, RXA and TXA have different frame sync lengths, which are selected by FSL1. FSL0 is ignored when the SYN bit is set
FSR controls the relative timing of the word length frame sync as referred to the data word. When FSR is cleared the word length frame sync is generated (or expected) with the first bit of the data word. When FSR is set the word length frame sync is generated (or expected) with the last bit of the previous word. FSR is ignored when a bit length frame sync is selected.

The SAP does not support a bit length late frame sync. If the SAP receiver is configured for external frame sync, and is connected to a device that provides a bit length late frame sync, the SAP receiver may be configured for word length late frame sync to work around this issue, as long as the SAP divide ratio is not zero (DC[4:0] = 00000 with word length frame sync is not allowed).
FSP controls the polarity of the frame sync. When FSP is cleared the polarity of the frame sync is positive i.e. the frame sync signal is asserted high. When FSP is set the polarity of the frame sync is negative i.e. the frame sync is asserted low.
The SAP receiver looks for a receive frame sync leading edge (trailing edge if FSP is set) only when the previous frame is completed. If the frame sync goes high before the frame is completed (or before the last bit of the frame is received in the case of a bit frame sync or a word length frame sync with FSR set), the current frame sync will not be recognized, and the receiver will be internally disabled until the next frame sync. Frames do not have to be adjacent – i.e., a new frame sync does not have to immediately follow the previous frame. Gaps of arbitrary periods can occur between frames. The transmitter will be three-stated during these gaps.

16.6.4.4 Shift Direction Selection

Some data formats, such as those used by codecs, specify MSB first other data formats, such as the AES-EBU digital audio, specify LSB first. To interface with devices from both systems, the shift registers in the SAP are bidirectional. The MSB/LSB selection is made by programming SHFD in the CRCA.
If SHFD equals zero, data is shifted into the receive shift register MSB first and shifted out of the transmit shift register MSB first. If SHFD equals one, data is shifted into the receive shift register LSB first and shifted out of the transmit shift register LSB first.

16.6.4.5 Timer Operating Mode

Upon enabling the timer (setting TCE bit in CRBA), the timer counter is loaded from the TCLR register and count down begins. When the counter rolls over a timer counter roll over interrupt to the DSP is issued, the counter is reloaded from the TCLR register and count down continues. In STOP mode, the timer will freeze. When getting out of the STOP mode the timer will continue from the current value.

16.6.5 Flags

Two SAP pins (SC1A and SC0A) are available as serial I/O flags. Their operation can be controlled by SYN, SCD0 and SCD1 bits in the CRCA. The control bits (OF1 and OF0) and status bits (IF1 and IF0) are double buffered to/from SC1A and SC0A. Double buffering the flags keeps them in sync with TXA and RXA.

The flags are available in the synchronous mode only (SYN=1). Each flag can be separately programmed. When flag 0 is enabled, its direction is selected by SCD0, SCD0=1 as output and SCD0=0 as input. In the same way when flag1 is enabled, its direction is selected by SCD1, SCD1=1 as output and SCD1=0 as input.
When programmed as input, SC0A and SC1A value, respectively, are latched at the same time as the first bit is sampled of the receive data word. Since the input was latched, the signal on the input flag pin (SC0A and SC1A), can change without affecting the input flag until the first bit of the next receive data word. When received data word is latched by RXA, the latched values are then latched by the SSISRA IF0 and IF1 bits respectively and can be read by software.
When programed as output, SC0A and SC1A value, is driven by the value, from the CRBA OF0 and OF1 bits respectively, latched when the TXA is transferred to the transmit shift register. The value on SC0A or SC1A, will be stable from the same time the first bit of the transmit data word is transmitted until the first bit of the next transmit data word is transmitted. Software can change the CRBA OF0 and OF1 values and thus controlling the SC0A and SC1A pin values for each transmitted word.

16.7 Appendix:

- When pin direction of a SAP pin is changed i.e it is made GPIO output from GPIO input, values on pin and the corresponding bit of PDRA go to 0 state. This Behavior is acceptable for Neptune.
- Transmitter should be enabled (TE=1) atleast two cycles of serial clock before valid time slot.

Chapter 17
DSP Peripherals DMA (DMA)

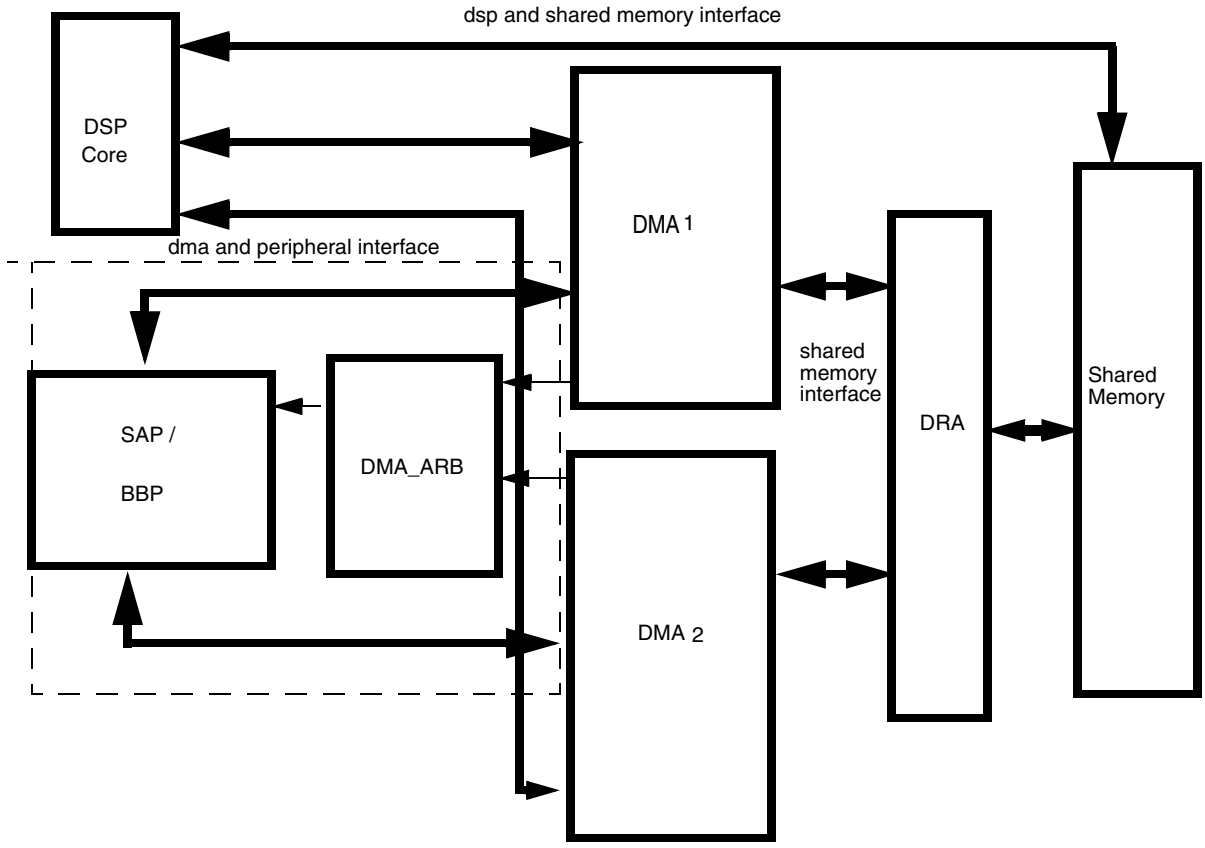
Revision History Table

Revision	Date	Author	Changes
0.0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	03/20/02	Saurabh Chutani	scan_en changed to ipt_test_clk_se
0.2	05/03/02	Saurabh Chutani Adil Kidwai	Updated Port List and Block diagram
0.3	05/24/02	Saurabh Chutani	Added DMA2 Register Addresses.
0.4	07/19/02	Saurabh Chutani Shannon Osgood	Updated per DDTS# DSPhl14371. Added note from DDTS# DSPhl13500 to DTOR bit description.
0.5	04/30/03	Basil Jackson, Shannon Osgood	Updated per DDTS# DSPhl16235.
0.6	05/07/03		Updated for LTE specification release.

The DSP Peripheral Modules Direct Memory Access (DMA) Controller is an on-chip peripheral that permits data transfers between internal memory and internal I/O in any combination, without intervention of the program. In Neptune there is one instantiation of the DMA block. Due to dedicated DMA address and data buses as well as internal memories partition, a high level of isolation is achieved where the DMA operation does not interfere or slow down the core operation.

Each DMA has one channel, which can be allocated by software to one of four possible peripheral channels (BBP receive and transmit, SAP receive and transmit). The DMA Control and Count Registers control the entire module. All the registers are memory-mapped in the internal DSP I/O memory space.

The DMA supports pack and unpack operations. Pack operation involves combining two DSP-memory bytes to create one peripheral word. Unpack operation splits one peripheral word into two DSP-memory bytes.



DMA Interfaced with DSP, Peripherals and Shared Memory

Figure 17-1. DMA Interfaces

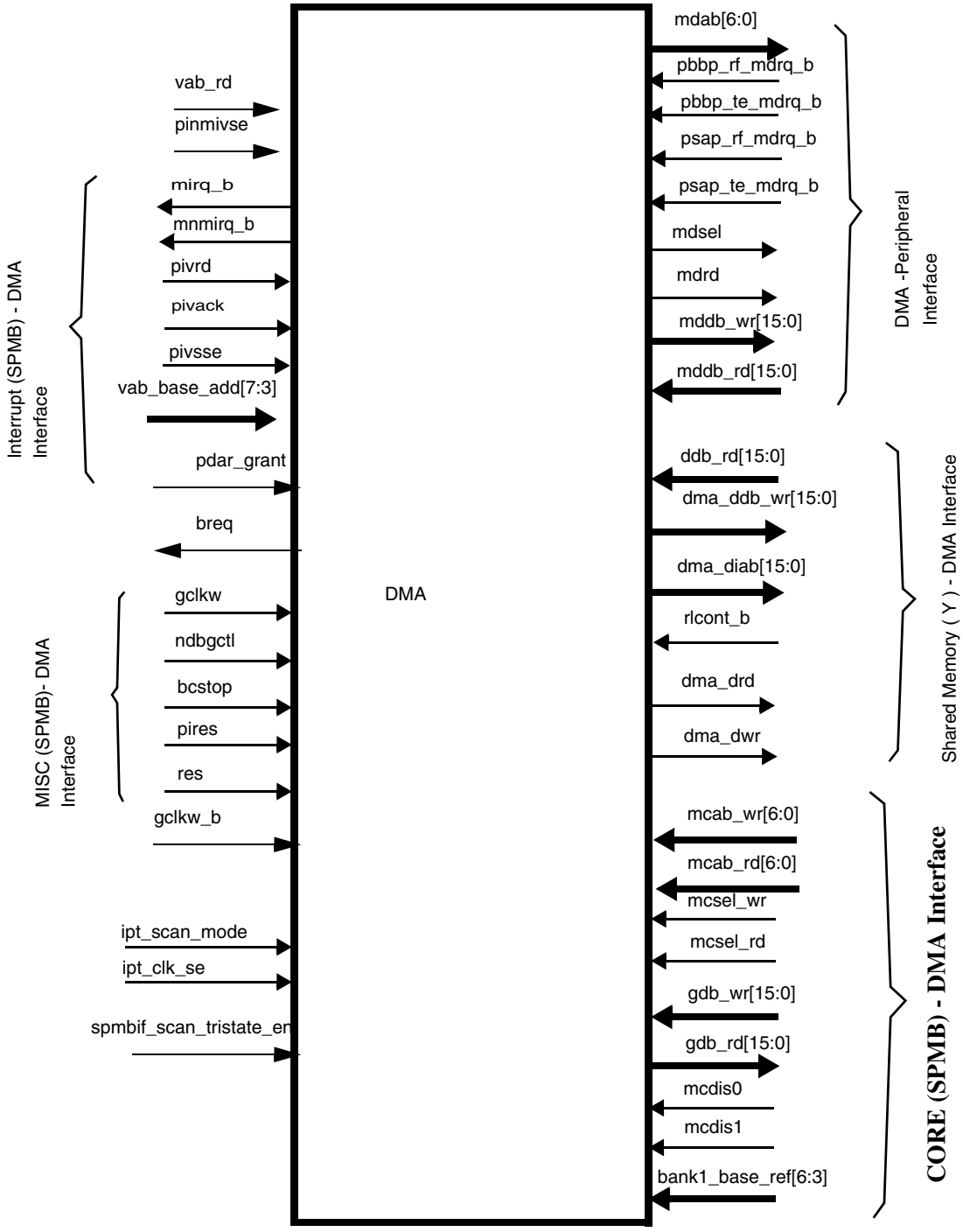


Figure 17-2. DMA Pin Diagram

17.1 DMA Module Pin List

Table 17-1 is a list of all the pins in the DMA module.

Table 17-1. DMA Module Pin List

Pin Name	Direction	Description	signal_name	envelope_pin
		CORE(SPMB) - DMA Interface		
mcab_wr[6:0]	I	Core Write address bus		
mcab_rd[6:0]	I	Core Read address bus		
mcsel_wr	I	X or Y IO Write select, connected at the module instantiation level to one of MCSELX_WR or MCSELY_WR		
mcsel_rd	I	X or Y IO Read select, connected at the module instantiation level to one of MCSELX_RD or MCSELY_RD		
gdb_wr[15:0]	I	Core Global Write data bus(GDB_WR[15:0] for 566xx family		
gdb_rd[15:0]	O	Core Global Read data bus(GDB_RD[15:0] for 566xx family		
mcdis0/mcdis1	I	Core pipeline stall Indication (used in SPMBIF only, hidden for the user)		
bank1_base ref	I	Bank select base address (to be compared with the address bus, to determine if the address is a valid one).		
		DMA (SPMB) - Peripheral Interface		
mdab[6:0]	O	DMA address bus		
mdsel	O	DMA-XIO Select. Connected at the module instantiation level to one of the four following PMB signals MDSELX,MDSELY.		
mdrd	O	DMA read		
mddb_wr[15:0]	O	DMA write data bus		
mddb_rd[15:0]	I	DMA read data bus		
pbbp_mdrqrf_b	I	DMA request from BBP (Tx).		
pbbp_mdrqte_b	I	DMA request from BBP (Rx).		
psap_mdrqrf_b	I	DMA request from SAP (Tx).		
psap_mdrqte_b	I	DMA request from SAP (Rx).		

Table 17-1. DMA Module Pin List

Pin Name	Direction	Description	signal _name	envelop e_pin
Interrupt (SPMB) - DMA Interface				
mirq_b	O	Peripheral Module Interrupt Request. Connected at the module instantiation level to one of the 12 interrupt request line mirqn[11:0]. Active low.		
mnmirq_b	O	Non Maskable Interrupt request.		
pivrd	I	Vector Read Strobe.		
pivack	I	Peripheral Module Interrupt Acknowledge.		
pivsse	I	Interrupt Vector Source Select. Connected at the module instantiation level to one of the 12 interrupt Vector Source Select line pivsse[11:0].		
pinmivse	I	Non Maskable Interrupt Vector Source Select. Used when one or more vec-tors correspond to a non maskable interrupt; else connected to gnd.		
vab[7:1]	O	Interrupt Vector Bus.		
vab_base_add [7:3]	I	Interrupt vector base address.		
MISC(SPMB) - DMA Interface				
gclkw	I	Raw clock for New Custom Module		
ndbgctl	I	The core is in the DEBUG mode' indication		
bcstop	I	This signal indicates that the core is in stop mode. It is asserted four cycles before the gclkw goes idle.		
pires	I	Peripheral Module's Software reset.		
res	I	Peripheral Module's Hardware reset.		
Shared Memory (Y) - DMA Interface				
ddb_rd[15:0]	I	Dma read data bus		
dma_ddb_wr[15:0]	O	Dma write data bus		
dma_diab [15: 0]	O	Dma address		
dma_drd	O	Dma read control		
dma_dwr	O	Dma write control		
rlcont_b	I	Asserted when dma access to memory conflicts with core access to the same memory module.		

Table 17-1. DMA Module Pin List

Pin Name	Direction	Description	signal_name	envelope_pin
DMA Scan Interface				
spmbif_scan_tristate_enable	I	Added for Neptune scan divergence mode.		
ipt_scan_mode	I	Scan mode signal		
ipt_clk_se	I	Scan enable source signal (for gating with scan mode).		

17.2 DSP Peripherals DMA Architecture

The block diagram of the DMA is given in Figure 17-3

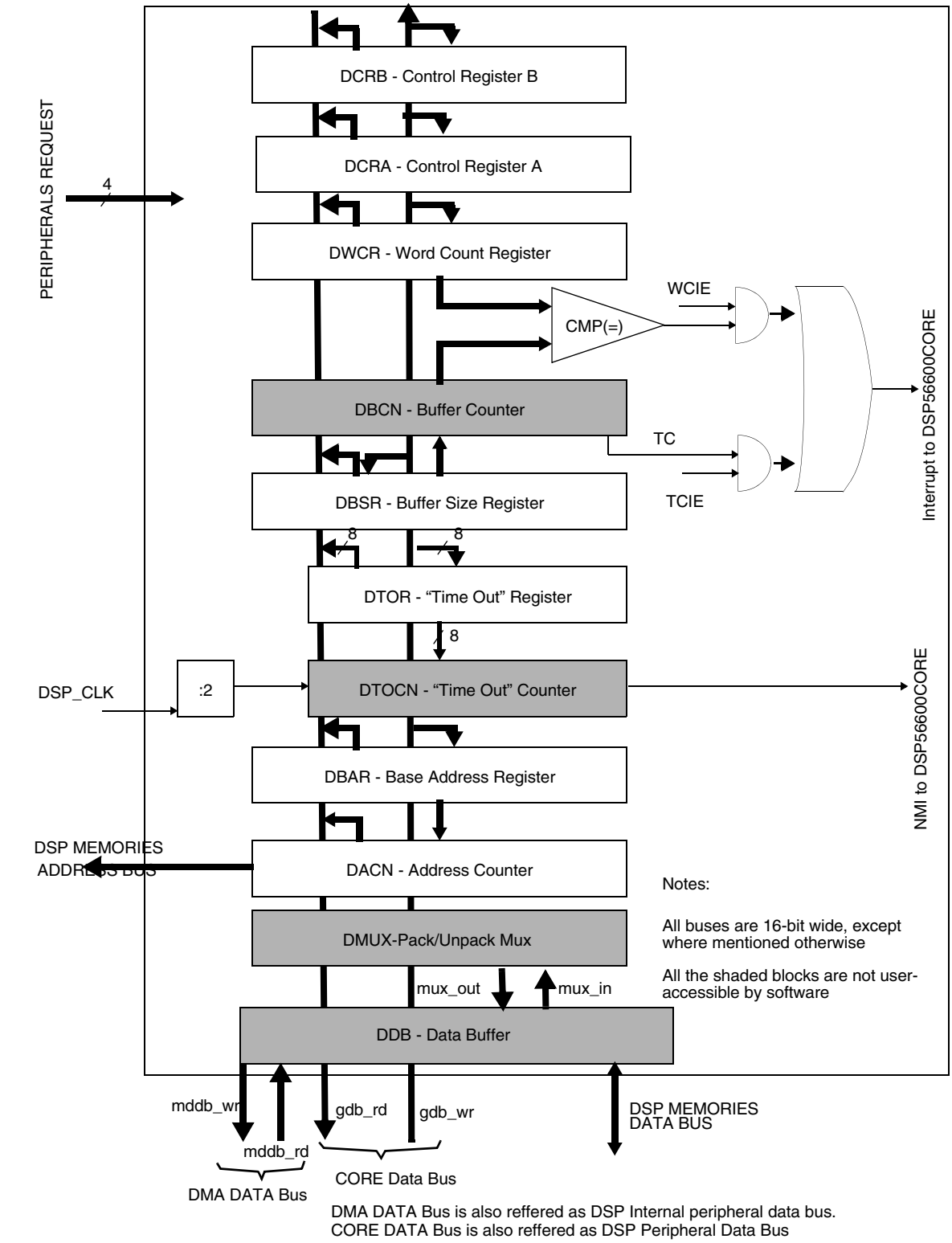


Figure 17-3. DMA Block Diagram

17.3 Programming Model

The DSP Peripherals DMA programming model is described in the following sections.

17.3.1 Register Summary

KEY: Always Reads One (1) | Always Reads Zero (0) | Read/Write Bit (bit) | Read-Only Bit (bit) | Write-Only Bit (bit) | Write 1 to Clear (bit, *W1C*) | Self-Clear Bit (0, bit) | N/A

Table 17-2. DMA1 Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCRB X:$FF99	R	0	0	0	0	0	0	0	0	0	0	0	PDA	PMM [1:0]		PBO	PKE
	W																
DTOR X:$FF9A	R	0	0	0	0	0	0	0	0	DTOR [7:0]							
	W																
DBSR X:$FF9B	R	DBSR [15:0]															
	W																
DWCR X:$FF9C	R	DWCR [15:0]															
	W																
DACN X:$FF9D	R	DACN [15:0]															
	W																
DBAR X:$FF9E	R	DBAR [15:0]															
	W																
DCRA X:$FF9F	R	DPE	TE	WCIE	TCIE	DAUTO	0	PRQ[2:0]		PRA [6:0]							
	W																

Table 17-3. DMA2 Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCRB Y:$FF99	R	0	0	0	0	0	0	0	0	0	0	0	PDA	PMM [1:0]		PBO	PKE
	W																
DTOR Y:$FF9A	R	0	0	0	0	0	0	0	0	DTOR [7:0]							
	W																
DBSR Y:$FF9B	R	DBSR [15:0]															
	W																
DWCR Y:$FF9C	R	DWCR [15:0]															
	W																
DACN Y:$FF9D	R	DACN [15:0]															
	W																
DBAR Y:$FF9E	R	DBAR [15:0]															
	W																
DCRA Y:$FF9F	R	DPE	TE	WCIE	TCIE	DAUTO	0	PRQ[2:0]		PRA [6:0]							
	W																

17.3.2 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various DMA registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

The read/write 16-bit DMA Control Register A(DCRA) enables the DMA, controls the operating mode and the interrupt mode. The DCR bits are described in the following paragraphs.

DCRA DMA Control Register A **Addr X:$FF9F**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	DPE	TE	WCIE	TCIE	DAUTO		PRQ [2:0]			PRA [6:0]						
TYPE:	rw	rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-4. DCRA Description

Name	Description	Settings
DPE Bit 15	**DMA Enable**—The read/write control bit DPE enables the operation of the DMA. This bit is cleared by hardware and software reset.	DPE = 0→ DMA is disabled. DPE = 1→ DMA is enabled.
TE Bit 14	**Transfer Enable**—The read/write control/status TE bit enables the start of the transfer and indicates the transfer completion. This bit can be set by software only, explicitly writing it to 1. It can be cleared by software (explicitly writing it to 0, which will stop the DMA activity after the completion of the current DMA buffer transfer, regardless of the DAUTO bit value) or by the internal DMA hardware (when the whole buffer has been completed, i.e. when a Terminal Count condition occurs). This bit is cleared after hardware and software reset. It is also cleared when exiting the STOP mode.	TE = 0 → Transfer was completed. TE = 1 → Transfer enabled and not completed.
WCIE Bit 13	**Word Count Interrupt Enable**—The read/write WCIE bit enables the Word Count interrupt.This bit is cleared after hardware and software reset.	WCIE = 0 → Interrupt disabled. WCIE = 1 → Interrupt enabled.
TCIE Bit 12	**Terminal Count Interrupt Enable**—The read/write TCIE bit enables the Terminal Count interrupt. This bit is cleared after hardware and software reset.	TCIE = 0 → Interrupt disabled. TCIE = 1 → Interrupt enabled.

Table 17-4. DCRA Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>DAUTO
Bit 11</td><td>Automatic Restart—The read/write DAUTO bit enables the automatic restart of the DMA activity after the transfer completion.
If this bit is set, then the TE bit is not cleared at the end of the transfer when the whole buffer transfer has been completed, but the DMA operation will restart, awaiting for new incoming transfer requests from the Peripheral Module.
This bit is cleared after hardware and software reset.</td><td>DAUTO = 0 → Automatic restart disabled.
DAUTO = 1 → Automatic restart enabled.</td></tr>
<tr><td>Bit 10</td><td>Reserved—This bit is read as zero and should be written with zero for future compatibility.</td><td>N/A</td></tr>
<tr><td>PRQ[2:0]
Bits 9-7</td><td>Peripheral Request—The read/write PRQ2-PRQ0 bits specify which one of the peripheral request lines will be the one to trigger a DMA transaction. These bits are cleared after hardware or software reset.</td><td>
<table>
<tr><th>PRQ[2:0]</th><th>Peripheral request</th></tr>
<tr><td>000</td><td>BBP RX</td></tr>
<tr><td>001</td><td>BBP TX</td></tr>
<tr><td>010</td><td>SAP RX</td></tr>
<tr><td>011</td><td>SAP TX</td></tr>
<tr><td>100</td><td>undefined</td></tr>
<tr><td>101</td><td>undefined</td></tr>
<tr><td>110</td><td>undefined</td></tr>
<tr><td>111</td><td>undefined</td></tr>
</table>
</td></tr>
<tr><td>PRA[6:0]
Bits 6-0</td><td>Peripheral Register Address—The read/write PRA6-PRA0 bits specify the 7 least significant bits of the address of the peripheral register which is connected to the DMA channel. These bits are cleared after hardware or software reset.</td><td>
<table>
<tr><th>PRA[6:0]</th><th>Peripheral register</th></tr>
<tr><td>$2a</td><td>BBP RX</td></tr>
<tr><td>$2c</td><td>BBP TX</td></tr>
<tr><td>$4a</td><td>SAP RX</td></tr>
<tr><td>$4c</td><td>SAP TX</td></tr>
</table>
</td></tr>
</table>

The read/write 16-bit DMA Control Register B(DCRB) Enables and controls the pack options.

DCRB DMA Control Register B **Addr X:$FF99**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
												PDA	PMM [1:0]		PBO	PKE
TYPE:	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-5. DCRB Description

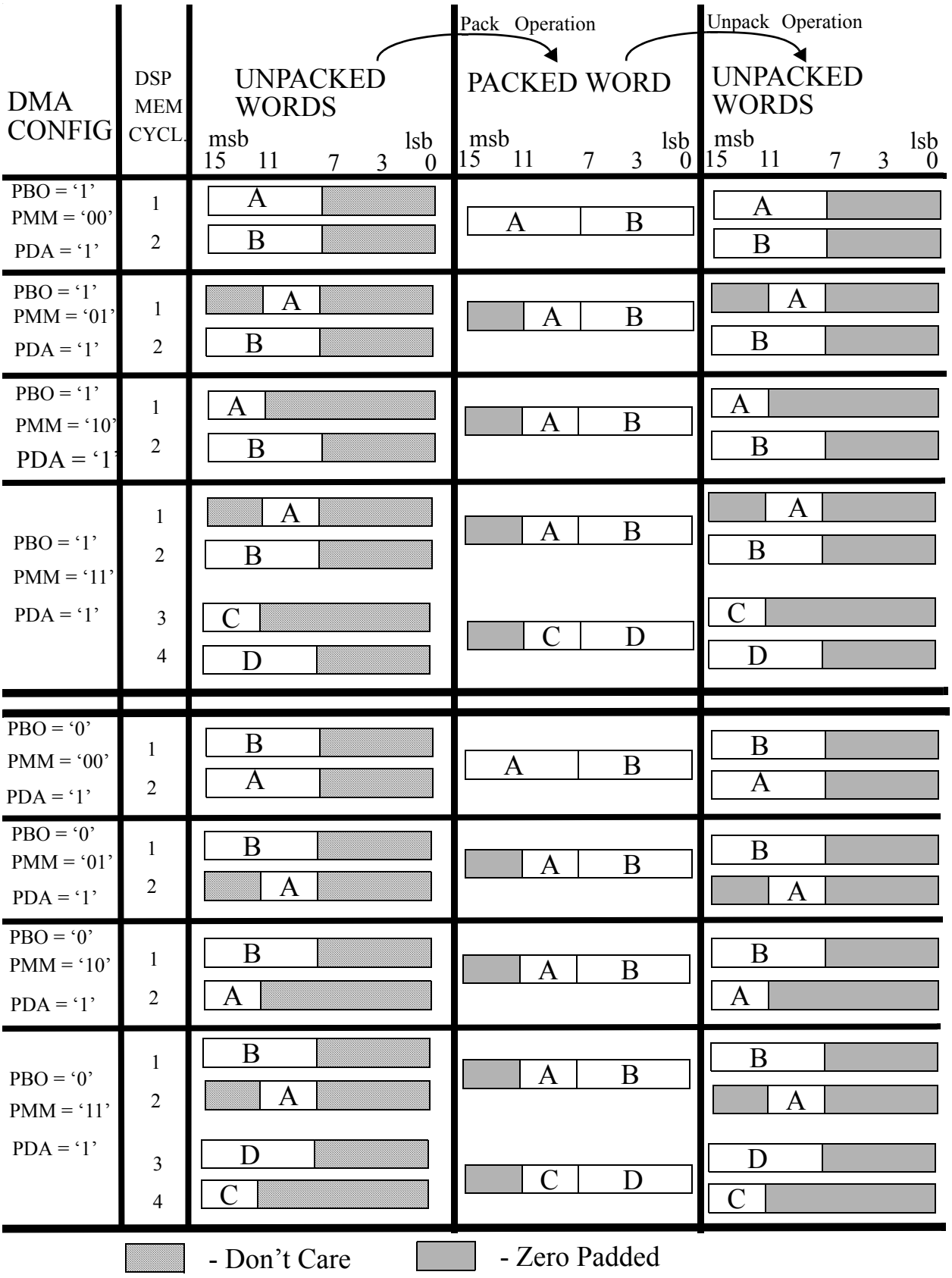
Name	Description	Settings
Bits 15-5	**Reserved bits** — These bits are read as zeros and should be written with zeros for future compatibility	N/A
PDA Bit 4	**Packed Data Alignment—**The read/write PDA bit selects the alignment of a12 packed bits in the DMA, before unpack operation, or after pack operation. PDA is sampled when TE bit asserted, and DMA is not in the middle of data transfer cycle.This bit is cleared after hardware or software reset. **Note:** When packed data is 16 bits wide (PMM[1:0] = '00'), PDA bit is irrelevant and ignored. Figure 17-4 describes PDA bit effect on DMA data transfer.	PDA = 0 → Packed data is left (MS bit) aligned PDA = 1→ Packed data is right (LS bit) aligned

Table 17-5. DCRB Description (Continued)

Name	Description	Settings
PMM[1:0] Bits 3-2	**Pack MS byte Mode—**The read/write PMM[1:0] bits select the way the MS (Most Significant) byte is processed during pack/unpack operations. PMM[1:0] is sampled when the TE bit is asserted, and the DMA is not in the middle of a data transfer cycle.These bits are cleared after hardware or software reset.Figure 17-4 describes PMM[1:0] bits effect on DMA data transfer.	PMM[1:0] = '00'→ All eight bits of the MS byte is packed/unpacked. PMM[1:0] = '01'→ Lower four bits of the MS byte are packed/unpacked.Upper four bits of MS byte ignored. During pack cycle, lower four bits of MS byte are combined with eight bit of LS byte to create the 12 bits of peripheral data word. During unpack operation, upper four out of 12 packed bits are used as the lower four bits of the DSP-memory data MS byte. PMM[1:0] = '10'→ Upper four bits of MS byte are packed/unpacked.Lower four bits of MS byte ignored. During pack cycle, upper four bits of MS byte are combined with eight bit of LS byte to create the 12 bits of peripheral data word. During unpack operation, upper four out of 12 packed bits are used as the upper four bits of the DSP-memory data MS byte. PMM[1:0] = '11' → Mixed operation. On the first pack/unpack cycle (of two consecutive DSP-memory transfers), lower four bits of MS byte are packed/unpacked, and higher four bits of MS byte ignored (same as PMM[1:0] = '01').On the next pack/unpack cycle (of the following two consecutive DSP-memory transfers), higher four bits of MS byte are packed/unpacked, and lower four bits of MS byte ignored (same as PMM[1:0] = '10'). When TE bit asserted, mixed operation is reset to start with selection of the lower four bits in the MS byte.

Table 17-5. DCRB Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>PBO
Bit 1</td><td>Pack Bytes Order (PBO)—The read/write PBO bit selects the order in which the DSP-memory data bytes are packed/unpacked. PBO is sampled when the TE bit is asserted and DMA is not in the middle of data transfer cycle.This bit is cleared after hardware or software reset.
When PBO is high during a peripheral to DSP-memory data transfer (unpack operation), the MS (Most Significant) byte of the peripheral data word is unpacked and transferred to DSP-memory first. When PBO is low, the LS (Least Significant) byte of the peripheral data word is unpacked and transferred to DSP-memory first.
When PBO is high during a DSP-memory to peripheral data transfer (pack operation), the first byte received from DSP-memory is stored as MS byte of packed word and the second byte is stored as LS byte. When PBO is low, the first byte received from DSP-memory is stored as LS byte of packed word and the second byte is stored as MS byte.
Figure 17-4 describes the PBO bit effect on DMA data transfer.</td><td><table>
<tr><th>PBO</th><th>Unpack Operation</th><th>Pack Operation</th></tr>
<tr><td>0</td><td>LS (Least Significant) byte of the peripheral data word is unpacked and transferred to DSP-memory first.</td><td>The first byte received from DSP-memory is stored as LS byte of packed word and the second byte is stored as MS byte.</td></tr>
<tr><td>1</td><td>MS (Most Significant) byte of the peripheral data word is unpacked and transferred to DSP-memory first.</td><td>The first byte received from DSP-memory is stored as MS byte of packed word and the second byte is stored as LS byte.</td></tr>
</table></td></tr>
<tr><td>PKE
Bit 0</td><td>Pack Enable—The read/write PKE bit enables the pack/unpack option. PKE is sampled when TE bit asserted, and DMA is not in the middle of data transfer cycle. This bit is cleared after hardware or software reset.
When PKE is enabled during a peripheral to DSP-memory data transfer, an unpack operation takes place. During unpack operation, the DMA splits the received peripheral word into two bytes, and transfers them to DSP-memory to the most significant bytes of two consecutive words. The least significant bytes of these words will be set to zero.
When PKE is enabled during a DSP-memory to peripheral data transfer, a pack operation takes place. During pack operation, the DMA combines two consecutive words from DSP-memory into one word to be sent to the peripheral.
The other sections of Table 17-5, DCRB Description, and Figure 17-4 describe the details and options of pack/unpack option.</td><td>PKE = 0 → Pack/Unpack option disabled.
PKE = 1 → Pack/Unpack option enabled.</td></tr>
</table>


DMA CONFIG
DSP MEM CYCL.
UNPACKED WORDS
msb 15 11 7 3 lsb 0
Pack Operation
PACKED WORD
msb 15 11 7 3 lsb 0
Unpack Operation
UNPACKED WORDS
msb 15 11 7 3 lsb 0
PBO = '1'
PMM = '00'
PDA = '1'
PBO = '1'
PMM = '01'
PDA = '1'
PBO = '1'
PMM = '10'
PDA = '1'
PBO = '1'
PMM = '11'
PDA = '1'
PBO = '0'
PMM = '00'
PDA = '1'
PBO = '0'
PMM = '01'
PDA = '1'
PBO = '0'
PMM = '10'
PDA = '1'
PBO = '0'
PMM = '11'
PDA = '1'
- Don't Care
- Zero Padded

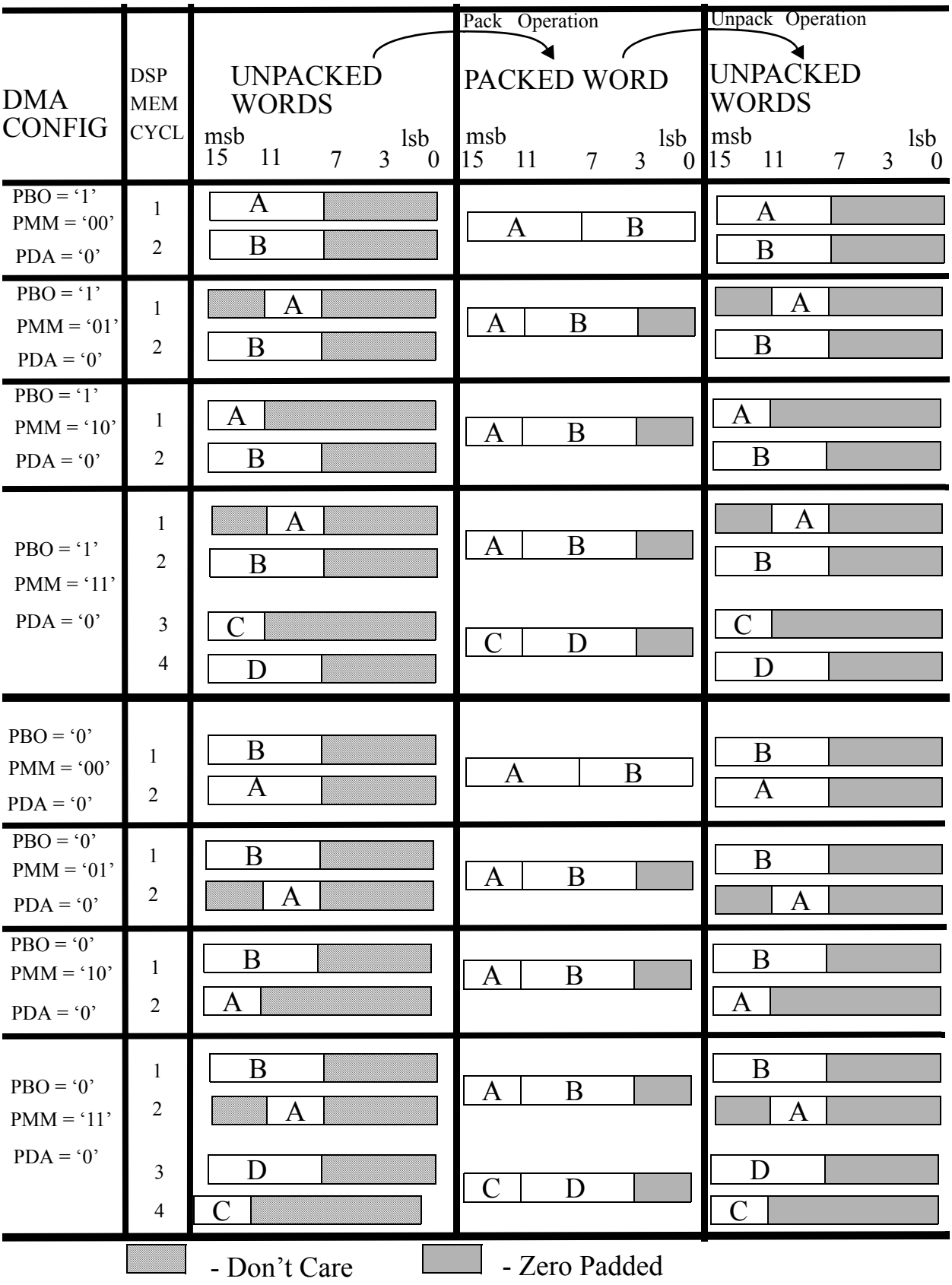


Figure 17-4. DMA Pack/Unpack Modes Summary

DBAR DMA Base Address Register **Addr X:$FF9E**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	DBAR[15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-6. DBAR Description

Name	Description	Settings
DBAR [15:0] Bits 15-0	**Base Address Register**—The read/write 16-bit DMA Base Address Register (DBAR) contains the 16-bit absolute start address of the data buffer. The DBAR bits are cleared after hardware reset.	N/A

DACN DMA Address Counter **Addr X:$FF9D**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	DACN[15:0]															
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Table 17-7. DACN Description

Name	Description	Settings
DACN [15:0] Bits 15-0	**Address Counter**—The 16-bit read-only DMA Address Counter (DACN) is an up-counter which generates the address for the memory location of the DMA transfer. DACN is loaded with the DBAR contents when the bit TE set to 1 by software and DMA is not in the middle of data transfer cycle. DACN increments after each DMA transfer up to 2^{16}-1, regardless of the allocated buffer addresses in the data memory and is read-only by software. **Note:** It is under user's responsibility to guarantee that the addresses generated by the DMA (DACN) will be within the memory addresses allocated to the DMA. **Note:** For a given number of peripheral data transfers, in pack/unpack mode, number of DACN counts is double, comparing with the case of pack/unpack disabled.	N/A

DWCR DMA Word Count Register **Addr X:$FF9C**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	DWCR[15:0]															
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-8. DWCR Description

Name	Description	Settings
DWCR [15:0] Bits 15-0	**Word Count Register**—The read/write 16-bit DMA Word Counter Register (DWCR) is compared to the DBCN value at the end of each DMA data transfer cycle. When DBCN equals the DWCR value a word count event interrupt (if enabled by the WCIE bit) is generated by the DMA to the DSP56600CORE. Additional (DWCR_value) data transfers take place between the word count event and the terminal count event at the end of the buffer. **Note:** When WDPD is in pack/unpack mode (PKE = '1' at DCRB register), DWCR should be written with even values only, or an erratic WDPD behavior may occur.(In pack/unpack mode transfers to/ from memory would always be no. of words x 2 =>always even).	N/A

DBSR DMA Buffer Size Register **Addr X:$FF9B**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	DBSR[15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-9. DBSR Description

Name	Description	Settings
DBSR [15:0] Bits 15-0	**Buffer Size Register**—The read/write 16-bit DMA Buffer Size Register (DBSR) contains the 16-bit value of the buffer length for the total number of DMA transfers after which a Terminal Count interrupt (if enabled by the TCIE bit) will be generated by the DMA to the DSP56600CORE. The DBSR bits are cleared after hardware and software reset. **Note:** For a buffer size <n>, write DBSR with <n-1>. When WDPD is in pack/unpack mode (PKE = '1' at DCRB register), DBSR should be written with odd values only, or an erratic WDPD behavior may occur.	N/A

17.3.3 DMA Buffer Counter (DBCN)

The 16-bit DMA Buffer Counter (DBCN) is a down-counter which is loaded with the buffer size value (from DBSR) and decrements after each DMA transfer, simultaneously with the DACN increment. At roll-over a "Terminal Count Interrupt" (if enabled by the TCIE bit) is generated by the DMA to the DSP56600CORE. DBCN is loaded with the DBSR contents in one of the following two conditions:

- bit TE set to 1 by software and DMA is not in the middle of data transfer cycle.
- at roll-over.

DBCN decrements after each DMA transfer and is not accessible by software.

NOTE:

For a given number of peripheral data transfers, in pack/unpack mode, number of DBCN counts is double, comparing with the case of pack/unpack disabled. (value of n in DBCN => n+1 transfers).

DTOR DMA Time Out Register **Addr X:$FF9A**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									DTOR[7:0]							
TYPE:	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-10. DTOR Description

Name	Description	Settings
Bits 15-8	**Reserved**	N/A
DTOR [7:0] Bits 7-0	**Time Out Register**—The read/write 16-bit DMA Time Out Register (DTOR) contains an 8-bit value, which, multiplied by 2 plus 1, is the number of DSP_clocks after which a non-maskable interrupt will be generated by the DMA to the DSP56600CORE. This condition may occur if a new DMA request is issued by the DMA and not completed after a number of (DTOR_value x 2 +1) DSP clocks, due to the consecutive accesses performed by the DSP56600CORE to the same 1/4K words memory module. The DTOR non-reserved bits cleared after hardware and software reset. **Note:** For proper functioning of DMA, DTOR should always be written, with a value not less than 4 (even when DTOR functionality is not required.)	N/A

17.3.4 DMA Time Out Counter (DTOCN)

The 8-bit DMA Buffer Counter (DTOCN) is a down-counter which is loaded with the value of DTOR after the Peripheral Module issues the DMA transfer request. During a peripheral to memory data transfer, DTOCN will start counting when a second peripheral request is issued before transfer cycle of the previous is finished.During a memory to peripheral data transfer, DTOCN will start counting when transfer cycle is delayed due to a DSP-core access to memory.DTOCN is decrementing at a rate of half the frequency of the DSP clock. The DTOCN is gated off and stops down counting when last peripheral request is serviced. At roll-over a “non-maskable interrupt is generated by the DMA to the DSP56600CORE. DTOCN is re-loaded with the DTOR contents when a new transfer request is issued by the Peripheral Module. DTOCN is not accessible by software.

17.3.5 DMA Data Buffer (DDB)

The 16-bit DMA Data Buffer (DDB) is a buffer which holds the data before or after the DMA transfer (depending on the transfer direction) and is not accessible by software.

17.4 Description of the Operation

17.4.1 DMA Initialization

The following list of operations describes the right sequence for safe DMA initialization:

1. Configure all DMA registers except for TE and DPE bits.
2. Set DPE and TE bits. DMA responds with loading DBAR value into DACN counter and DBSR value into DBCN counter.
3. Configure and enable the DSP peripherals served by the DMA.

17.4.2 Operating DMA with BBP and SAP

BBP and SAP are two serial communication ports. DMA can be used to load transmitted data, or unload received data from these peripherals. Supposing all DMA interrupt (word count, terminal count, time out) are enabled, here is a recommendation for BBP/SAP interrupt configuration:

1. Rx full and Tx empty are not recommended to be enabled when DMA is active, since DSP core has nothing to do in addition to the DMA service.
2. Rx error and Tx error are recommended to be enabled, since they can indicate a DMA failure to load/unload the peripheral.
3. Last slot and frame count could be operated, although a similar functionality could be received from DMA's word and terminal count interrupts.

17.4.3 Peripheral to DSP-memory data transfer

The following is a description of the consecutive events happen during peripheral to DSP-memory data transfer:

1. Peripheral sends Rx request toward the DMA.
2. DMA requests for peripheral access, and replied with grant signal by the DMA arbitration logic[1].
3. DMA fetches 16 bit word from the peripheral.
4. Peripheral removes the Rx request.
5. DMA generate a write cycle to the DSP-memory. If unpack option disabled (PKE = 0), it contains all the 16 bits received from the peripheral. If unpack option enabled (PKE =1), only one byte of the peripheral data word is transmitted to DSP-memory.
6. In case the same 1/4K DSP-memory is occupied by DSP core, DMA write cycle will be finished at the next cycle in which DSP-memory is free again.
7. If next peripheral request arrives during this write to DSP-memory latency, DTOCN starts down counting. DTOCN rollover generates non-maskable interrupt toward the DSP core. DTOCN stops counting when the peripheral request caused it is serviced and removed.
8. DACN is incremented and DBCN is decremented.Word count and terminal count are issued if necessary.

1. DMA arbitration logic is not present in Neptune since there is only a single DMA; the DMA should always have access to the peripheral.

9. If unpack option enabled (PKE =1), stages 5,6,7 and 8 are repeated for transfer of the second byte of the peripheral data word. TOCN count which is started during the first DSP-memory access, will continue uninterrupted during the second access and stop only when the peripheral request caused it is serviced.
10. If terminal count not yet happened (DBCN >0), or DAUTO bit is set, then the TE bit will be not be cleared by the DMA hardware and the DMA activity will be resumed, waiting for a new Peripheral Module request.
11. If terminal count happened (DBCN = 0) and DAUTO bit is cleared, the TE bit will be cleared by the DMA hardware and the DMA activity will be frozen and any peripheral request will be ignored.

17.4.4 DSP-memory to Peripheral data transfer

The following is a description of the consecutive events happen during DSP-memory to peripheral data transfer:

1. Peripheral sends Tx request toward the DMA.
2. DMA generate a read cycle to the DSP-memory. If pack option disabled (PKE = 0), it contains all the 16 bits to be transmitted to the peripheral. If pack option enabled (PKE =1), only one byte is used to build the peripheral data word.
3. In case the same 1/4K DSP-memory is occupied by DSP core, DMA read cycle will be finished at the next cycle in which DSP-memory is free again.
4. During this read to DSP-memory latency, DTOCN is down counting. DTOCN rollover generates non-maskable interrupt toward the DSP core. DTOCN stops counting when the present peripheral request is serviced and removed.
5. DACN is incremented and DBCN is decremented.Word count and terminal count are issued if necessary.
6. If pack option enabled (PKE =1), stages 5,6,7 and 8 are repeated to receive the second byte of the peripheral data word.TOCN count which is started during the first DSP-memory access, will continue uninterrupted during the second access and stop only when the present peripheral request is serviced.
7. DMA requests for peripheral access, and replied with grant signal by the DMA arbitration logic[1].
8. DMA writes 16 bit word to the peripheral.
9. Peripheral removes the Tx request.
10. If terminal count not yet happened (DBCN >0), or DAUTO bit is set, then the TE bit will be not be cleared by the DMA hardware and the DMA activity will be resumed, waiting for a new Peripheral Module request.
11. If terminal count happened (DBCN = 0) and DAUTO bit is cleared, the TE bit will be cleared by the DMA hardware and the DMA activity will be frozen and any peripheral request will be ignored.

1. DMA arbitration logic is not present in Neptune since there is only a single DMA; the DMA should always have access to the peripheral.

17.4.5 DMA processing cycle times

Table 17-11 describes the DMA data transfer cycle time in terms of DSP cycles. Latency of peripheral access grant or contention period in DSP-memory access, if happened, should be added to Table 17-11 rates

Table 17-11. Number of DSP cycles per 16-bit data transfer

	memory to periph.	periph. to memory
Normal operation	7	7
Pack/Unpack operation	9	9

17.4.6 DSP Low Power Modes

During WAIT state, the DMA will. continue to operate normally.

During STOP state the DMA will be frozen and the TE bit will be cleared when returning to normal mode. Therefore any DMA transfer started before entering the STOP state will be aborted and will not be resumed automatically when returning to normal mode.

NOTE:

DSP peripherals registers and DSP-memory buffer which can be addressed by the DMA are chip dependent information and hence are given in the Chip Configuration Chapter.

17.5 Programmers Notes

- Writing DBSR with <n> means buffer size of <n+1>. Writing DWCR with <n> means interrupt DSP when <n> word remain in buffer.
- On Redcap2, the DMA Address Count Register (DACN) is not updated at the same time the word count interrupt is generated. On Neptune, the word count register is updated at the same time the word count interrupt is generated. As a result, if the value of DACN is read immediately after the word count interrupt (in the interrupt service routine, for example), the value read on Neptune will be one word larger than the value read on Redcap2. To summarize:
  - 1) The interrupt is generated at the same point in both designs.
  - 2) The value read from DACN after a word count interrupt on Neptune should be one word bigger than on Redcap2.

Chapter 18
DMA Arbitration (DMA_ARB)

Revision History Table

Revision	Date	Author	Changes
0.0	12/13/01		
0.1	03/20/02	Saurabh Chutani	scan_en changed to ipt_test_clk_se
0.2	05/07/03		Updated for LTE specification release.

18.1 Overview

DMA arbitration logic (PDAR module) is designed to meet the following requirements:

1. Co-existence of up to four DMA controller to peripheral connections on board.
2. Sharing one interrupt port of DSP core with up to 4 interrupt sources.DMA Controllers to DSP peripherals interface

PDAR module supports the arbitration of up to four DMA-peripheral connections (two are utilized on Patriot-Indy: DMA1, DMA2,). Multiple DMA connections through the PDAR are based on the PMB interface definition for a single DMA controller, with the addition of a bus_request and grant signals to allow the single DMA using the peripherals interface. When more than one DMA controller is requesting a peripheral access, PDAR will grant them one after the other, according to a hard-wired priority where DMA1 is the highest and DMA2 has the lower priority. Address bus (mdab), read/write (mdrd) and peripheral select (mdsel), are driven by the PDAR, while data bus (mddbn) and DMA service request (mdrq) are connected directly from DMA controllers to peripherals. PDAR receives the per_sel signals from the different DMA controllers and combines them to a single mdsel signal. In the same way, peripheral reads are combined to create mdrd and per_dab buses are combined to create the mdab bus. DMA service requests from all the peripheral are wired to all the DMA controllers, so any peripheral (BBP, SAP, or AAM) can be serviced by either DMA. The DMA controller's interface to theDSP-memory remains the same as in the PMB definition, and each DMA has separate buses and signal to memory.

Figure shows a timing diagram for a typical PDAR operation. The figure demonstrates write and read requests asserted together, and serviced one after the other. Write requests have the higher priority and hence they are serviced first.

Figure shows the PDAR application in Patriot Indy Chip. PDAR supports 2 DMA controllers: two DMA modules and 2 DSP-peripherals: BBP,SAP

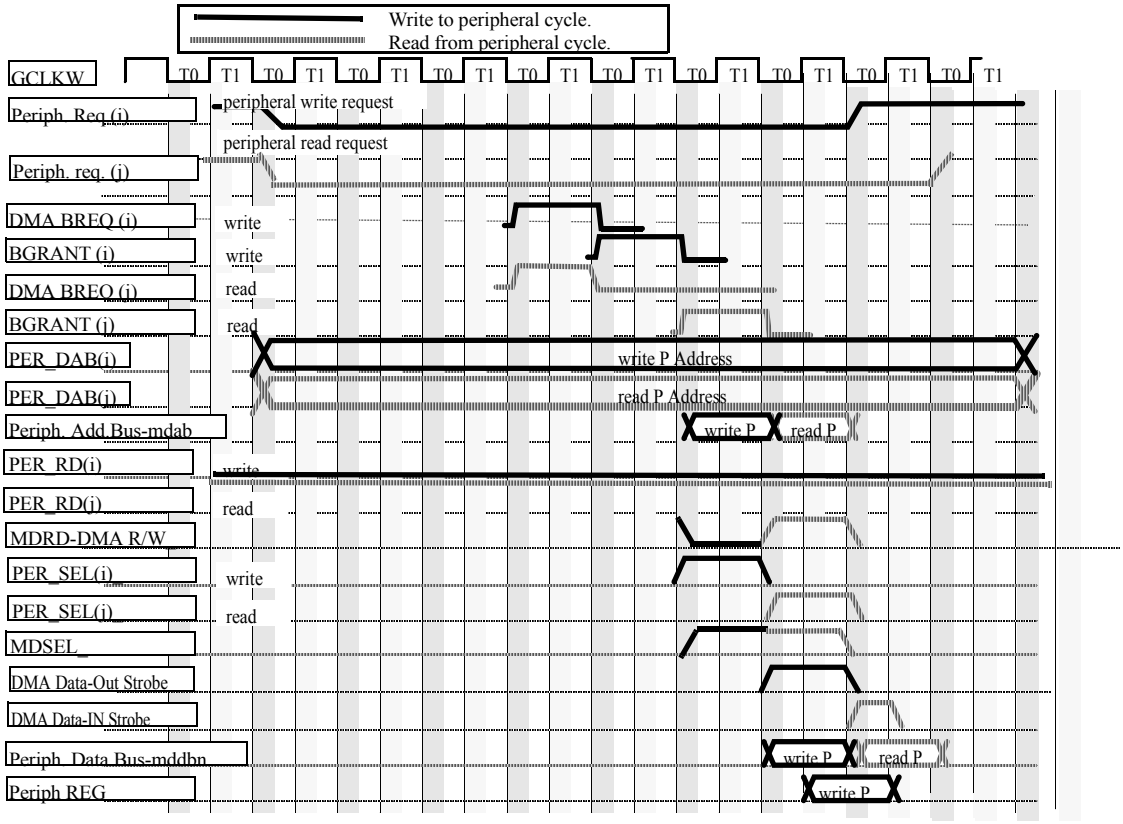


Figure 18-1. DMA-Peripheral timing diagram

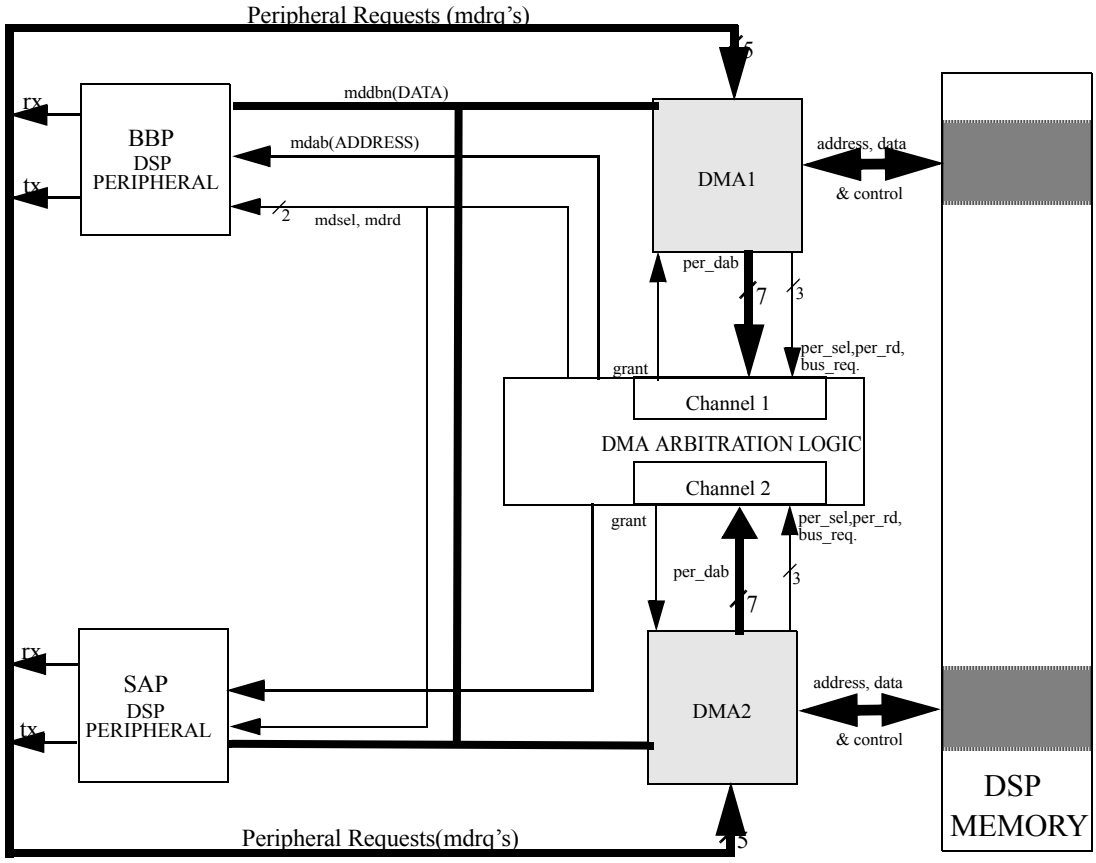


Figure 18-2. DMA-DSP Peripherals Channels Inside Patriot -Indy Chip

18.1.1 Interrupt Arbitration between DMA Controllers

Because of the DSP-core interrupt handling limitations, all DMA modules have to share the same maskable (mirq) and non-maskable (mnmirq) lines. Dealing with the non-maskable interrupt, PDAR has also to support the MDI block, and combines its NMI with those of the DMA's to create the single NMI input of the DSP-core.

Up to four maskable interrupt requests and their four interrupt selects are muxed to create a single maskable interrupt (mirq) and a single interrupt vector select (pivsse) between PDAR and DSP-core.

Up to four non-maskable interrupt requests and their four interrupt selects are muxed to create a single non-maskable interrupt (mnmirq) and a single non-maskable interrupt vector select (pinmivse) between PDAR and DSP-core.

When more than one interrupt request are issued to PDAR module, it serves them one after the other, according to a hard-wired priority when DMA1 is the highest, and DMA2 has the lowest priority.

The remaining signals of the interrupt handling interface: pivrd, pivack and vab[7:1] remain unchanged and connected directly between DSP-core and each DMA or MDI module.

Figure 18-3 shows a schematic description for the DMA interrupts arbitration inside the Patriot chip. Two normal interrupt sources: PDPD0, PDPD1 are serviced by PDAR, when PDPD0 has the higher priority. Three non-maskable interrupt sources: PDPD0, PDPD1, PMDI are serviced by PDAR, when PMDI has the highest priority, and PDPD1 has the lowest.

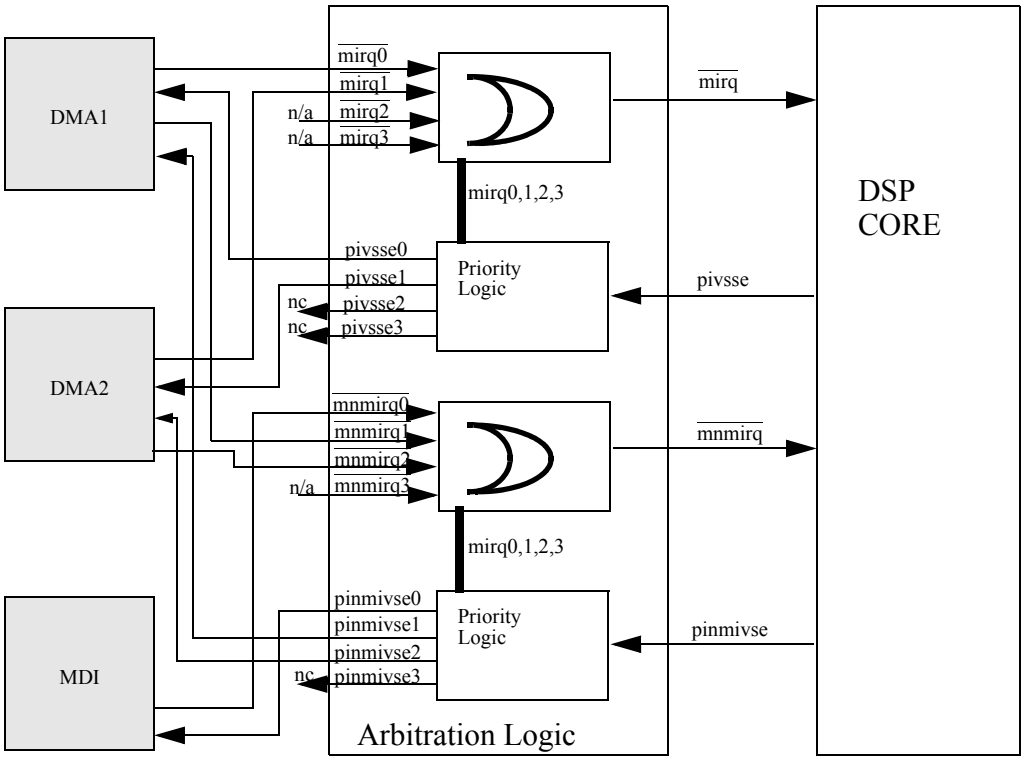

DMA1
DMA2
MDI
mirq0
mirq1
n/a mirq2
n/a mirq3
mirq
mirq0,1,2,3
DSP
CORE
pivsse0
pivsse1
nc pivsse2
nc pivsse3
Priority
Logic
pivsse
mnmirq0
mnmirq1
mnmirq2
n/a mnmirq3
mnmirq
mirq0,1,2,3
pinmivse0
pinmivse1
pinmivse2
nc pinmivse3
Priority
Logic
pinmivse
Arbitration Logic

Figure 18-3.

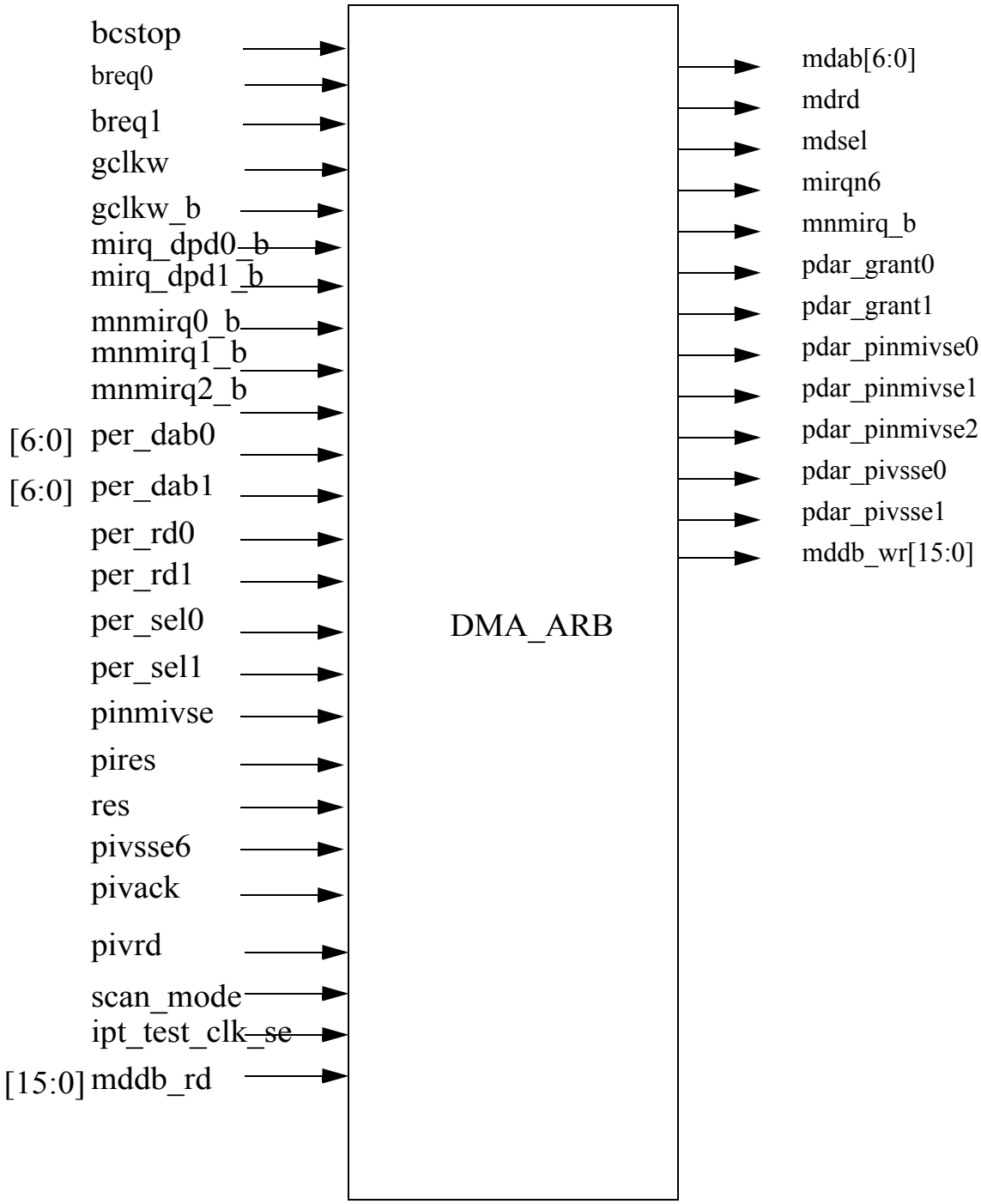


Figure 18-4. DMA_ARB Pin Diagram

Table 18-1. DMA_ARB Pin List

Name	Type	Description
gclkw	I	Input clock
gclkw_b	I	Input inverted clock
bcstop	I	core debug signal
breq0	I	Request signal from DMA1
breq1	I	Request signal from DMA2
mirq_dpd0_b	I	Interrupt signal from DMA1
mirq_dpd1_b	I	Interrupt signal from DMA2
mnmirq0_b	I	Non-maskable interrupt signal from MDI
mnmirq1_b	I	Non-maskable interrupt signal from DMA1
mnmirq2_b	I	Non-maskable interrupt signal from DMA2
per_dab0[6:0]	I	Address signal of DMA1
per_dab1[6:0]	I	Address signal of DMA2
per_rd0	I	read signal of DMA1
per_rd1	I	read signal of DMA2
per_sel0	I	peripheral select signal of DMA1
per_sel1	I	peripheral select signal of DMA2
pinmivse	I	Non-maskable source select signal
pires	I	peripheral reset signal
res	I	hardware reset
pivsse6	I	maskable source select signal
pivack	I	peripheral module vector acknowledge
pivrd	I	Interrupt Vector Read Strobe
scan_mode	I	scan_mode
ipt_test_clk_se	I	scan_en
mddb_rd[15:0]	I	data bus

Table 18-1. DMA_ARB Pin List

Name	Type	Description
mdab[6:0]	O	address signal
mdrd	O	read signal
mdsel	O	module select signal
mirqn6	O	maskabale interrupt signal to DSP
mnmirq_b	O	non-maskable interrupt signal to DSP
pdar_grant0	O	Grant signal to DMA1
pdar_grant1	O	Grant signal to DMA2
pdar_pinmivse0	O	non-maskable source select signal to MDI
pdar_pinmivse1	O	non-maskable interrupt source select signal to DMA1
pdar_pinmivse2	O	non-maskable interrupt source select signal to DMA2
pdar_pivsse0	O	maskable interrupt source select signal to DMA1
pdar_pivsse1	O	maskable interrupt source select signal to DMA2
mddb_wr[15:0]	O	data bus

Chapter 19
Viterbi Accelerator (VIAC)

Revision History

Revision	Date	Author	Changes
1.0	10/08/99		Initial Release.
1.1	11/10/00	Arnab Kr. Mitra	Added module pin connectivity diagram and pinout table
1.2	2/19/01	Shannon Osgood	Made applicable updates for v1.9 release.
1.3	10/2/01	Biju varghese	Modified module pin connectivity diagram and pinout table for indy
1.4	01/4/02	Biju Varghese	Added DRA interface for the module
1.5	03/19/02	Manish Mittal	Added description for shared memory access and changed description for generation of contention signal. Also added the difference document table in the specs.
1.6	04/30/03	Manish Mittal	Updated per DDTS# DSPhl17239, DSPhl17325, and DSPhl17326.
1.7	05/07/03		Updated for LTE specification release.

19.1 Introduction

The Viterbi Accelerator (VIAC) is a peripheral module, that resides on the S-PMB bus of the DSP 56600 Core (henceforth called “DSP”). The VIAC is designed to accelerate performing the Viterbi Algorithms for GSM:

1. MLSE (Maximum Likelihood Sequential Estimation) for channel equalization
2. Convolutional decoding

The VIAC calculates a path metric corresponding to each state in a trellis, chooses the transition whose metric is the best and updates the path history. Calculations of a branch metric and traceback are executed by the DSP. The accelerator performs its function in parallelism and pipelining for providing high throughput.

The VIAC operates in lockstep with the DSP or independent to the DSP, using 2 internal DMA channels that share access, to a DPRAM, with the DSP via DRA interface .

The VIAC is used in 2 functions of the MODEM application: MLSE and Convolutional Decoding as shown in Figure 18-1.

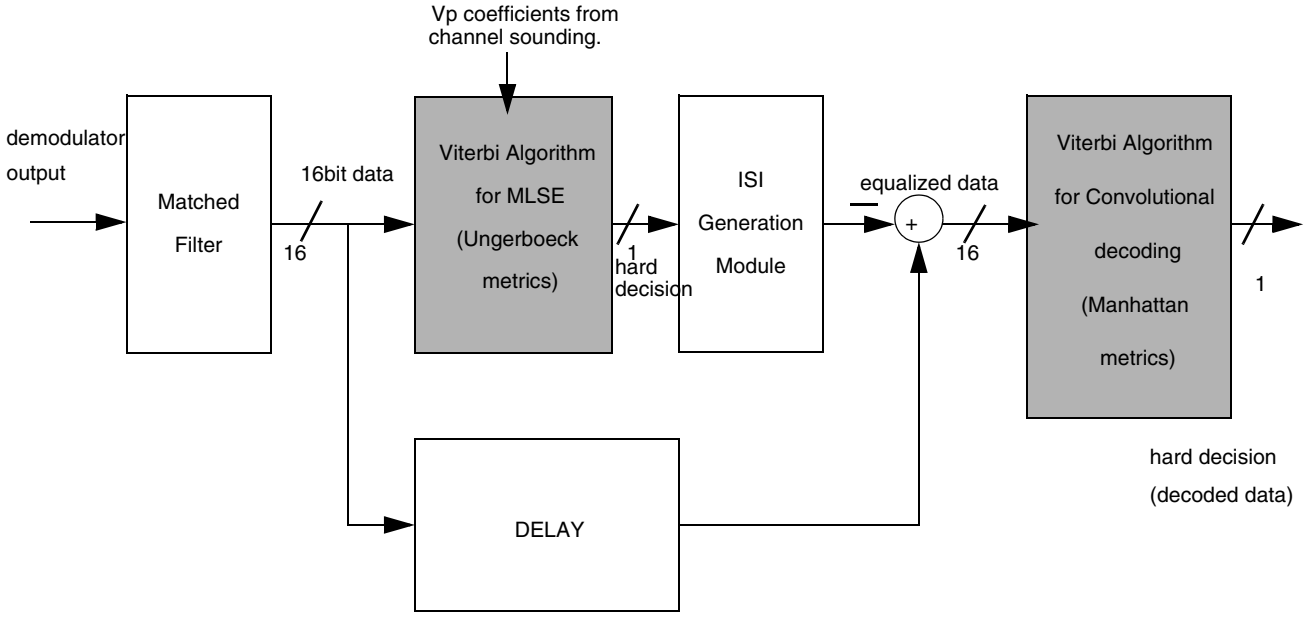


Figure 19-1. Using VIAC in MODEM procedure

19.2 Key Features

The following is a list of the VIAC's important features:

- Implements MLSE machine using Viterbi algorithm for channel equalization and Convolutional decoding
- Uses Ungerboeck metrics [1] for channel equalization
- Constraint length of 5 and 7 (16 or 64 Trellis states)
- Code rates of 1/2, 1/3 or 1/6
- Up to three polynomials for Convolutional decoding
- 22-bit Path Metric values for each of the 16 or 64 paths
- 16-bit Window Error Detection values for each of the 64 paths (as required for GSM Half-rate code)
- 2 operating modes:
- Channel Equalization mode
- Convolutional Decoding mode
- The VIAC operates either in lockstep mode or in independent mode, using 2 DMA channels
- The 2 DMA channels may access a limited sized Dual Ported RAM (henceforth called "DPRAM"). The acces of the DPRAM performed through Data Ram Arbiter interface (DRA)The physical starting address and the size of the DPRAM is defined in the Memory Map specification.
- DMA unpacking for channel decoding (for 4 and 8 bits packing)

19.3 Module Interface

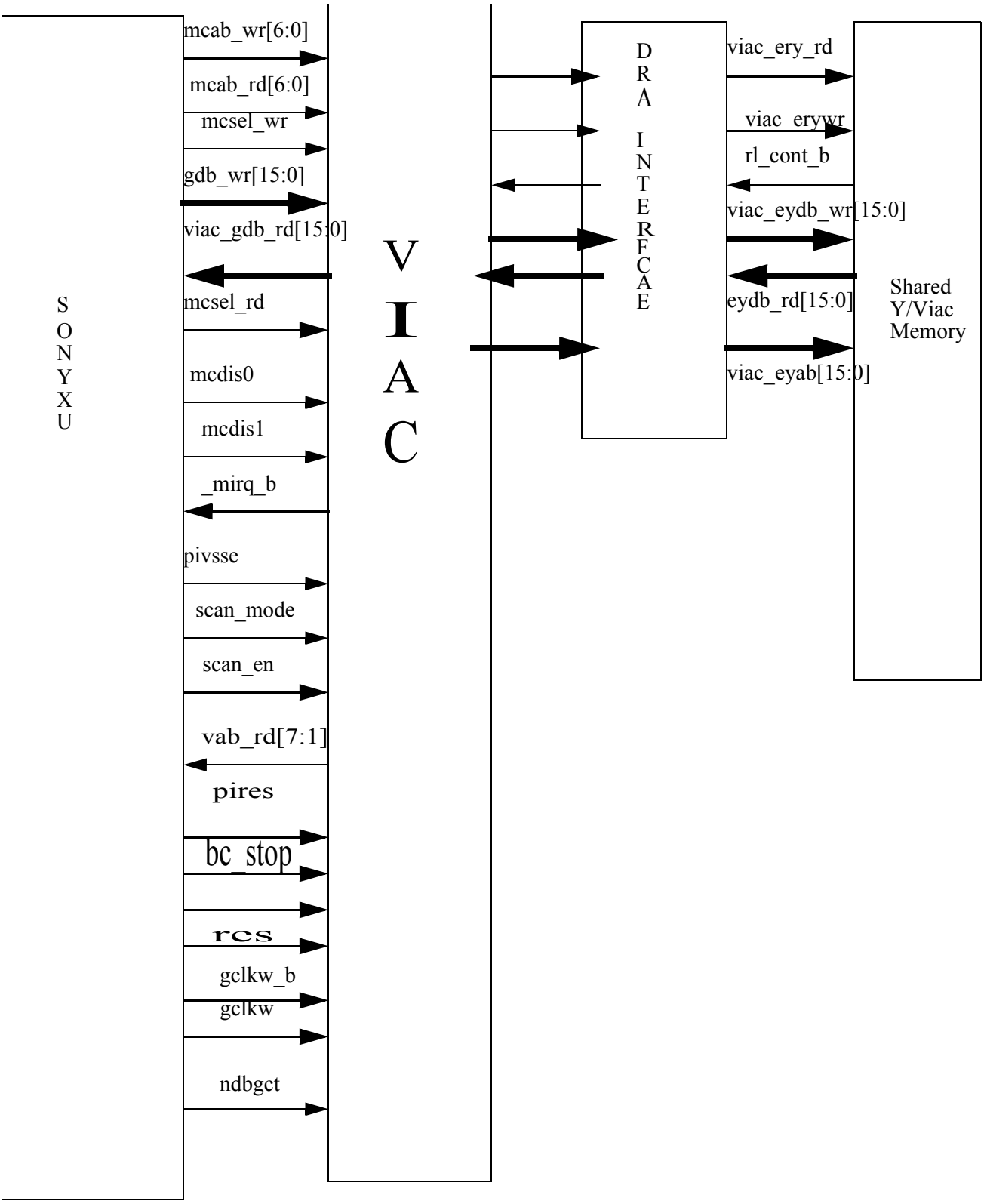


Figure 19-2. Viterbi Accelerator Interface Diagram

19.4 VIAC Pin Description

Table 19-1 lists the pins in the VIAC module.

Table 19-1. VIAC Module Pin List

Pin Name	Direction	Description
Core Access Signals		
mcab_wr[6:0]	Input	Core Write address bus
mcab_rd[6:0]	Input	Core Read address bus
mcsel_wr	Input	X or Y IO Write select, connected at the module instantiation level to one of MCSELX_WR or MCSELY_WR
mcsel_rd	Input	X or Y IO Read select, connected at the module instantiation level to one of MCSELX_RD or MCSELY_RD
gdb_wr[15:0]	Input	Core Global Write data bus
gdb_rd[15:0]	Output	Core Global Read data bus
mcdis0/mcdis1	Input	Core pipeline stall Indication (used in SPMBIF only, hidden for the user)
Interrupt Request Signals		
mirq_b	Output	Peripheral Module Interrupt Request. Connected at the module instantiation level to one of the 12 interrupt request line mirqn[11:0]. Active low.
pivsse	Input	Interrupt Vector Source Select. Connected at the module instantiation level to one of the 12 interrupt Vector Source Select line pivsse[11:0].
vab_rd[7:1]	Output	Interrupt Vector Bus.
Misc		
gclkw_b	Input	Inverted Raw clock forNew Custom Module
gclkw	Input	Raw clock for New Custom Module
pires	Input	Peripheral Module's Software reset.
bcstop	Input	Signal indicates core stop mode.It is asserted 4 cycles before gclkw goes idle.
ndbgctl	Input	The core is in 'DEBUG mode' indication.
res	Input	Peripheral Module's Hardware reset.
Memory Access Signals to DRA		
eydb_rd[15:0]	Input	VIAC read data bus

Table 19-1. VIAC Module Pin List

Pin Name	Direction	Description
viac_eydb_wr[15:0]	Output	VIAC write data bus
viac_eyab [15: 0]	Output	VIAC address
viac_eryrd	Output	VIAC read control
viac_erywr	Output	VIAC write control
rl_cont_b	Input	This signal gets asserted (for one and half cycle) whenever the VIAC accesses the shared memory. The signal will remain asserted for more than one & half cycle when VIAC's dma access to memory conflicts with core or DMA access to the same memory module.
VIAC Scan Interface		
scan_mode	Input	Scan mode signal
scan_en	Input	Scan enable source signal (for gating with scan mode).

19.5 Performance analysis

The number of DSP clocks required to decode one bit is growing exponentially depending on the constraint length factor (linear with number of states). The VIAC pipelines the input/output operation with the butterflies calculations.

The throughput of the VIAC is:

$$T_{stage} = \frac{N_{ts} \cdot T_{acs}}{2}$$

where N_{ts} - the number of trellis states.

T_{acs} :

2 clocks are required to perform one ACS (Add Compare Select) butterfly.

The input/output is done in parallel and adds to the latency of the total calculations:

$$T_{latency} = T_{IO} + N_{stages}\left(\frac{N_{ts} \cdot T_{acs}}{2}\right)$$

Where T_{IO} is the added latency for the input/output operations:

2 clocks are required to perform input/output operations in channel equalization mode.

3 clocks are required to perform input/output operations in convolutional decoding mode, the code rate of 1/2.

$(4 + N_{ts}/16)$ clocks are required to perform input/output operations in convolutional decoding mode, the code rate of 1/3, 1/6.

19.5.1 VIAC's Organization

The VIAC is comprised of several basic blocks as illustrated in Figure 19-3. The S-PMB interface is the VIAC's interface with the DSP. The DMA block, including 2 DMA channels, is used to enable the VIAC and the DSP to operate independently. The Input register is used while in equalization to feed the VIAC with the matched filter's output. The Branch Metric RAM is used in equalization to save the Ungerboeck metrics and in Convolutional decoding to feed the VIAC with the Manhattan metrics. The Path Metric RAM is used to save the scores given to each path, reflecting its probability. The WED RAM is used to save the minimum decision difference in a window, for each trellis path. The Path Metric Update Unit is used for calculating the Viterbi butterfly, performing ACS (add compare select) and WED update functions as illustrated in Section 19.5.2 and Section 19.5.3 on page 19-8. The Data out register is used to pack the butterflies decisions bits to be read by the DSP.

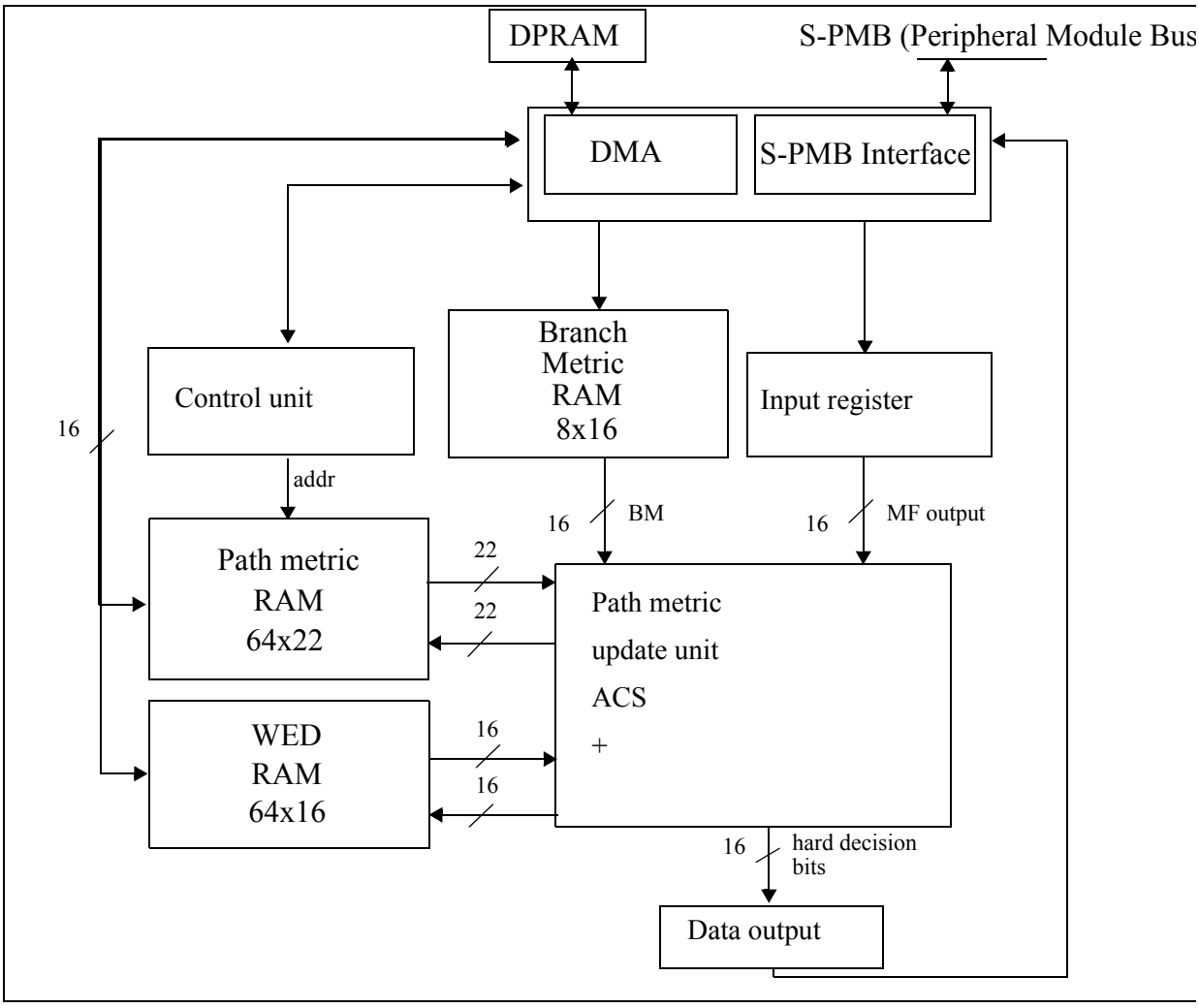


Figure 19-3. Viterbi Accelerator Block Diagram

19.5.2 ACS - Add Compare Select

The Add Compare Select is the basic element in Viterbi Butterfly calculation. The VIAC includes a dedicated hardware for this calculation illustrated in Figure 19-4.

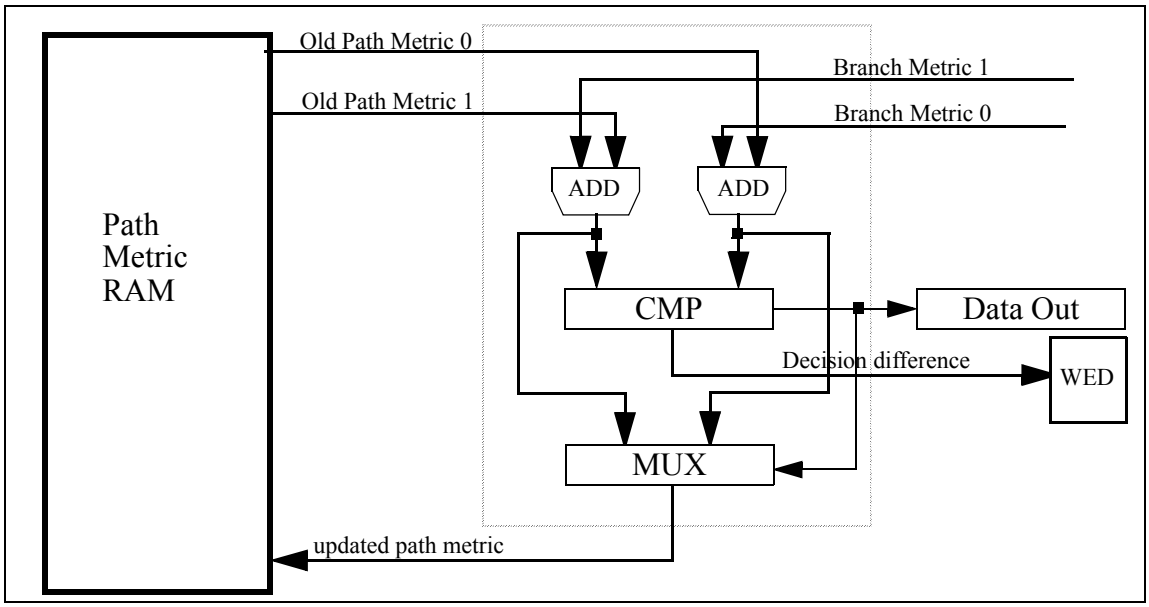


Figure 19-4. ACS - Add Compare Select Function

19.5.3 WED - Window Error Detection

The Window Error Detection hardware calculates the minimum decision difference in a window, for each trellis path as illustrated in Figure 19-5.

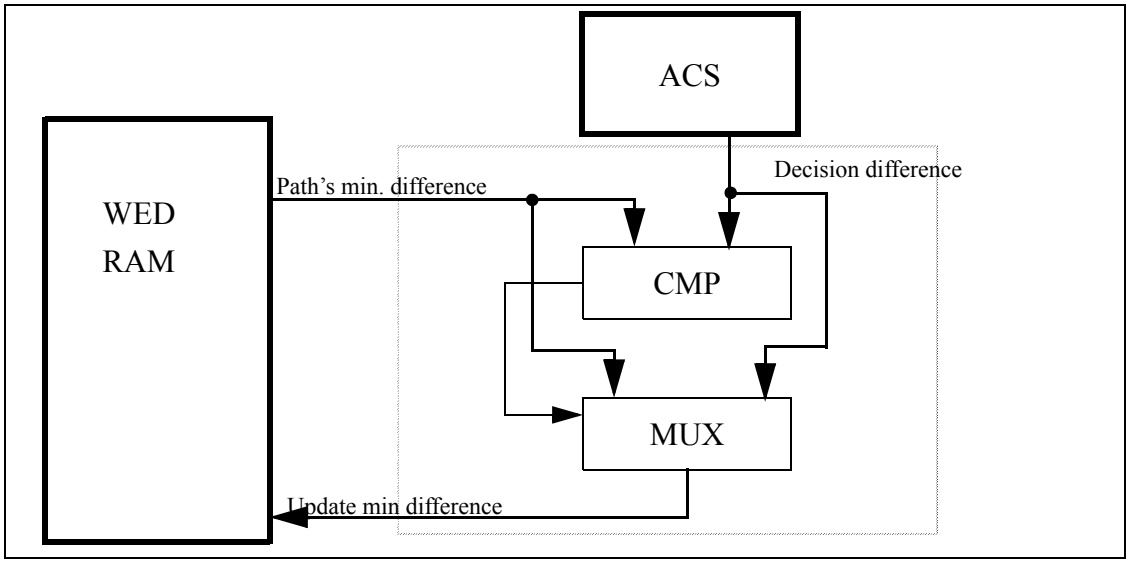


Figure 19-5. WED - Window Error Detection Function

19.5.4 VIAC's pipeline

The VIAC may operates either in lockstep with the DSP or independent to the DSP. When the VIAC operates in lockstep mode, the DSP has to feed the VIAC with input parameters and read the VIAC's outputs once every trellis loop. When the VIAC operates in independent mode, the DMA input channel reads the input parameters from the DPRAM and the DMA output channel writes the results to the DPRAM every trellis loop, without interfering with the DSP operation. It is the responsibility of the DSP, to save the input parameters in the DPRAM before the VIAC reads them.

The following sections describes the VIAC pipeline in lockstep mode and in independent mode.

19.5.4.1 VIAC's pipeline in lockstep mode

The VIAC performs the Viterbi basic loop on the input parameters written by the DSP, and generates decision bits to be read by the DSP. The VIAC includes input and output buffers which are both double-buffered to obtain full pipelining. While the previous written input parameters are processed, the current input parameters can be written without stalling the VIAC. While the current results are read by the DSP, the VIAC continues to generate the next results.

In equalization, the input parameter on each loop, is the match filter's output which is written to a double buffered input register (refer to Table 19-3 on page 19-28). In convolutional decoding, the input parameters on each loop, are the Manhattan metrics, which are fed into a double buffered fifo (refer to Table 19-4). The depth of the fifo relates to the code rate and reflects the number of needed input parameters (2 for 1/2 code rate, 4 for 1/3 or 1/6 code rate).

The Figure 19-6, illustrates the logical structure of the input buffers in the various configurations. In equalization the input buffer is the VIDR. In convolutional decoding in code rate of 1/2, the input buffer is the 2 entries VBMR fifo. In convolutional decoding in code rate of 1/3 or 1/6, the input buffer is the 4 entries VBMR fifo.

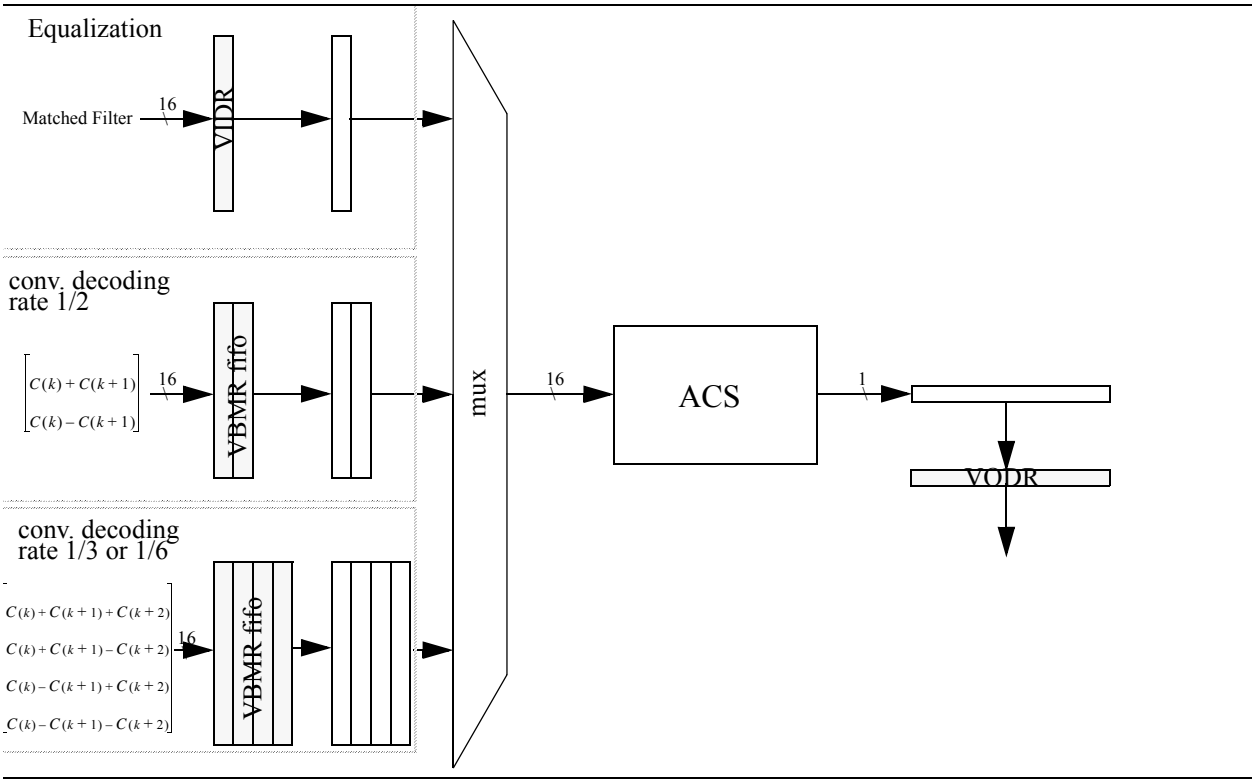


Figure 19-6. Feeding VIAC in lockstep mode

The VIAC generates decision bits in a constant rate of 1 bit per DSP clock.

In equalization, the basic loop period is 16 clocks and the number of result bits, in each loop, is 16. Therefore the DSP has to read the 16 bits output register (VODR), once per loop. Each loop the DSP writes the matched filter's output into the input register (VIDR). Figure 19-7, illustrates the VIAC's basic pipeline in this mode.

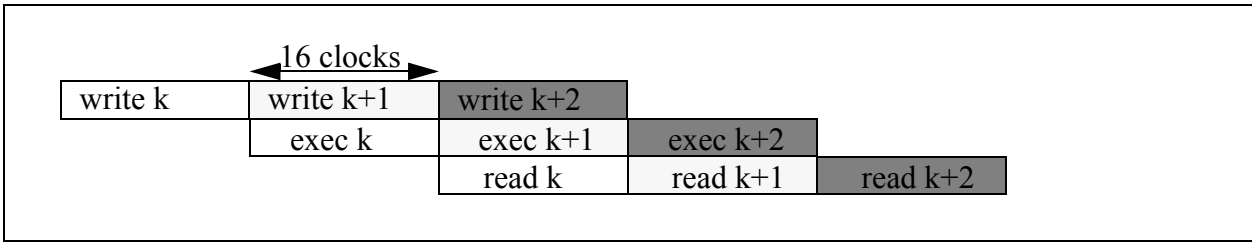


Figure 19-7. VIAC's Pipeline in equalization, in lockstep mode

In convolutional decoding in constraint length of 5, when code rate is 1/2, the basic loop period is 16 clocks and the number of result bits, in each loop, is 16. Therefore the DSP has to read the 16 bits output register (VODR), once per loop. Each loop the DSP writes the 2 Manhattan metrics $\begin{bmatrix} C(k)+C(k+1) \\ C(k)-C(k+1) \end{bmatrix}$ into the VBMR fifo register. Figure 19-8, illustrates the VIAC's basic pipeline in this mode.

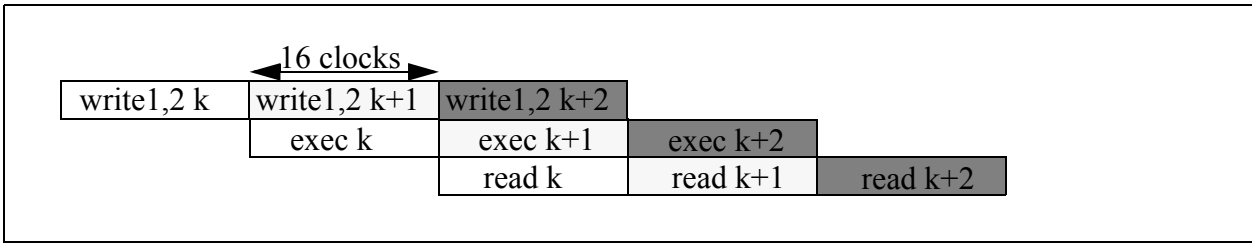


Figure 19-8. VIAC's Pipeline in Convolutional decoding, constraint length of 5, 1/2 code rate, in lockstep mode

In convolutional decoding in constraint length of 5, when code rate is 1/3 or 1/6, the basic loop period is 16 clocks and the number of result bits, in each loop, is 16. Therefore the DSP has to read the 16 bits output register (VODR), once per loop. Each loop the DSP writes the 4 Manhattan metrics $\begin{bmatrix} C(k)+C(k+1)+C(k+2) \\ C(k)+C(k+1)-C(k+2) \\ C(k)-C(k+1)+C(k+2) \\ C(k)-C(k+1)-C(k+2) \end{bmatrix}$ into the VBMR fifo register.

Figure 19-9, illustrates the VIAC's basic pipeline in this mode.

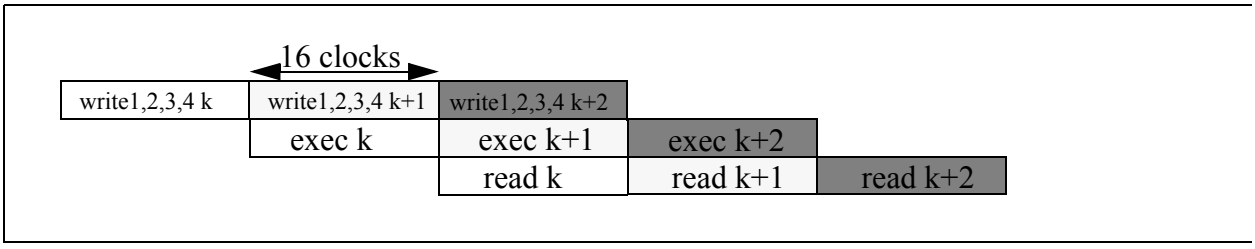


Figure 19-9. VIAC's Pipeline in Convolutional decoding, constraint length of 5, 1/3 or 1/6 code rate, in lockstep mode

In convolutional decoding in constraint length of 7, the basic loop period is 64 clocks and the number of result bits, in each loop, is 64. Therefore the DSP has to read the 16 bits output register (VODR), 4 times per loop.Each loop the DSP writes the 4 Manhattan metrics $\begin{bmatrix} C(k)+C(k+1)+C(k+2) \\ C(k)+C(k+1)-C(k+2) \\ C(k)-C(k+1)+C(k+2) \\ C(k)-C(k+1)-C(k+2) \end{bmatrix}$ into the VBMR fifo register.

Figure 19-10, illustrates the VIAC's basic pipeline in this mode.

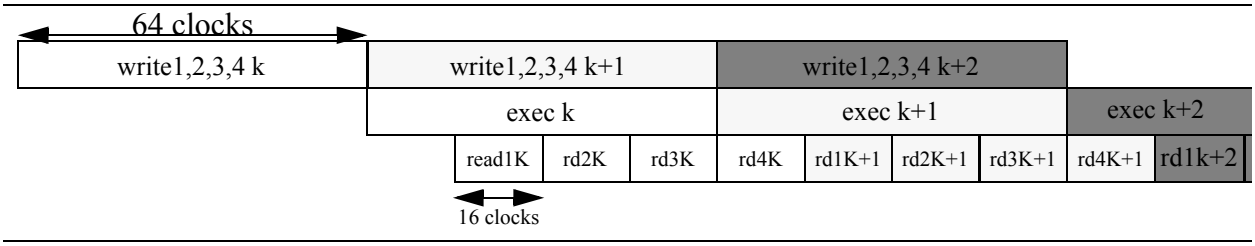


Figure 19-10. VIAC's Pipeline in convolutional decoding in constraint length of 7, 1/3 or 1/6 code rate, in lockstep mode

19.5.4.2 VIAC's pipeline in independent mode

The VIAC performs the Viterbi basic loop on the input parameters previously written by the DSP to the DPRAM, and generates decision bits to be read later by the DSP from the DPRAM. The VIAC includes 2 DMA channels: DMA input channel, that is used to read the input parameters from the DPRAM and DMA output channel, that is used to write the results into the DPRAM through DRA Interface . The VIAC includes input and output buffers which are both double-buffered to obtain full pipelining. While the previous written input parameters are processed, the current input parameters are read by the DMA input channel from the DPRAM without stalling the VIAC. While the current results are written by the DMA output channel to the DPRAM, the VIAC continues to generate the next results.

In equalization, the input parameter on each loop, is the match filter's output. In convolutional decoding, the input parameters on each loop, are the Manhattan metrics. In both cases the input parameters are read by the DMA input channel from the DPRAM. Note that while in convolutional decoding the number of parameters which are read by the DMA, in each loop, relates to the code rate (2 for 1/2 code rate, 4 for 1/3 or 1/6 code rate).

The Figure 19-11, illustrates the logical structure of the input buffers in the various configurations. In equalization the input buffer is the VIDR. In convolutional decoding in code rate 1/2, the input buffer is the 2 entries VBMR fifo. In convolutional decoding in code rate 1/3 or 1/6, the input buffer is the 4 entries VBMR fifo.

During trellis processing phase the DSP is not involved with the VIAC operation. Any DSP access to the same 1/4K DPRAM, which is currently accessed by the DMA channels will stall the DMA and therefore may stall the VIAC.

Whenever DSP core (or DMA) and the VIAC both try to access the memory (DPRAM-Dual Port RAM) through DRA, DRA grants the memory access to DSP first and VIAC later. Similarly If VIAC and DMA both try to access the memory (DPRAM) through DRA, DRA grants the momory access to VIAC first.

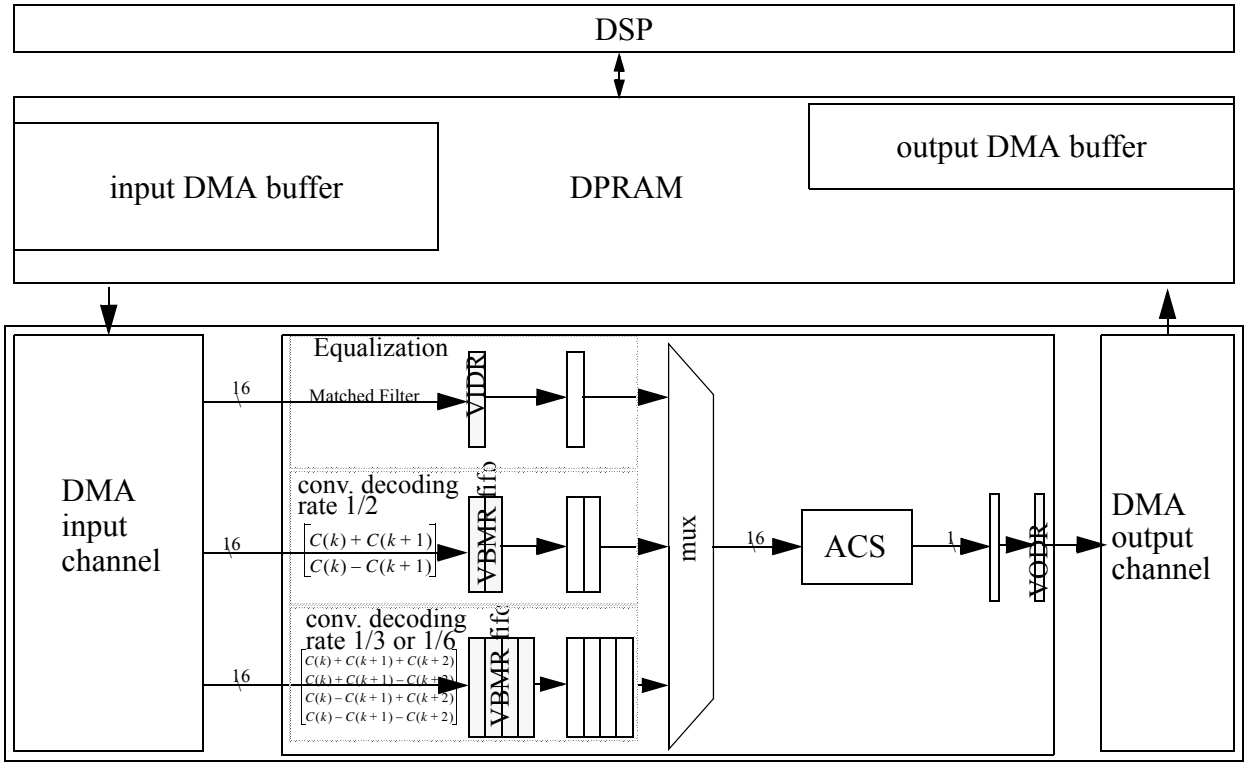


Figure 19-11. VIAC's DMA accesses in independent mode

The VIAC generates decision bits in a constant rate of 1 bit per DSP clock.

In equalization, the basic loop period is 16 clocks and the number of result bits, in each loop, is 16. Therefore the DMA output channel writes the 16 bits output register (VODR) to the DPRAM, once per loop. Each loop the DMA input channel reads the matched filter's output from the DPRAM, into the input register (VIDR). Figure 19-12, illustrates the VIAC's basic pipeline in this mode.

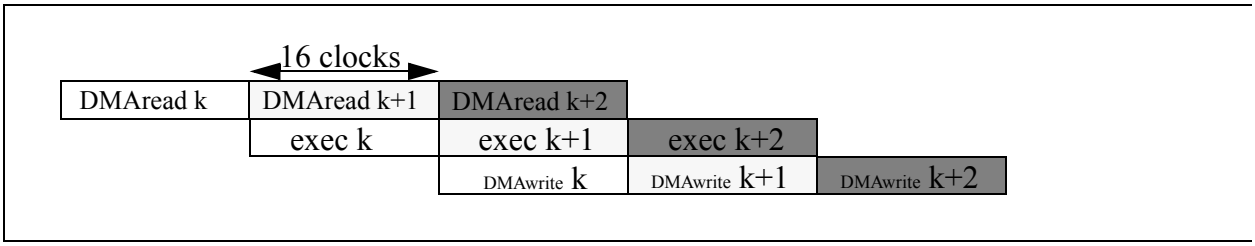


Figure 19-12. VIAC's Pipeline in Equalization, in independent mode

In convolutional decoding in constraint length of 5, when code rate is 1/2, the basic loop period is 16 clocks and the number of result bits, in each loop, is 16. Therefore the DMA output channel writes the 16 bits output register (VODR), once per loop. Each loop the DMA input channel reads the 2 Manhattan metrics $\begin{bmatrix} C(k)+C(k+1) \\ C(k)-C(k+1) \end{bmatrix}$ into the VBMR fifo register. Figure 19-13, illustrates the VIAC's basic pipeline in this mode.

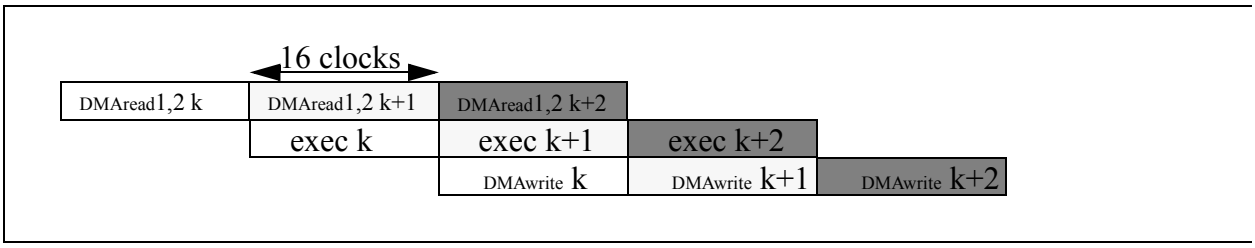


Figure 19-13. VIAC's Pipeline in Convolutional decoding, constraint length of 5, 1/2 code rate, in independent mode

In convolutional decoding in constraint length of 5, when code rate is 1/3 or 1/6, the basic loop period is 16 clocks and the number of result bits, in each loop, is 16. Therefore the DMA output channel writes the 16 bits output register (VODR), once per loop. Each loop the DMA input channel reads the 4 Manhattan metrics $\begin{bmatrix} C(k)+C(k+1)+C(k+2) \\ C(k)+C(k+1)-C(k+2) \\ C(k)-C(k+1)+C(k+2) \\ C(k)-C(k+1)-C(k+2) \end{bmatrix}$ into the VBMR fifo register.
Figure 19-14, illustrates the VIAC's basic pipeline in this mode.

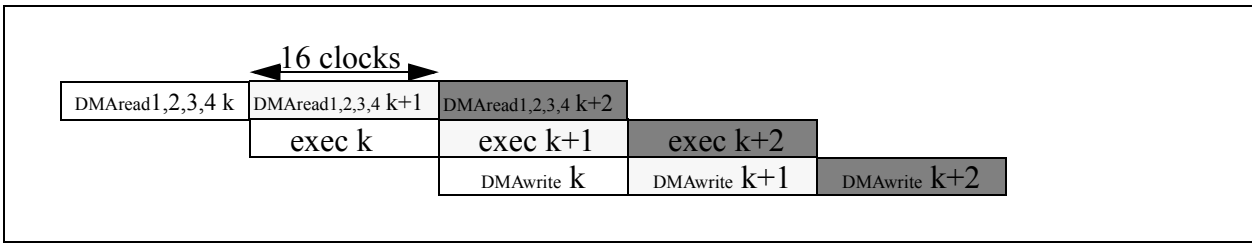


Figure 19-14. VIAC's Pipeline in Convolutional decoding, constraint length of 5, 1/3 or 1/6 code rate, in independent mode

In convolutional decoding in constraint length of 7, the basic loop period is 64 clocks and the number of result bits, in each loop, is 64. Therefore the DMA output channel writes the 16 bits output register (VODR), 4 times per loop. Each loop the DMA input channel reads the 4 Manhattan metrics $\begin{bmatrix} C(k)+C(k+1)+C(k+2) \\ C(k)+C(k+1)-C(k+2) \\ C(k)-C(k+1)+C(k+2) \\ C(k)-C(k+1)-C(k+2) \end{bmatrix}$ into the VBMR fifo register.
Figure 19-15, illustrates the VIAC's basic pipeline in this mode.

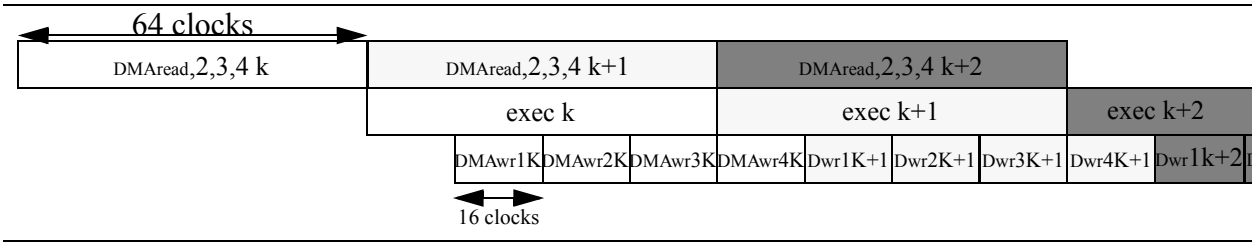


Figure 19-15. VIAC's Pipeline in convolutional decoding in constraint length of 7, in independent mode

19.5.5 Path Metric RAM access using VPMAR FIFO

The Path Metric RAM is updated every Viterbi butterfly by the ACS. Its contents can also be accessed by the DSP. When the DSP accesses the Path Metric RAM the RAM is organized as a FIFO.The Path Metric RAM can be read and written using a 22 bits fifo called VPMAR. This fifo is accessed using two registers called VPMARA,VPMARB. Both registers must be accessed consecutively starting with VPMARA and then VPMARB to obtain one fifo entry. The VIAC includes an internal pointer to the PMRAM holding the current fifo entry to be accessed. This pointer is cleared on reset and upon START command.

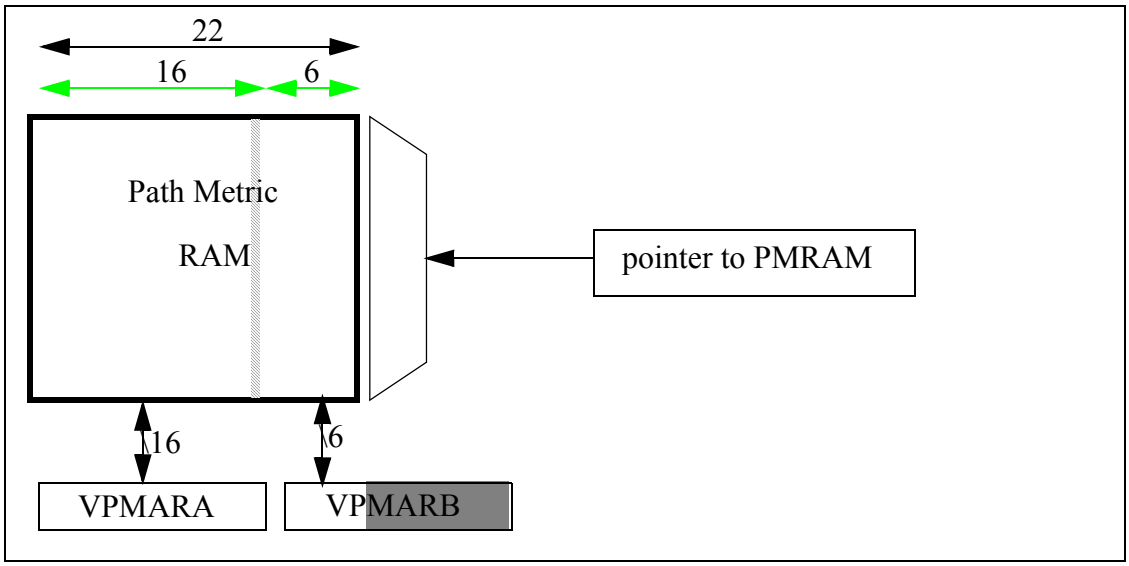


Figure 19-16. VPMAR FIFO

Each time VPMARB is accessed both VPMARA and VPMARB are either written into the RAM or loaded with data from the RAM. The internal pointer to the PMRAM is increased each time VPMARB is accessed. Note the pointer does not change when accessing VPMARA.

Note that all the FIFO must be either written or read consecutively starting from a cleared pointer:

- Reading or writing the FIFO, the VIAC must be in the WAIT state which occurred due to an end of stage or an end of trellis (refer to Section 19.7.2).
- Writing to the FIFO, the PMI bit in VMR register must be set (refer to Table 19-8, PMI bit 5), before executing the START command. Any attempt to write to the fifo without meeting the above conditions is erroneous.
- Note that the VIAC includes a single FIFO pointer. Therefore the FIFO should be read or written consecutively. The DSP may not write the FIFO in the middle of reading it and vice versa.
- Note that It is not permissible to initialize the Path Metric Ram with values which differ from each other by a size which is bigger then the possible difference (Maximum is limited to 6×2^{17}).

19.5.6 The Viterbi butterfly implementation

The bits enter the Trellis state via the Least Significant Bit side (i.e. from right to left). As an example, the 16-state trellis (constraint length equals 5) is assumed. Let w, x, y, z denote a binary digit such as ‘wxyz’ represent any state in the trellis. The Viterbi butterfly is then defined as a transition from the present two states of ‘0xyz’ and ‘1xyz’ to the next two states of ‘xyz0’ and ‘xyz1’ as shown in Figure 19-17.

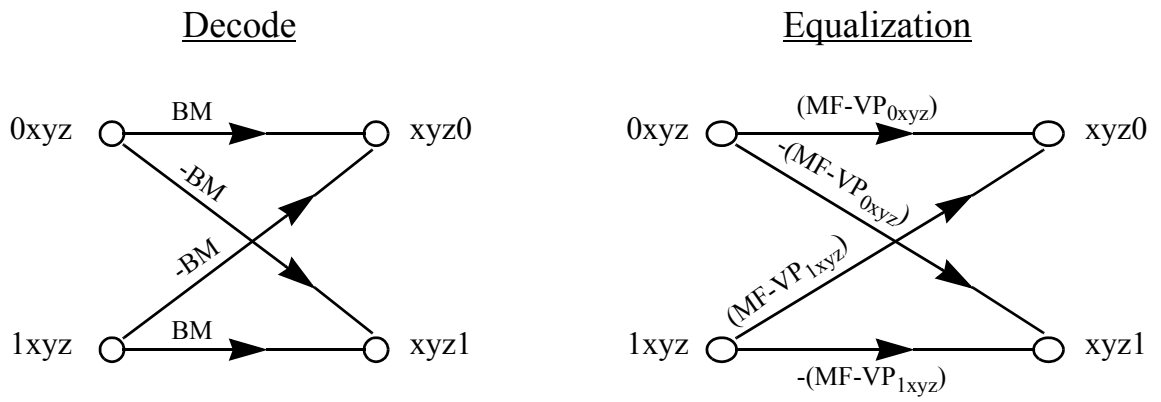


Figure 19-17. Viterbi Butterfly Structure

In Channel Decoding, a Branch Metric (BM) value is evaluated for each butterfly. In general, BM is a weighted value of the received data symbols with reference to the expected convolutional encoded bits of that particular state.

In Equalization, the branch metric for each transition within a Viterbi butterfly is a function of the Matched Filter output (MF) and the L-Metric Viterbi Parameters (VP). The VP values come from the channel sounding. After extracting the channel impulse response coefficients via a cross correlation process (also referred to as S-Parameters and used as the ISI FIR coefficients), the VP value for a particular state 'wxyz' is usually calculated as follows:

$$VP(w,x,y,z) = (-1)^{w} \cdot S_4 + (-1)^{x} \cdot S_3 + (-1)^{y} \cdot S_2 + (-1)^{z} \cdot S_1$$

The VP values calculation is done by software within the DSP core. Notice the symmetrical property of the VP values, $VP(w,x,y,z) = -VP(w,x,y,z)$, thus requiring the storage of only half of the L-metric table in the VP RAM.

19.5.6.1 Handling overflow in path metrics

The VIAC operates in arithmetic of 22 bits. The following assumptions lead to the conclusion that 22 bits are enough for holding the accumulated metrics:

- One wants to be able to order all the metrics according to their magnitude by the end of each trellis stage.
- Within an arbitrary stage the differences between the path metrics is limited to ((constraint length +1) * the maximum size of the branch metric).
- The Equalization input parameters are limited to 16 bits. Therefore, after adding the Ungerboeck metrics, the branch metrics are limited to 17 bits. Note that the branch metrics of Channel Decoding are limited to 16 bits.
- Considering the above assumptions: Within an arbitrary stage the path metrics all reside within a window whose maximum size is smaller than 6×2^{17} .

The path metrics window whose maximum size is 6×2^{17} circles within a larger window whose size is determined by the number of bits of the arithmetic. Each time the path metrics window resides completely within the upper quarter of the large window, it is wrapped around by the VIAC to the lower quarter as described in Figure 19-18 on page 19-16.

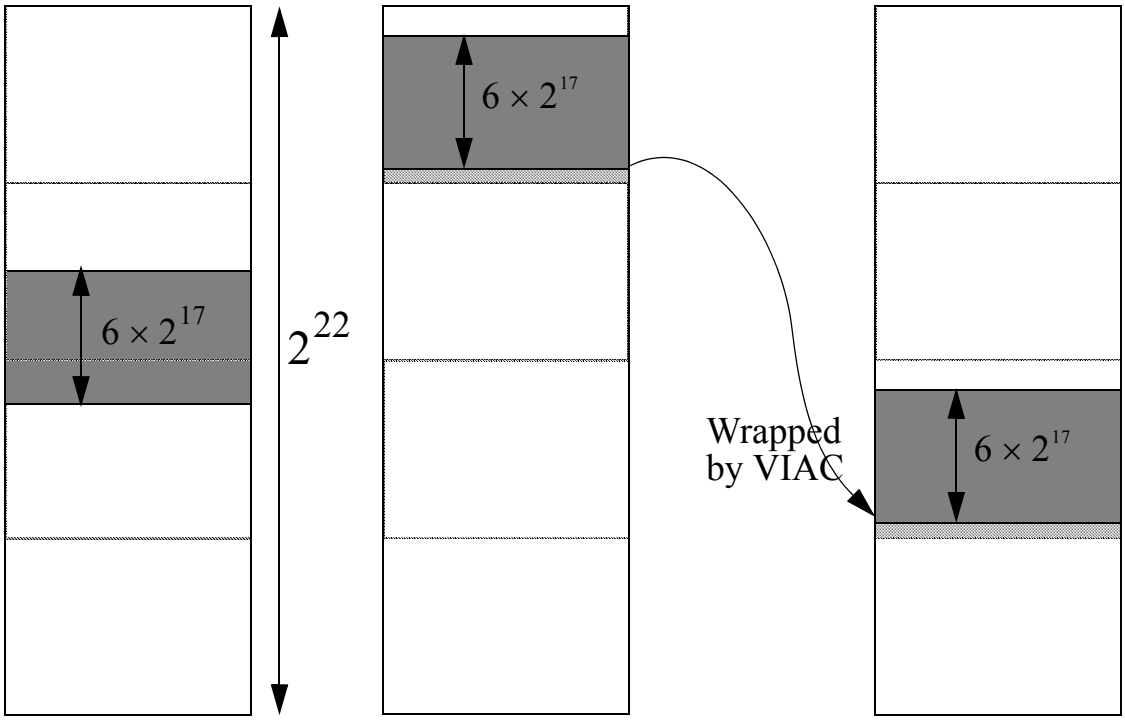


Figure 19-18. Wrap around handles overflow of path metrics

19.5.7 DMA buffers organization

The VIAC includes 2 DMA channels which share access to a DPRAM, with the DSP. These 2 DMA channels are used only while in independent mode. The DMA input channel is used to feed the VIAC with input parameters read from the DPRAM. The DMA output channel is used to save the results in the DPRAM.

In Channel decoding VIAC, while in DMA mode, provides an optional unpacking. The DSP may provide the input parameters in 4 or 8 bits each, packing them sequentially as described in the following figures. The DMA input channel unpacks the read parameters and performs sign extension to 16 bits. In unpacking 4 bits mode the VIAC unpacks each of the four 4 bit nibbles into four 16 bits words, performing sign extension on the most significant bits. In unpacking 8 bits mode the VIAC unpacks each of the two 8 bit bytes into two16 bits words, performing sign extension on the most significant bits.

Each DMA channel contains a base address register which points to the starting address of the DMA buffer. This buffer is accessed sequentially by the DMA channel, according to the VIAC's demand for data. The amount of input parameters or results while executing the Viterbi algorithm differentiates according to the selected configuration. The following illustrations describes the DMA buffers organization, according to the selected configuration:

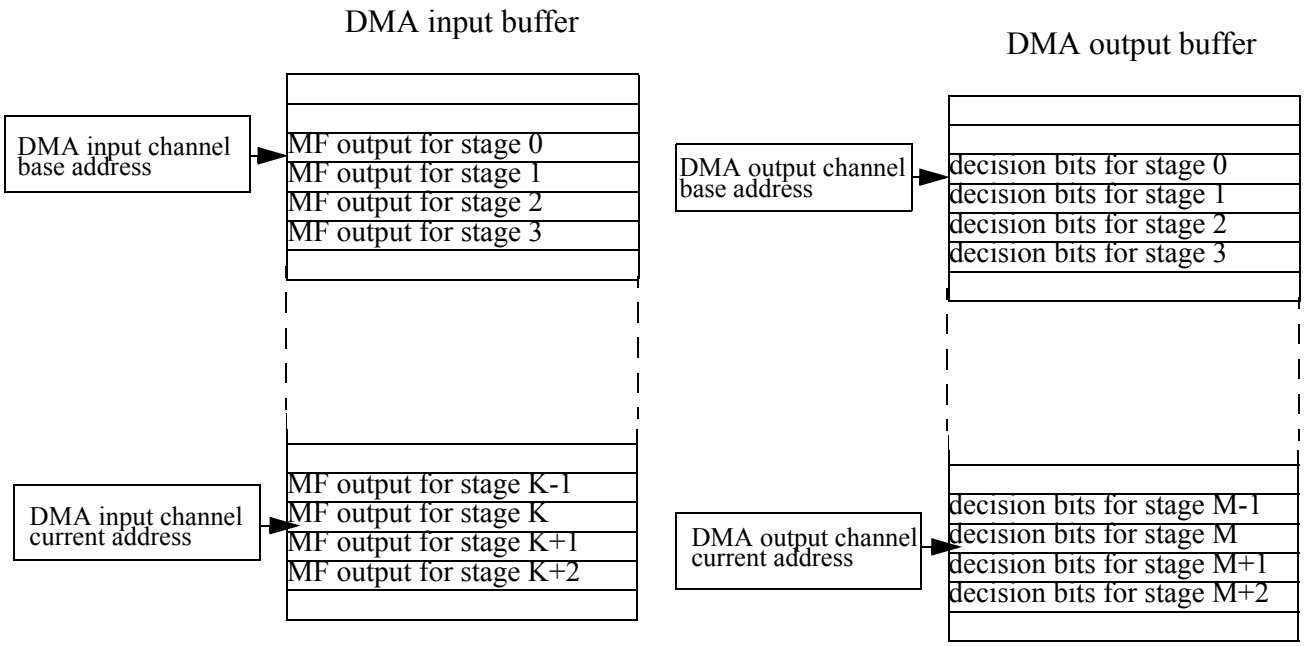


Figure 19-19. DMA buffers organization, in equalization

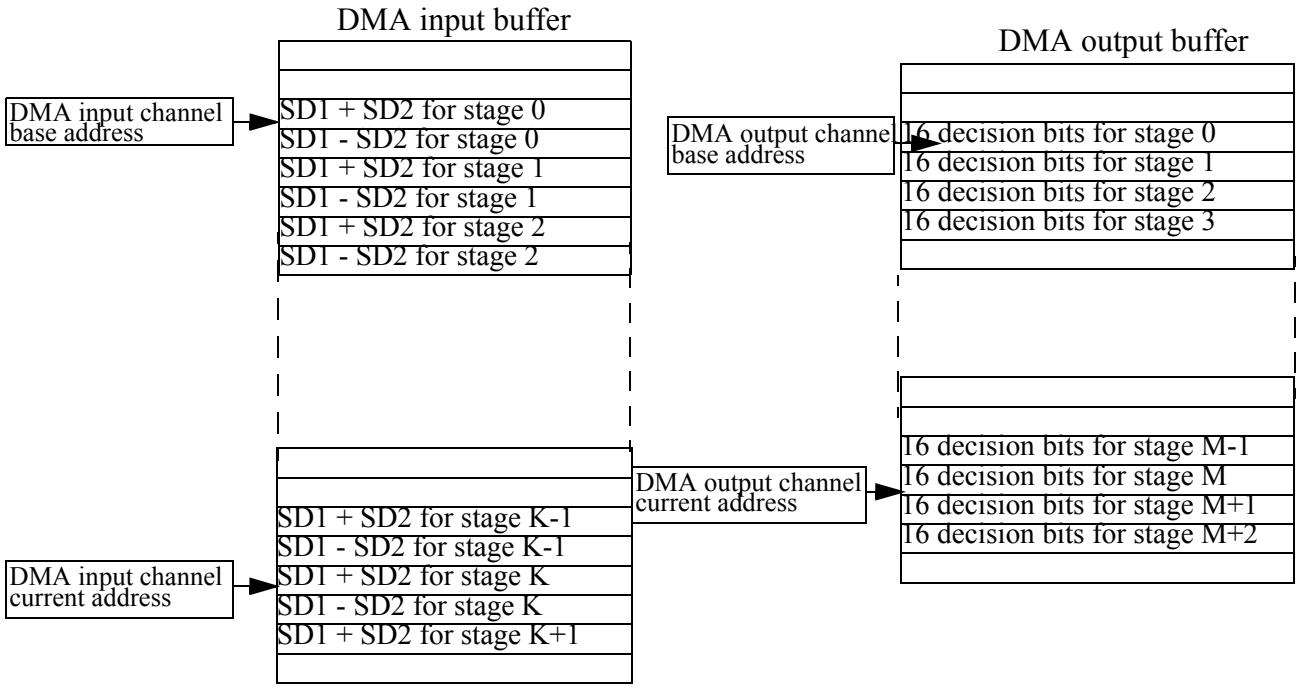


Figure 19-20. DMA buffers organization, in convolutional decoding, code rate 1/2, constraint length of 5, no packing

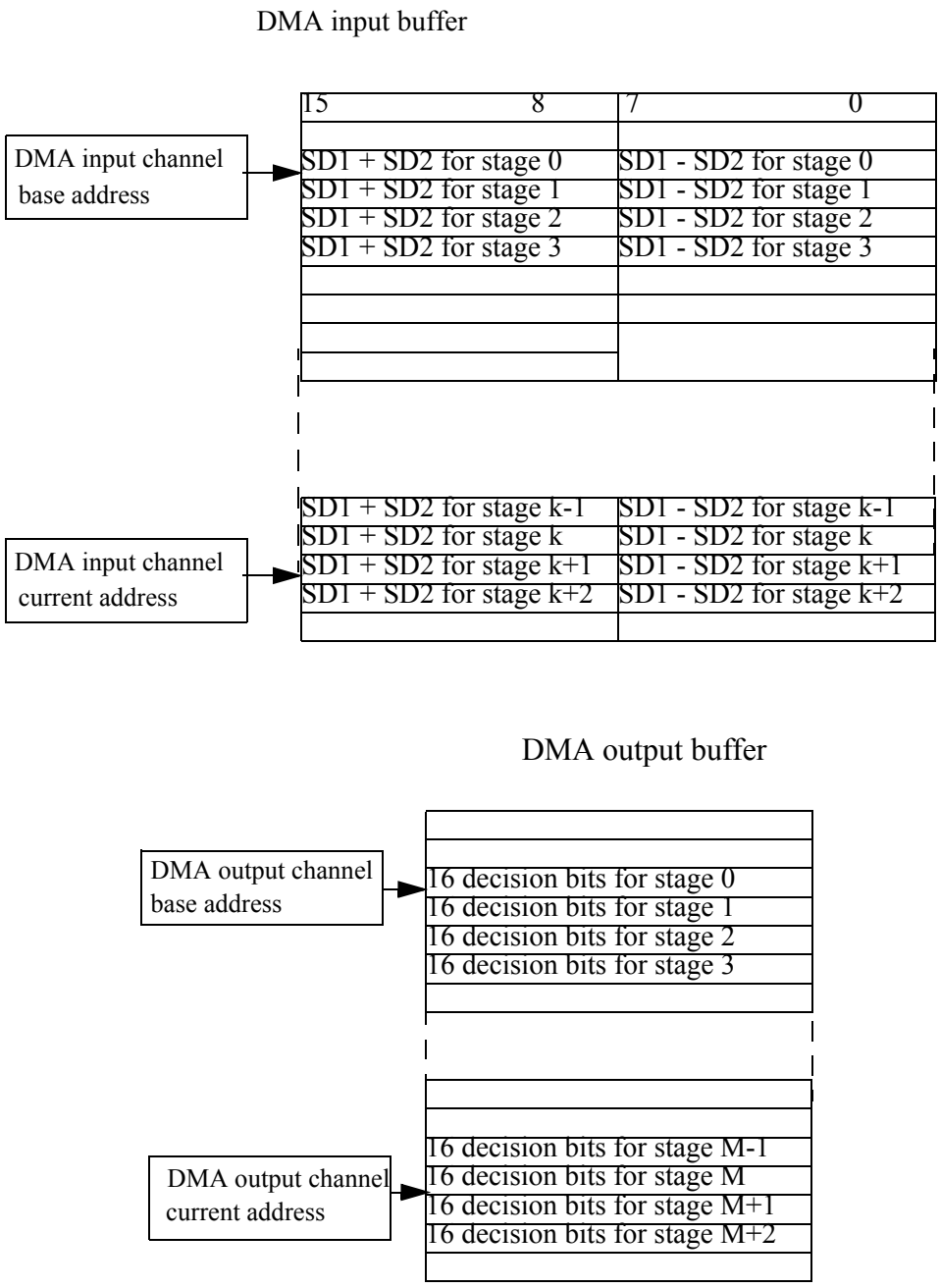


Figure 19-21. DMA buffers organization, in convolutional decoding, code rate 1/2, constraint length of 5, packing of 8 bits

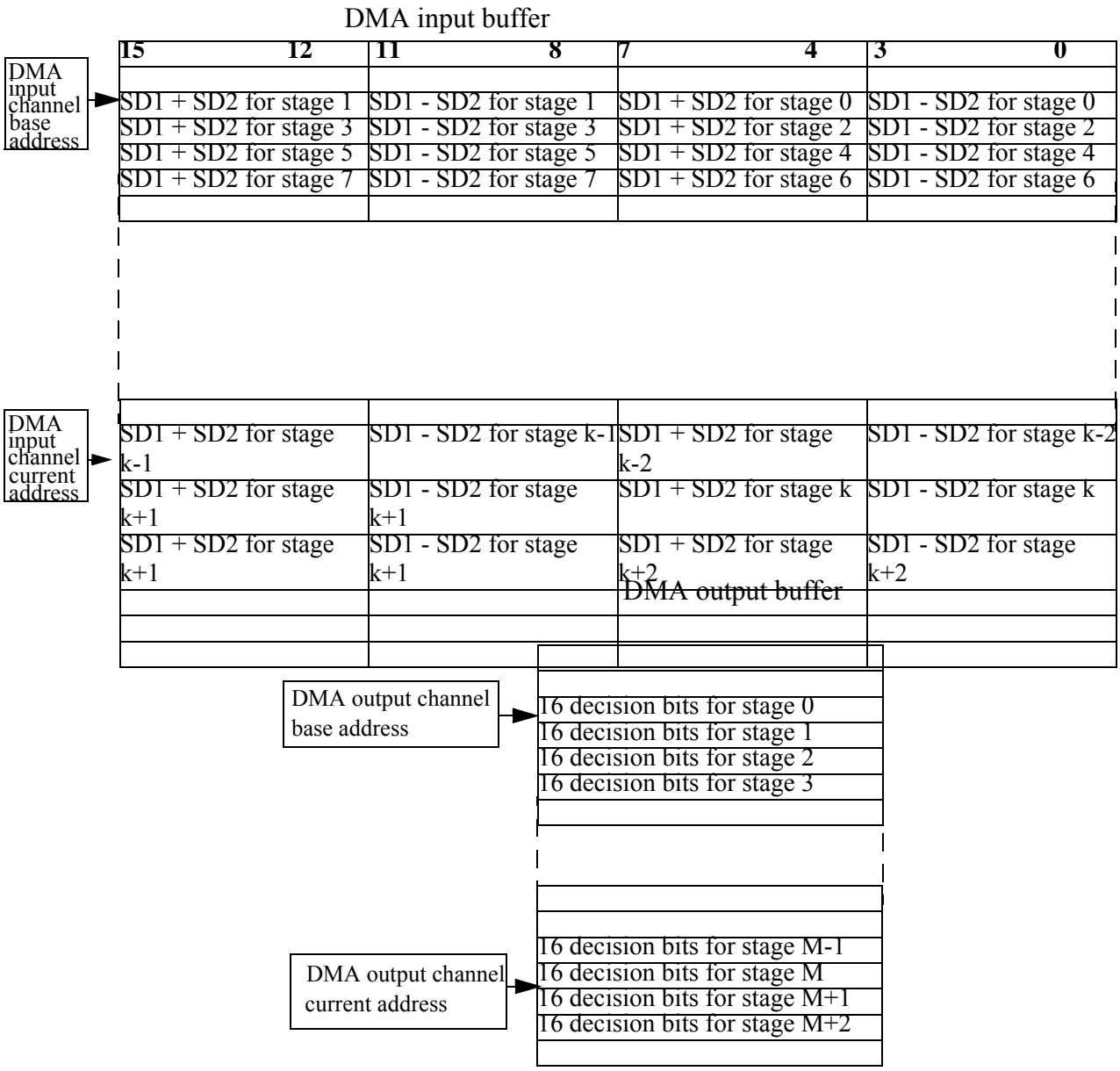


Figure 19-22. DMA buffers organization, in convolutional decoding, code rate 1/2, constraint length of 5, packing of 4 bits

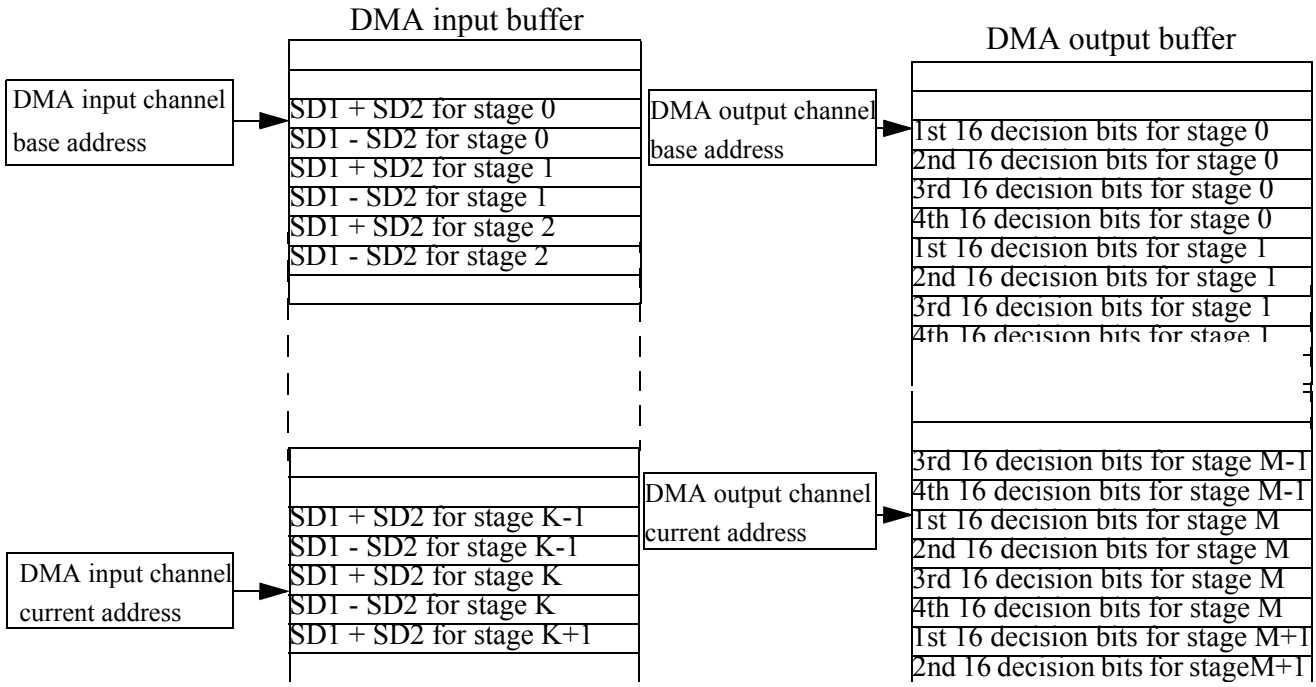


Figure 19-23. DMA buffers organization, in convolutional decoding, code rate 1/2, constraint length of 7, no packing

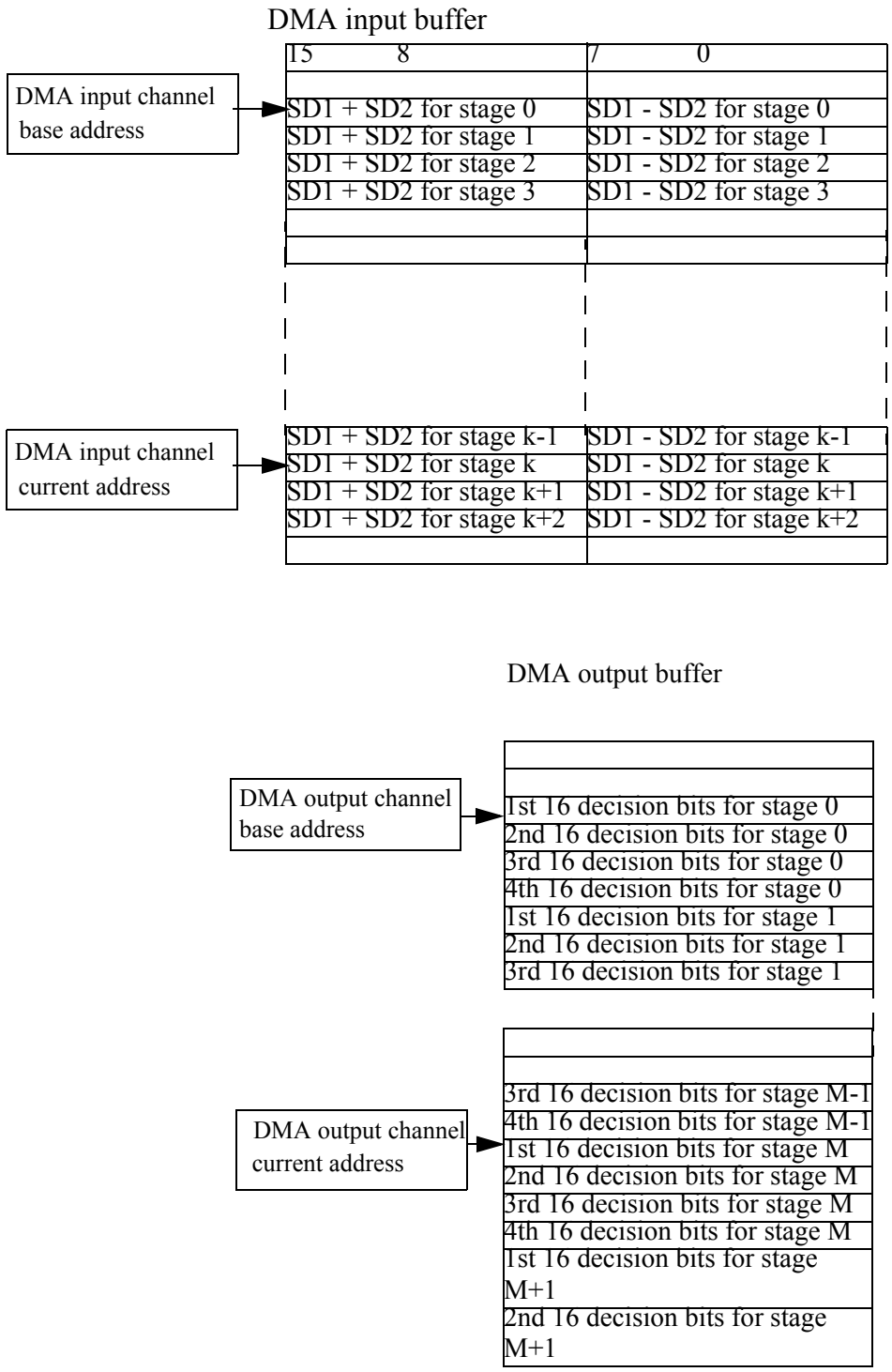


Figure 19-24. DMA buffers organization, in convolutional decoding, code rate 1/2, constraint length of 7, packing of 8 bits

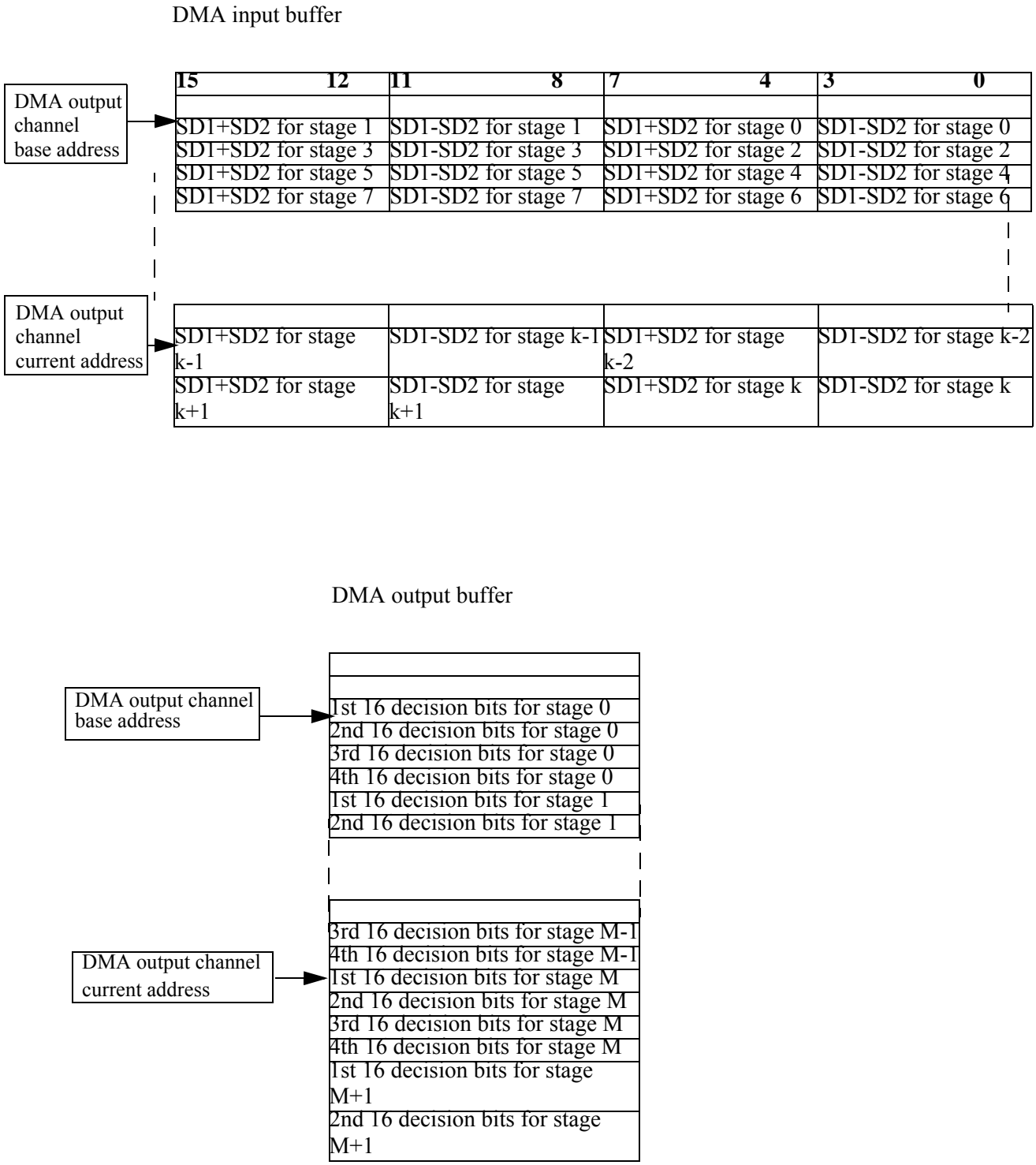


Figure 19-25. DMA buffers organization, in convolutional decoding, code rate 1/2, constraint length of 7, packing of 4 bits

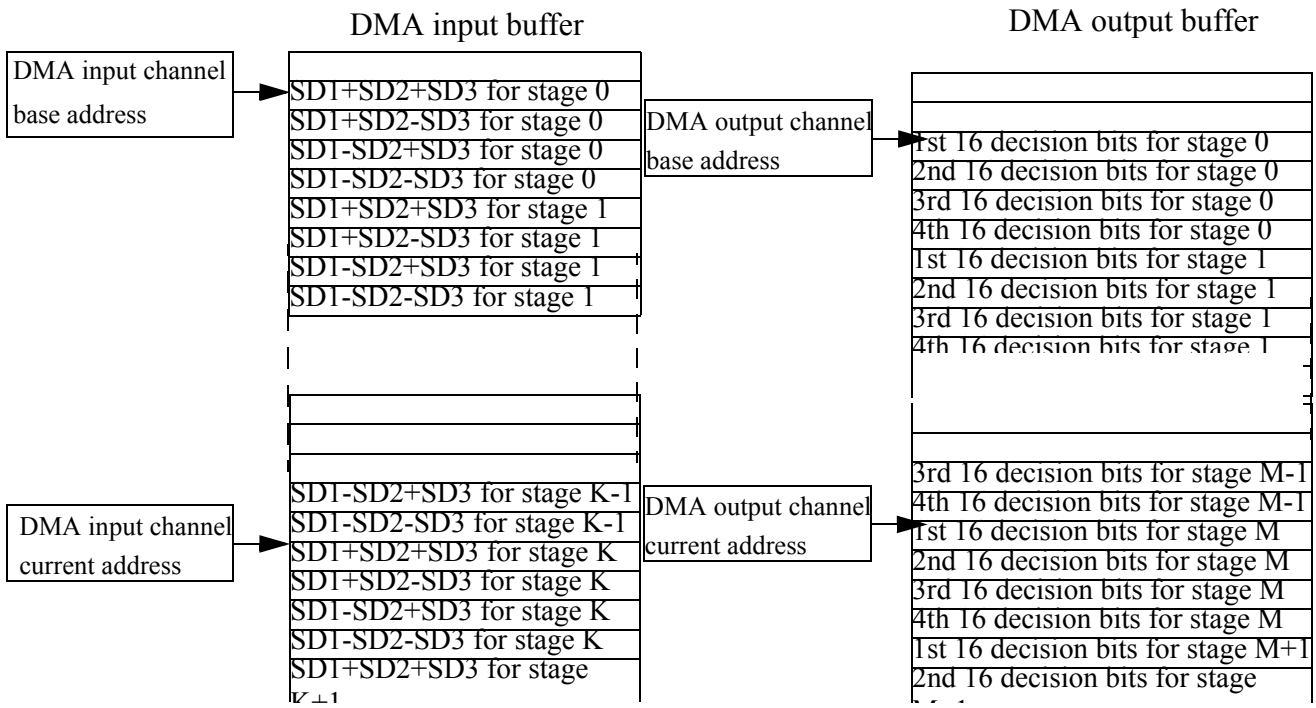


Figure 19-26. DMA buffers organization, in convolutional decoding, code rate 1/3 or 1/6, constraint length of 7

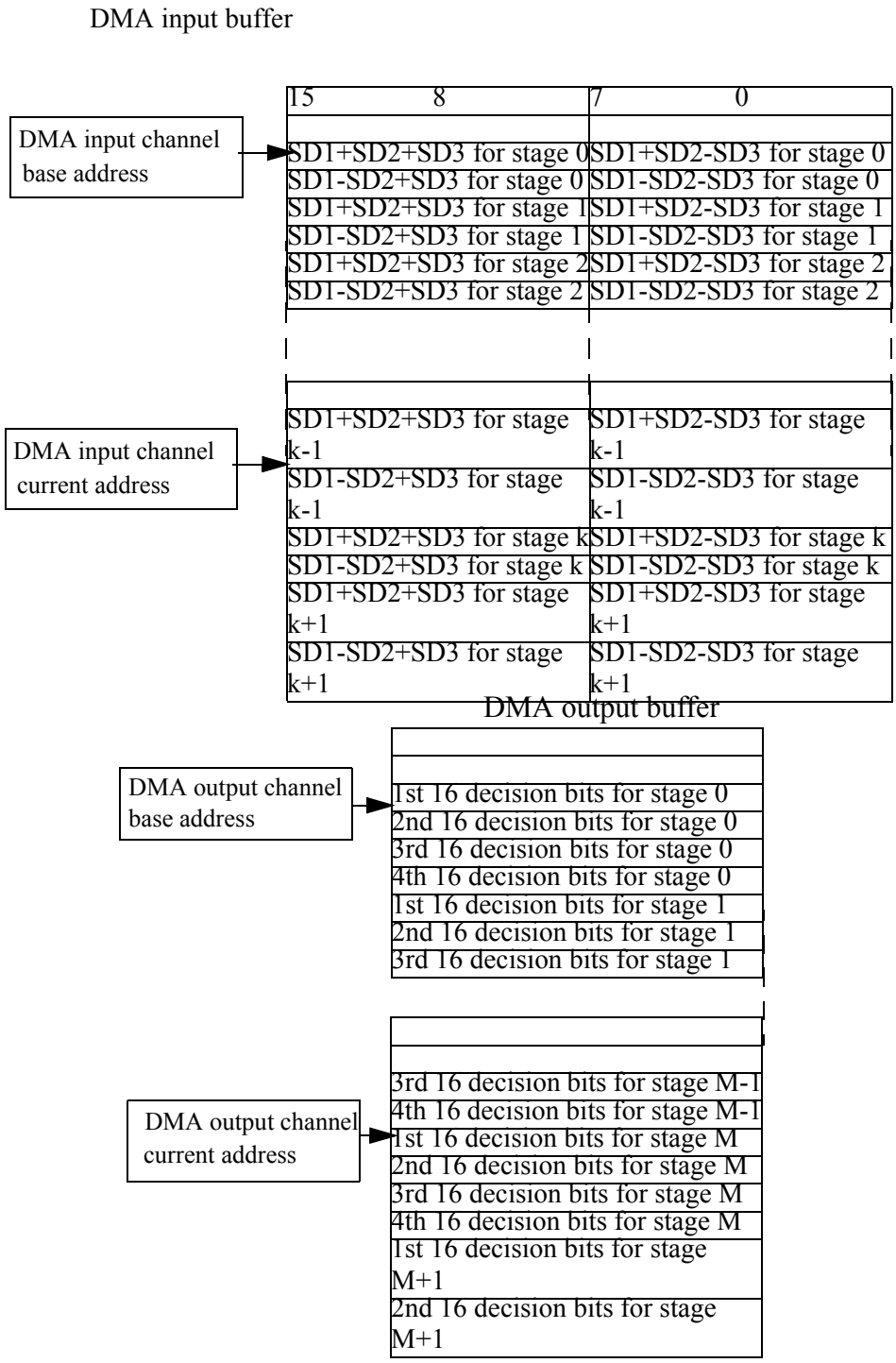


Figure 19-27. DMA buffers organization, in convolutional decoding, code rate 1/2, constraint length of 7, packing of 8 bits

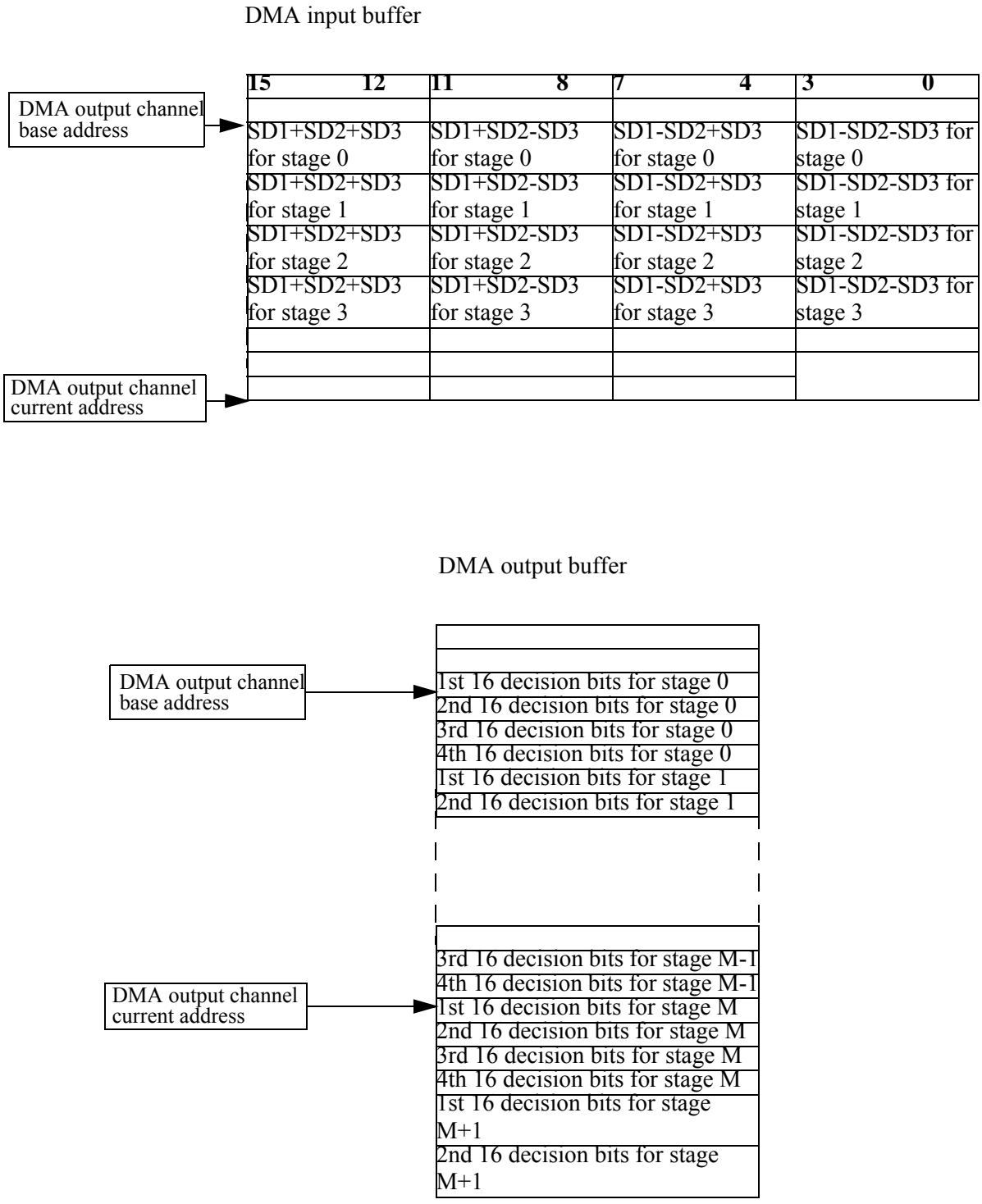


Figure 19-28. DMA buffers organization, in convolutional decoding, code rate 1/2, constraint length of 7, packing of 4 bits

19.6 Programming Model

19.6.1 Viterbi Accelerator (VIAC) Register Summary

The Register Summary lists all registers of the module by ascending address. The absolute address of each register is given, and the value of each bit for reads and writes is given using the conventions in the Legend.

Figure 0-1.

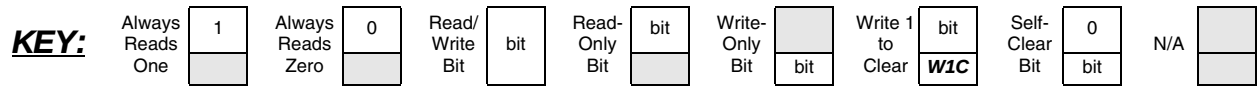


Table 19-2. VIAC Register Summary
Figure 0-2.

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VIDR (X:$FFB0)	R	VIDR[15:0]															
	W																
VBMR (X:$FFB1)	R																
	W	VBMR[15:0]															
VPTR (X:$FFB2)	R	0	TAPG2[4:0]					TAPG1[4:0]					TAPG0[4:0]				
	W																
VODR (X:$FFB3)	R	VODR[15:0]															
	W																
VCSR (X:$FFB4)	R	0				0	0	OERR	IERR	STATE[1:0]		WEDV	WED E	PCF	DOR	DINF	RESE T
	W	CMD[3:0]															
VMR (X:$FFB5)	R	0	0	0	0	0	UP4	UPE	ERRIE	PCIE	DMA	PMI	RATE	0	0	TSN	EDM
	W																
VTCR (X:$FFB6)	R	0	0	0	0	0	TRELCNT[10:0]										
	W																
VWEDR (X:$FFB7)	R	VWEDR[15:0]															
	W																
VWADRR (X:$FFB8)	R	0	0	0	0	0	0	0	0	0	0	VWADRR[5:0]					
	W																
VPMAR A (X:$FFB9)	R	PMFIFO[21:6]															
	W																
VPMAR B (X:$FFBA)	R	PMFIFO[5:0]						0	0	0	0	0	0	0	0	0	0
	W																
VDIBA (X:$FFBB)	R	VDIBA[15:0]															
	W																
VDICA (X:$FFBC)	R	VDICA[15:0]															
	W																
VDOBA (X:$FFBD)	R	VDOBA[15:0]															
	W																
VDOCA (X:$FFBE)	R	VDOCA[15:0]															
	W																

19.6.2 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various VIAC registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

VIDR VIAC Input Data Register **Addr X:$FFB0**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	VIDR[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 19-3. VIDR Description

Name	Description	Settings
VIDR[15:0] Bits 15-0	**VIAC Input Data Register —**The VIDR is a 16-bit read/write register used for receiving an input parameter (a matched filter output) in the equalization mode. Any attempt to access this register while in independent mode is erroneous	N/A

VBMR VIAC Branch Metric Register/FIFO **Addr X:$FFB1**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	VBMR[15:0]															
TYPE	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
RESET	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 19-4. VBMR Description

Name	Description	Settings
VBMR[15:0] Bits 15-0	**VIAC branch metric register/FIFO —**The VBMR is a variable depth FIFO: 8 registers fifo in equalization mode In Convolutional decoding mode: 2 registers fifo for Convolutional decoding mode in 1/2 code rate. 4 registers fifo for Convolutional decoding mode in 1/3 or 1/6 code rate. In Equalization mode: The DSP should write Ungerboeck metrics once per GSM burst processing cycle. In the Decoding mode: The DSP should write Manhattan metrics set every trellis stage processing cycle. A “double-buffering” is implemented to obtain maximum throughput. The VBMR is a Write-only register **Note:** Any attempt to access this register while in independent mode is erroneous	N/A

The VIAC polynomial tap register (VPTR) is a 16-bit read/write register which is programmed with the convolutional code tap polynomials for Convolutional decoding. The VPTR defines 3 polynomials.

— ‘1’ in each tapX[i] defines an existing tap

— ‘0’ in each tapX[i] defines a non-existing tap.

For constraint length of 7 the tap bits do not include the MSB and LSB of the polynomial, which are ‘1’ for all codes.

For constraint length of 5 the polynomial is defined by VPTR as follows:

$$G_j = \sum_{i=0}^{4} TapG_j[i] \times D^i$$

For constraint length of 7 the polynomial is defined by VPTR as follows:

$$G_j = 1 \times D^0 + \sum_{i=1}^{5} TapG_j[i] \times D^i + 1 \times D^6$$

Note:

NOTE:

Note that the GSM05.03 standard defines G1,G2,G3 while here they are called G0,G1,G2 accordingly.

The tap values must be programmed before starting the Convolutional decoding.

The VPTR is a Read/Write register.

VPTR VIAC Polynomial Tap Register **Addr X:$FFB2**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
		TAPG2[4:0]					TAPG1[4:0]					TAPG0[4:0]				
TYPE	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 19-5. VPTR Description

Name	Description	Settings
Bit 15	Reserved Bit	N/A
TAPG2[4:0] Bits 14-10	Tap Vector (polynomial) G2 (TAPG2) — The tap vector C (TAPG2[4-0]) bits contains the vector 2 taps (For constraint length of 7, excluding its MSB and LSB taps).	N/A
TAPG1[4:0] Bits 9-5	**Tap Vector (polynomial) G1 (TAPG1)—**The tap vector B (TAPG1[4-0]) bits contains the vector 1 taps (For constraint length of 7, excluding its MSB and LSB taps).	N/A
TAPG0[4:0] Bits 4-0	**Tap Vector (polynomial) G0 (TAPG0)—**The tap vector A (TAPG0[4-0]) bits contains the vector 0 taps (For constraint length of 7, excluding its MSB and LSB taps).	N/A

VODR VIAC Output Data Register **Addr X:$FFB3**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	VODR[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 19-6. VODR Description

Name	Description	Settings
VODR Bits 15-0	**VIAC Output Data Register—**The VODR is a 16-bit double-buffered register used to output 16 packed decision bits. The VODR is a read-only register.	N/A

The VCSR register is a 16-bit register used by the DSP both to control the VIAC and to examine the VIAC status. VCSR[9-0] bits are all status bits. VCSR[15-12] are used for command

VCSR VIAC Command and Status Register **Addr X:$FFB4**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	CMD[3:0]						OERR	IERR	STATE[1:0]		WEDV	WEDE	PCF	DOR	DINF	RESET
TYPE	slfclr	slfclr	slfclr	slfclr	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 19-7. VCSR Description

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>CMD[3:0]
Bits 15-12</td><td>CMD[3:0] Bits—VIAC command field
Reading these bits always delivers a’0’.</td><td>Following are the VIAC commands:
<table>
<tr><th>CMD [3:0]</th><th>Function</th><th>Description</th></tr>
<tr><td>0000</td><td>reserved</td><td>N/A</td></tr>
<tr><td>0001</td><td>RESET VIAC</td><td>Reset VIAC internal logic, VIAC enters STOP state</td></tr>
<tr><td>0010</td><td>STOP VIAC</td><td>VIAC enters STOP state</td></tr>
<tr><td>0011</td><td>EXIT STOP</td><td>if in STOP state, VIAC enters WAIT state</td></tr>
<tr><td>0100</td><td>START trellis procedure</td><td>Trigger start of trellis procedure</td></tr>
<tr><td>0101</td><td>WED ENABLE</td><td>Enable Window Error Detection function</td></tr>
<tr><td>0110</td><td>WED DISABLE</td><td>Disable Window Error Detection function</td></tr>
<tr><td>0111</td><td>START DMA</td><td>Start DMA, VIAC enters ACTIVE state</td></tr>
<tr><td>1000</td><td>STOP DMA</td><td>Stop DMA, VIAC enters WAIT state</td></tr>
<tr><td>Others</td><td>reserved</td><td>N/A</td></tr>
</table></td></tr>
<tr><td>Bits 11-10</td><td>Reserved Bits</td><td>N/A</td></tr>
</table>

Table 19-7. VCSR Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>OERR
Bit 9</td><td>DMA output Error (OERR) Status Bit—When the DMA output channel performs a write to an erroneous address this bit is set.
If the ERRIE bit in the VMR is set, setting the OERR bit generates an ERR interrupt.
This Bit is a Read-Only bit. Any attempt to write to this bit is ignored.</td><td>0 = no error on DMA output channel accesses
1 = error on DMA output channel accesses</td></tr>
<tr><td>IERR
Bit 8</td><td>DMA Input Error (IERR) Status Bit—When the DMA input channel performs a read from an erroneous address this bit is set.
If the ERRIE bit in the VMR is set, setting the IERR bit generates an ERR interrupt.
Reading the VCSR, clears this bit.
This Bit is a Read-Only bit. Any attempt to write to this bit is ignored.</td><td>0 = no error on DMA input channel accesses
1 = error on DMA input channel accesses</td></tr>
<tr><td>STATE[1:0]
Bits 7-6</td><td>State Status Bits—The VIAC can be either in one of 3 states: STOP, WAIT or ACTIVE. The VIAC states are described in Section 19.7.2, “VIAC states.” .
These Bits are Read-Only bits. Any attempt to write to these bits is ignored.</td><td><table><tr><th>STATE[1:0]</th><th>state</th></tr><tr><td>00</td><td>STOP</td></tr><tr><td>01</td><td>WAIT</td></tr><tr><td>10</td><td>ACTIVE</td></tr></table></td></tr>
<tr><td>WEDV
Bit 5</td><td>Window Error Detection Valid (WEDV) Status Bit—Writing the VWADRR register clears the WEDV bit.(refer to Table 19-11, VWADRR Description).
The WEDV bit is set when the VWEDR register contains valid data.
Note: WED RAM changes while in ACTIVE when WEDE is set.
It is permissible to read the WED only in WAIT.
Note: The reading from WED is considered an atomic operation. After writing the VWADRR register, it is prohibited to perform any access other than read the VCSR or read the VWEDR until the VWEDR is read. Reading of VWEDR terminates this atomic operation.
This Bit is a Read-Only bit. Any attempt to write to this bit is ignored.</td><td>0 = WED value is not valid for reading
1 = WED value is valid for reading in VWEDR register</td></tr>
</table>

Table 19-7. VCSR Description (Continued)

Name	Description	Settings
WEDE Bit 4	**Window Error Detection Enable (WEDE) Status Bit**—Upon execution of "WED enable" command the VIAC sets the WEDE bit and performs window error detection function on the rest of the trellis. **Note:** Whenever the "WED enable" command is executed the WED Ram values are initialized to FFFF. The initialization to FFFF occurs during the first stage. Upon execution of "WED disable" WEDE bit is cleared. This Bit is a Read-Only bit. Any attempt to write to this bit is ignored.	0 = WED function is disabled 1 = WED function is enabled
PCF Bit 3	**Processing Completion Flag (PCF) Status Bit**—The PCF bit is set by the VIAC when the VTCR register decrements to zero (refer to Table 19-9, VTCR Description) and treliss procedure is completed. Writing to the VTCR register clears the PCF bit. **Note:** If PCIE bit in VMR register is set, the VIAC will also generate a Processing Completion interrupt. To clear the PCF interrupt, zero should be written into VTCR register. This Bit is a Read-Only bit. Any attempt to write to this bit is ignored.	0 = Trellis procedure is not completed 1 = Trellis procedure is completed
DOR Bit 2	**Data Output Ready (DOR) Status Bit**—In lockstep mode, the DOR bit is used by the VIAC to inform the DSP that the Data Output buffer is ready for reading and contains the last 16 decision bits. The DOR bit is set whenever the VIAC has new 16 decision bits. The DOR bit is cleared whenever the DSP reads the VODR register. **Note:** If the Data Output buffer is not read in the time frame of the next trellis loop, the VIAC stalls and enters the WAIT state until the data is read. This Bit is a Read-Only bit. Any attempt to write to this bit is ignored.	0 = Data Output is not ready 1 = Data Output is Ready

Table 19-7. VCSR Description (Continued)

Name	Description	Settings
DINF Bit 1	**Data Input Not Full (DINF) Status Bit—**In lockstep mode, the DINF bit is used by the VIAC to request data input (the matched filter output in the equalization mode or the branch metrics in convolutional decoding mode). In equalization the DINF reflects the status of the VIDR. In convolutional decoding the DINF reflects the status of the VBMR fifo. The bit is set when Data Input buffer in not full and the VIAC needs to be fed with new data within the next trellis loop period. Refer to Section 19.5.4, “VIAC’s pipeline.” . **Note:** Note that if the Data Input buffer is not written in the time frame of the next trellis loop, the VIAC stalls and enters the WAIT state until the data is written. **Note:** Note that writing new parameters to the input buffers, when DINF is cleared, is erroneous. This Bit is a Read-Only bit. Any attempt to write to this bit is ignored.	0 = Data Input buffer is full 1 = Data Input buffer is not full
RESET Bit 0	**Reset Status Bit—**Executing RESET command sets RESET bit.The VIAC clears this bit when the reset process is completed. Upon completion of reset process, the VIAC enters the STOP state. This Bit is a Read-Only bit. Any attempt to write to this bit is ignored.	0 = VIAC is not in reset 1 = VIAC is in reset

The VMR register is a 16-bit read/write register used to program the VIAC's operating mode.

VMR VIAC Mode Register **Addr X:$FFB5**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
						UP4	UPE	ERRIE	PCIE	DMA	PMI	RATE			TSN	EDM
TYPE	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 19-8. VMR Description

Name	Description	Settings
Bits 15-11	Reserved Bits	N/A
UP4 Bit 10	**Unpacking 4 bits mode (UP4) bit—** The UP4 bit defines if unpacking operation is done in 4 bits mode or in 8 bits mode as described in Section 19.5.7, "DMA buffers organization." . In 8 bits mode the VIAC unpacks each of the two 8 bit bytes into two16 bits words, performing sign extension on the most significant bits. In 4 bits mode the VIAC unpacks each of the four 4 bit nibbles into four 16 bits words, performing sign extension on the most significant bits.	0 = Unpacking is done in 8 bits mode 1 = Unpacking is done in 4 bits mode
UPE Bit 9	Unpacking Enable (UPE) bit— The UPE bit defines if upon convolutional decoding, in DMA mode, the VIAC performs unpacking operation on the input parameters coming from the DMA input buffer. The Unpacking operation is done in 2 modes: 8 bit unpacking mode and 4 bit unpacking mode (according to UP4 bit) as described in Section 19.5.7, "DMA buffers organization." .	0 = Unpacking is disabled 1 = Unpacking is enabled
ERRIE Bit 8	**DMA Error Interrupt Enable (ERRIE) Bit—** In independent mode, the ERRIE bit defines if when one of the DMA channels accesses an erroneous address, the VIAC will generate an ERR interrupt. This bit is ignored in lockstep mode.	0 = DMA Error Interrupt disable 1 = DMA Error Interrupt enable
PCIE Bit 7	**Processing Completion Interrupt Enable (PCIE) Bit —** The PCIE bit defines if upon the end of the trellis procedure the VIAC will generate a Processing Completion Interrupt.	0 = Processing Completion Interrupt disable 1 = Processing Completion Interrupt enable

Table 19-8. VMR Description (Continued)

Name	Description	Settings
DMA Bit 6	**Independent Mode (DMA) Bit—** The DMA bit defines whether the VIAC operates in lockstep with the DSP or independent to the DSP. In lockstep mode the DSP accesses the VIAC every loop, while in independent mode the VIAC accesses the DPRAM using its 2 DMA channels. **Note:** Note that when DMA bit is cleared, START DMA and STOP DMA commands are ignored.	0 = VIAC in lockstep mode 1 = VIAC in independent mode
PMI Bit 5	**Path Metric Initialization (PMI) Bit—** The PMI bit defines the form of path metric initialization. Writing to the FIFO, the VIAC must be in the WAIT state (refer to Section 19.7.2, “VIAC states,”) and the PMI bit in VMR register must be set, before executing the START command. Any attempt to write to the fifo without meeting the conditions in the settings column is erroneous. The Path Metric RAM initialization is described in Section 19.5.5, “Path Metric RAM access using VPMAR FIFO.” .	0 = All Path Metrics are cleared. 1 = Path Metric RAM is initialized by the DSP CORE
RATE Bit 4	**Code Rate (RATE) Bit—**The RATE bit defines the convolutional code rate for Convolutional decoding. This bit is ignored in the equalization mode.	0 = Code Rate is 1/2 1 = Code Rate is 1/3 or 1/6
Bits 3-2	Reserved	N/A
TSN Bit 1	**Trellis States Number (TSN) Bit—** Defines the number of states in the convolutional decoding trellis.This bit is ignored in the equalization mode.	0 = Constraint Length is 5, Number of trellis states is 16 1 = Constraint Length is 7, Number of trellis states is 64
EDM Bit 0	Equalization/Decoding Mode (EDM) Bit— Defines whether the VIAC operates in channel equalization mode, or convolutional decoding mode.	0 = channel equalization mode 1 = convolutional decoding mode

VTCR VIAC Trellis Counter Register **Addr X:$FFB6**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
						TRELCNT[10:0]										
TYPE	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 19-9. VTCR Description

Name	Description	Settings
TRELCNT[10:0] Bits 10-0	**VIAC Trellis Counter Register —**The VTCR is a 16-bit read/write register used to set the number of trellis stages to be executed. During trellis processing, the contents of the VTCR shows the number of remaining trellis stages. One may extend the processing by repeatedly updating the VTCR. Updating VTCR is permissible only when the VIAC is in WAIT state (caused by lack of input parameters or before START command).	N/A

VWEDR VIAC Window Error Detection Data Register **Addr X:$FFB7**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	VWEDR[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 19-10. VWEDR Description

Name	Description	Settings
VWEDR[15:0] Bits 15-0	**VIAC Window Error Detection Data Register—**The VWEDR is a 16-bit read/write register used for reading the Window Error Detection value of a specified trellis state. The path is specified by writing the trellis state number, whose WED we want to read, to the VWADRR register. The WED RAM changes while in ACTIVE when WEDE is set. It is permissible to read the WED only in WAIT. The VWEDR is valid for reading only when the WEDV bit in VCSR is set. Reading or writing the VWEDR clears the WEDV bit.	N/A

VWADRR VIAC Window Error Detection Address Register **Addr X:$FFB8**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
											VWADRR[5:0]					
TYPE	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?

Table 19-11. VWADRR Description

Name	Description	Settings
VWADRR[5:0] Bits 5-0	**VIAC Window Error Detection Address Register (VWADRR)—**The VWADRR is a 16-bit read/write register used to specify the trellis state whose Window Error Detection value is to be read from VWEDR.	N/A

The VPMAR partitioned into two read/write registers: VPMARA and VPMARB. In the path metric initialization mode, the VPMAR is used as a FIFO memory buffer to initialize the path metrics.

VPMARA VIAC Path Metric Access FIFO A Register **Addr X:$FFB9**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PMFIFO[21:6]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 19-12. VPMARA Description

Name	Description	Settings
PMFIFO[15:0] Bits 15-0	**VIAC Path Metric Access FIFO A Register—** VPMARA includes 16 most significant bits of the 22-bit path metric fifo.	N/A

VPMARB VIAC Path Metric Access FIFO B Register **Addr X:$FFBA**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PMFIFO[5:0]															
TYPE	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r	r	r
RESET	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0

Table 19-13. VPMARB Description

Name	Description	Settings
PMFIFO[5:0] Bits 15-0	**VIAC Path Metric Access FIFO B Register—** VPMARB includes 6 least significant bits of the 22-bit path metric fifo VPMAR is a variable depth 22-bit FIFO. The FIFO depth is equal to the number of trellis states (either 16 or 64). Refer to Figure 19-16., "VPMAR FIFO".	N/A

VDIBA VIAC DMA Input Channel Base Address Register **Addr X:$FFBB**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	VDIBA[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 19-14. VDIBA Description

Name	Description	Settings
VDIBA[15:0] Bits 15-0	V**IAC DMA Input Channel Base Address —**The VDIBA is the base address register of the DMA input channel. In independent mode this register includes the base address of the input DMA buffer, in the DPRAM. In ACTIVE state, in independent mode the DMA accesses the DPRAM every trellis loop, to supply sufficient input parameters. The VDIBA may be written, only while VIAC is in WAIT state. Any attempt to write this register, while in ACTIVE will be ignored. **Note:** A change in VDIBA influences the VIAC only on START_DMA command. The VDIBA is a read/write register.	N/A

VDICA VIAC DMA Input Channel Current Address Register **Addr X:$FFBC**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	VDICA[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 19-15. VDICA Description

Name	Description	Settings
VDICA[15:0] Bits 15-0	V**IAC DMA Input Channel Current Address**—The VDICA is the current address register of the DMA input channel. In independent mode this register includes the address of the current word which is being accessed by the DMA input channel. In ACTIVE state, in independent mode the DMA accesses the DPRAM every trellis loop, to supply sufficient input parameters. Whenever the DMA input channel accesses an erroneous address, the VIAC, sets the IERR bit in the VCSR. **Note:** If the ERRIE bit in the VMR is set, the VIAC will also generate an ERR interrupt. The VDICA is a read-only register.	N/A

19.6.3 VIAC DMA Output Channel Base Address (VDOBA)

VDOBA — VIAC DMA Output Channel Base Address Register — **Addr X:$FFBD**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	VDOBA[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 19-16. VDOBA Description

Name	Description	Settings
VDOBA[15:0] Bits 15-0	V**IAC DMA Output Channel Base Address**—The VDOBA is the base address register of the DMA output channel. In independent mode this register includes the base address of the output DMA buffer, in the DPRAM. In ACTIVE state, in independent mode the DMA accesses the DPRAM every trellis loop, to save the decisions results. In ACTIVE state, in independent mode, the DMA output channel saves the 16 decision bits every 16 DSP clocks. The VDOBA may be written, only while VIAC is in WAIT state. Any attempt to write this register, while in ACTIVE will be ignored. **Note:** That change in VDOBA influences the VIAC only on START_DMA command. The VDOBA is a read/write register.	N/A

19.6.4 VIAC DMA Output Channel Current Address (VDOCA)

VDOCA VIAC DMA Output Channel Current Address Register **Addr X:$FFBE**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	VDOCA[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 19-17. VDOCA Description

Name	Description	Settings
VDOCA[15:0] Bits 15-0	V**IAC DMA Output Channel Current Address**—The VDOCA is the current address register of the DMA output channel. In independent mode this register includes the address of the current word which is being accessed by the DMA input channel. In ACTIVE state, in independent mode the DMA accesses the DPRAM every trellis loop, to save the decisions results. Whenever the DMA output channel accesses an erroneous address the VIAC sets the OERR bit in the VCSR. Note that if the ERRIE bit in the VMR is set, the VIAC will also generate an ERR interrupt. The VDOCA is a read-only register.	N/A

19.7 Operating Modes

19.7.1 General description

The VIAC can operate in two main modes: MLSE for channel equalization mode or convolutional decoding mode.

19.7.2 VIAC states

The VIAC can be either in one of 3 states: STOP, WAIT or ACTIVE.

- When the VIAC is in STOP its internal clocks are negated and all internal logic is disabled. The only accessible register is the VCSR.
- When the VIAC is in WAIT state it does not perform the trellis loop since its input buffer has no sufficient data or its output buffer is full with unread data.When the VIAC is in ACTIVE state it performs the trellis main loop. Note that during the ACTIVE state, it is prohibited to write the VTCR, VPTR
  and VMR registers.

Note that the transitions between the states always occur upon termination of a single trellis loop although the commands are executed asynchronously in the middle of a trellis loop processing.

Figure 19-29 illustrates the possible VIAC state transitions.

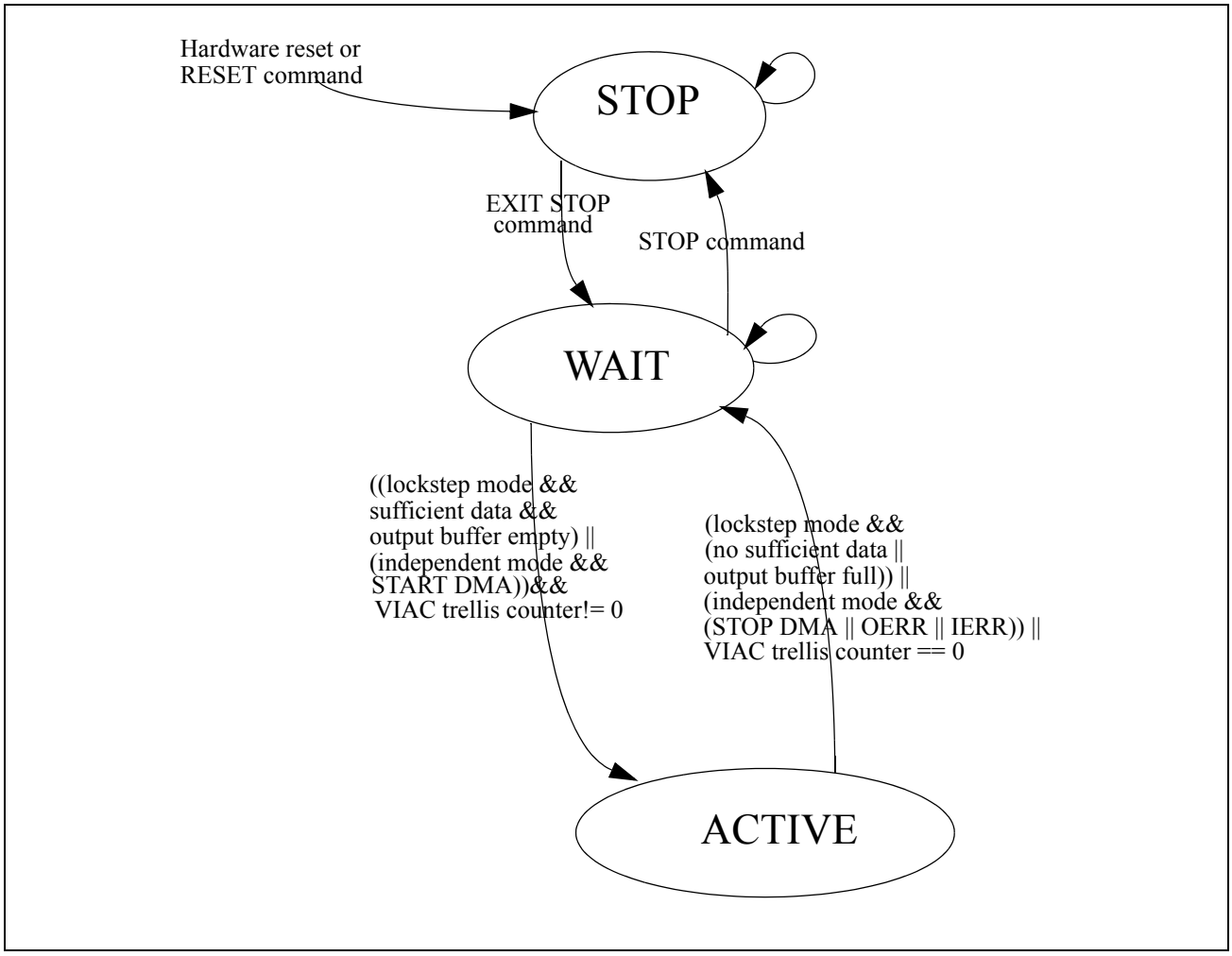


Figure 19-29. VIAC states

Upon RESET the VIAC enters STOP state. Execution of EXIT STOP command causes the VIAC to enter WAIT state. The DSP programs the VIAC with the needed Viterbi algorithm characteristics.

In lockstep mode, upon starting the trellis procedure and after sufficient data is loaded into the input buffer the VIAC enters the ACTIVE state where the main trellis loop is being processed. In ACTIVE the DSP is required to feed the VIAC with sufficient input parameters and to read the results. When the DSP does not meet this requirement the VIAC returns to WAIT state until the DSP feeds the input buffers or reads the output buffer.

In independent mode, upon starting the trellis procedure and executing the START DMA command the VIAC enters the ACTIVE state where the main trellis loop is being processed. In ACTIVE the DMA input channel reads the DPRAM and feeds the VIAC with input parameters and the DMA output channel saves the results in the DPRAM. Upon the end of the trellis, or upon executing STOP DMA command the VIAC enters the WAIT state until restarted again.

Upon STOP command, while in WAIT state, the VIAC enters the STOP state and disables all internal logic.

The following figures illustrates an example for the VIAC normal operation.

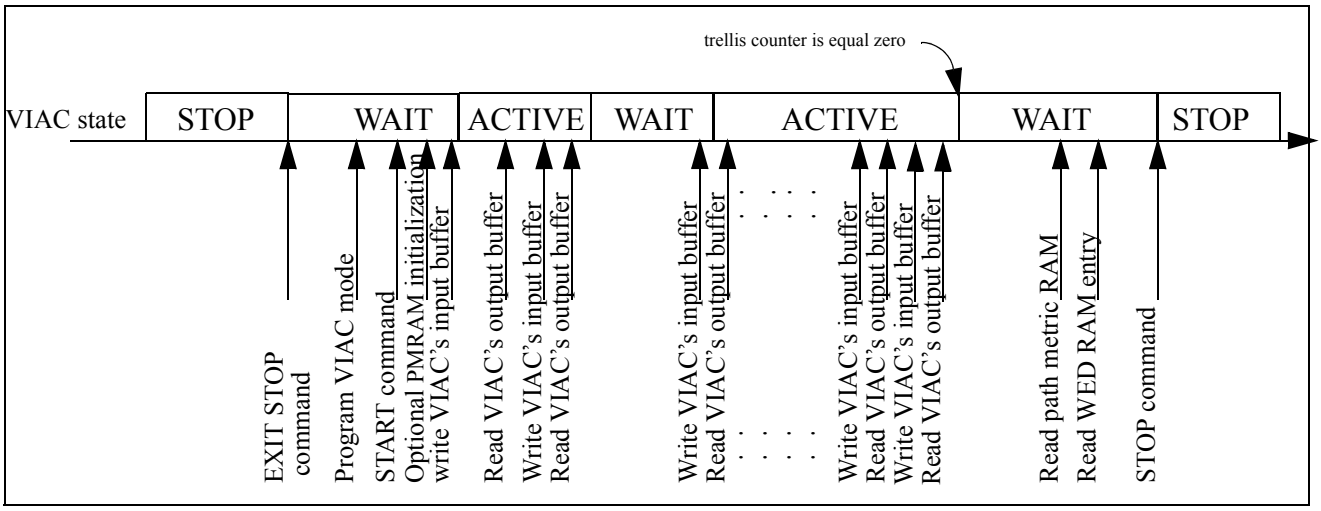


Figure 19-30. Example of VIAC normal operation in lockstep mode

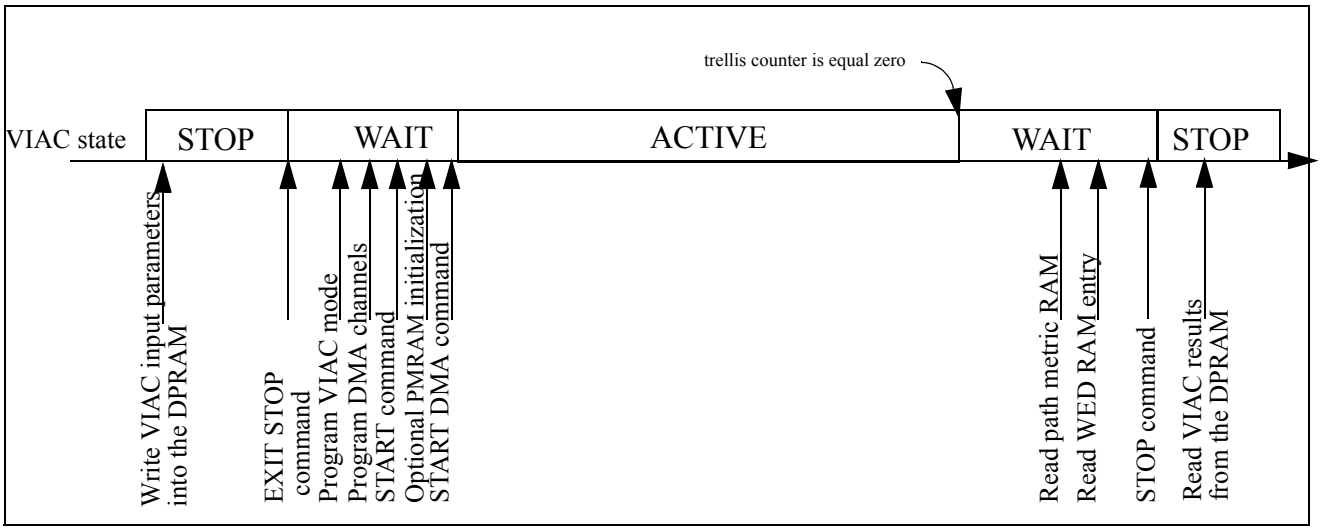


Figure 19-31. Example of VIAC normal operation in independent mode

A trellis processing procedure includes the following stages:

19.7.3 Trellis processing in equalization

19.7.3.1 Preparing the VIAC for trellis main loop

Assuming the VIAC is in WAIT state. Preparing the VIAC for the trellis main loop includes the following steps (Refer to Figure 19-32),

1. Initialize Branch Metric RAM:
   - initialize the Branch Metric RAM with Ungerboeck metrics by writing to VBMR fifo.
   - Note that because of branch metrics anti-symmetry, only half of the coefficients set need to be loaded into the VBMR fifo.
2. VTCR update:
   - Write the length of the trellis (the number of trellis stages to be executed) into VTCR.
3. Select VIAC operating mode - VMR programing:
   - Select which path metric initialization is to be performed by writing the PMI bit.
   - The actual initialization process is to be applied after executing the START command.
   - Selecting “Equalization” by clearing the EDM bit.
4. In independent mode only: Prepare DMA channels
   - It is assumed that the DMA input buffer, located in the DPRAM contains valid input parameters needed for the trellis procedure. In equalization, the input parameters needed are all the method filter’s output parameters.
   - Program the two DMA base address registers, of the DMA input channel and of the DMA output channel.
5. Trigger the VIAC to start initialization phase
   - The VIAC is triggered to start the initialization phase by executing the START command
6. Path metric initialization phase
   - If PMI bit in VMR is set, the DSP needs to initiate the PMRAM by writing VPMARA,VPMARB fifo registers (Refer to Section 19.5.5).
   - If PMI bit in VMR is clear the VIAC continues directly to the next phase, trellis processing, with all path metrics cleared

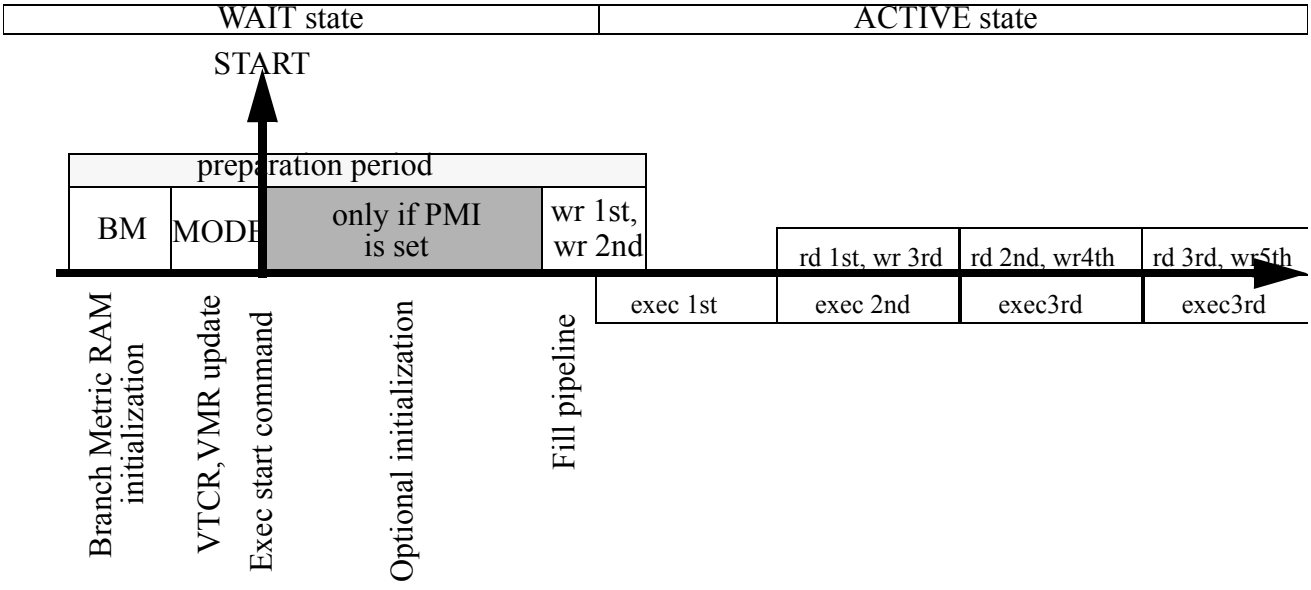


Figure 19-32. Preparing the VIAC for trellis main loop, in lockstep mode

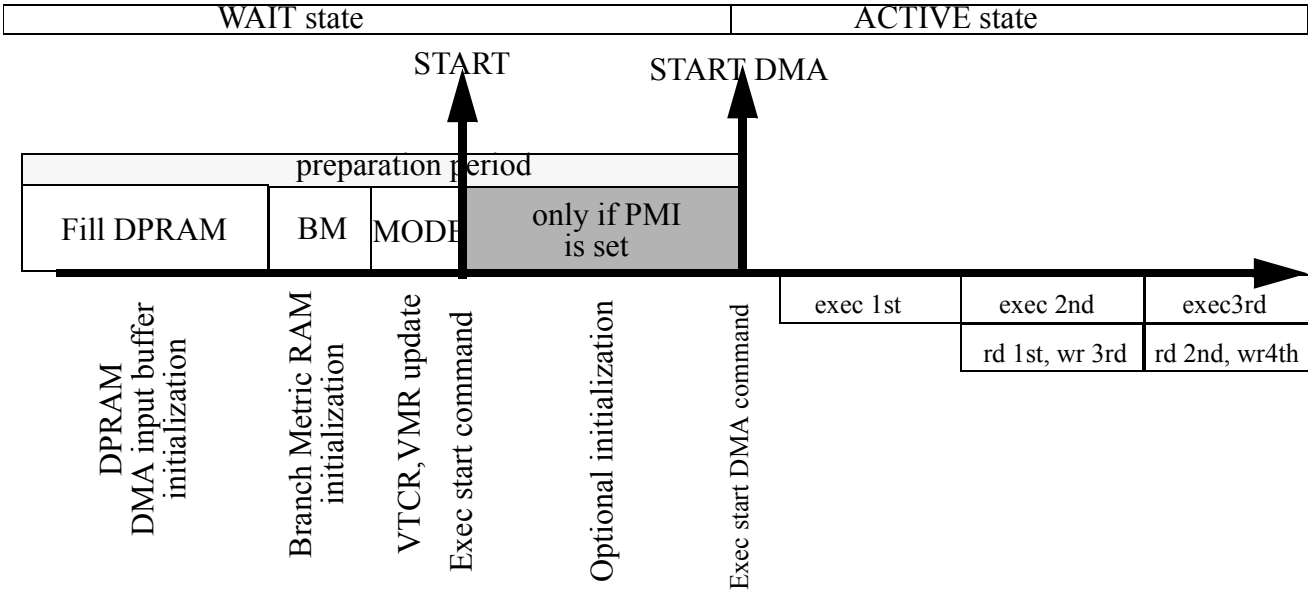


Figure 19-33. Preparing the VIAC for trellis main loop, in equalization, in independent mode

19.7.3.2 trellis main loop

1. In lockstep mode:
   - During trellis processing phase, the DSP feeds the VIAC with a set of data: a 16-bit matched filter output sample is fed into VIDR each trellis stage. Refer to Section 19.5.4.
   - For each trellis stage ACS calculations are executed for all trellis states. In each ACS

calculation a single decision bit is delivered from the ACS block to VODR.

— On completion of processing of the 16 trellis states, the 16 decision bits are put in the VODR and have to be read by the DSP.

— Double-buffering of the VIDR and the VODR allows writing new data during the data processing in the VIAC.

2. In independent mode:

— During trellis processing phase the DSP is not involved with the VIAC operation. Any DSP access to the same 1/4K DPRAM, which is currently accessed by the DMA channels will stall the DMA and therefore may stall the VIAC.

19.7.3.3 Trellis Completion

19.7.3.3.1 Notifying the DSP

- After completing the programmed number of trellis stages the VIAC sets the PCF bit in VCSR. If the PCIE bit in the VMR is set, the VIAC will also generate a Processing Completion interrupt.

19.7.3.3.2 Reading Path Metric RAM

- Whenever the VIAC is in WAIT state the DSP may read the Path Metric RAM (Refer to Section 19.5.5).
- Note that the VIAC enters the WAIT state when VTCR equals zero or when the DSP does not meet the input/output requirements. (Refer to Section 19.7.2).

19.7.4 Trellis processing in convolutional decoding

19.7.4.1 Preparing the VIAC for trellis main loop

The VIAC is in WAIT state. Preparing the VIAC for the trellis main loop includes the following steps (Refer to Figure 19-34).

1. Select VIAC operating mode - VMR programing:

— Select which path metric initialization is to be performed by writing the PMI bit.

— The actual initialization process is to be applied after executing the START command.

- Selecting Code Rate by writing the RATE bit.
- Selecting Trellis states number by writing the TSN bit.
- Selecting “Convolutional decoding” by setting the EDM bit.

2. VTCR update:

— Write the length of the trellis (the number of trellis stages to be executed) into VTCR.

3. VPTR update:

— Write the 3 polynomial taps into VPTR.

4. In independent mode only: Prepare DMA channels

— It is assumed that the DMA input buffer, located in the DPRAM contains valid input parameters needed for the trellis procedure. In convolutional decoding, the

input parameters needed are all the Manhattan metrics parameters.

— Program the two DMA base address registers, of the DMA input channel and of the DMA output channel.

5. Trigger the VIAC to start initialization phase

— The VIAC is triggered to start the initialization phase by executing the START command

6. Path metric initialization phase

— If PMI bit in VMR is set, the DSP needs to initiate the PMRAM by writing VPMARA,VPMARB fifo registers (Refer to Section 19.5.5).

— If PMI bit in VMR is clear the VIAC continues directly to the next phase, trellis processing, with all path metrics cleared

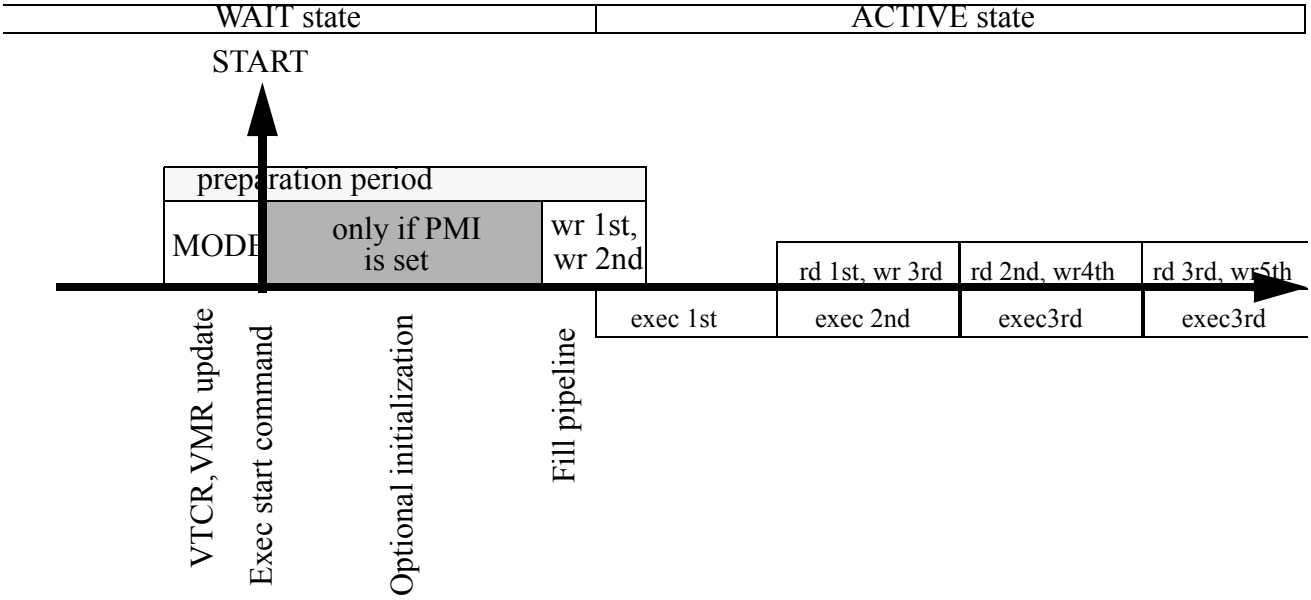


Figure 19-34. Preparing the VIAC for trellis main loop, in convolutional decoding, in lockstep mode

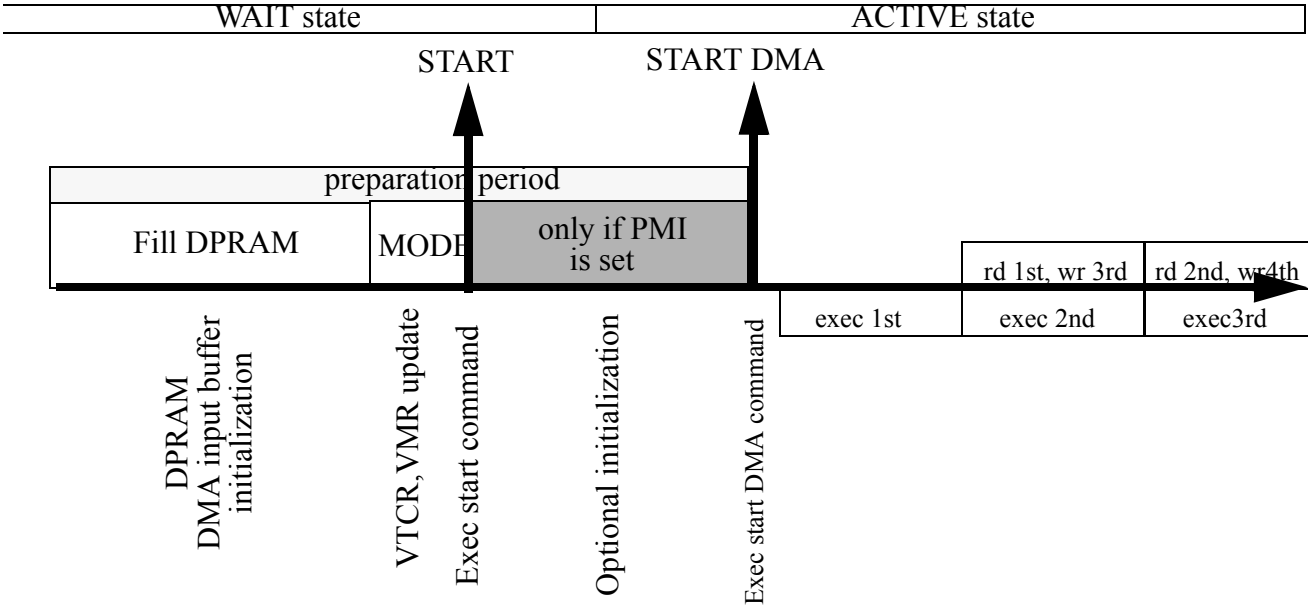


Figure 19-35. Preparing the VIAC for trellis main loop, in convolutional decoding, in independent mode

19.7.4.2 trellis main loop

1. in lockstep mode:
   - During trellis processing phase, the DSP feeds the VIAC with a set of data and continues its processing in preparation for the next set:
     - In Convolutional decoding: 2 (for 1/2 code rate) or 4 (for 1/3 or 1/6 code rate) Manhatten metrics are written into VBMR fifo:
     - for the 1/2 code rate:

$$\begin{bmatrix} C(k)+C(k-1) \\ C(k)-C(k-1) \end{bmatrix}$$

   - for the 1/3 code rate:
   - for the 1/6 code rate:

$$\begin{bmatrix} C(k)+C(k+1)+C(k+2)+C(k+3)+C(k+4)+C(k+5) \\ C(k)+C(k+1)-C(k+2)+C(k+3)+C(k+4)-C(k+5) \\ C(k)-C(k+1)+C(k+2)+C(k+3)-C(k+4)+C(k+5) \\ C(k)-C(k+1)-C(k+2)+C(k+3)-C(k+4)-C(k+5) \end{bmatrix}$$

$$\begin{bmatrix} C(k)+C(k+1)+C(k+2) \\ C(k)+C(k+1)-C(k+2) \\ C(k)-C(k+1)+C(k+2) \\ C(k)-C(k+1)-C(k+2) \end{bmatrix}$$

— For each trellis stage ACS calculations are executed for all trellis states. In each ACS calculation a single decision bit is delivered from the ACS block to VODR.

— On the completion of processing 16 trellis states, the 16 decision bits are put in the VODR and has to be read by the DSP.

— Double-buffering of the VIDR and the VODR allows writing new data during the data processing in the VIAC.

2. in independent mode:

— During trellis processing phase the DSP is not involved with the VIAC operation. Any DSP access to the same 1/4K DPRAM, which is currently accessed by the DMA channels will stall the DMA and therefore may stall the VIAC.

19.7.4.3 Trellis Completion

19.7.4.3.1 Notifying the DSP

- After completing the programmed number of trellis stages the VIAC sets the PCF bit in VCSR. If the PCIE bit in the VMR is set, the VIAC will also generate a Processing Completion interrupt.

19.7.4.3.2 Reading Path Metric RAM

- Whenever the VIAC is in WAIT state the DSP may read the Path Metric RAM
- Reading includes Reading all 16 or 64 Path Metrics by reading VPMARA,VPMARB fifo registers.
- Note that the VIAC enters the WAIT state when VTCR equals zero or when the DSP does not meet the input/output requirements. (Refer to Section 19.7.2).

19.7.4.3.3 Reading WED

- After selecting a path the DSP may read the WED result which relates to this specified path
- The sequence of reading the WED value of a specific path includes 3 steps:
  — Write the number of state which the path start from to the VWADRR register.
  — Wait for WEDV bit in the VCSR register to be set.
  — Read the VWEDR register which contains the WED value.

19.7.5 VIAC's interrupts

The VIAC generates 2 interrupts:

1. Processing completion interrupt
   — If the PCIE bit in the VMR is set, the VIAC generates a Processing Completion interrupt, upon the completion of the trellis procedure.
2. ERR interrupt
   — In independent mode, if the ERRIE bit in the VMR is set, the VIAC generates an ERR interrupt whenever the DMA channel access an erroneous address.

References:

1. Adaptive Maximum Likelihood Receiver for Carrier Modulated Data Transmission Systems, Gottfied Ungerboeck, IEEE Trans. on Comm. May 1974.
2. Soft decision information from an MLSE Equalizer via ISI Cancellation, David Borth and Phil Raskey. (Motorola Internal Documentation)
3. The Viterbi Algorithm by G. David Forney, JR, Proc. IEEE, vol. 61 pp. 268-278, March 73.

Motorola - Noida Development Center

DIFFERENCES BETWEEN

HARD VIAC (From Patriot)

&

VIAC RTL (For Neptune)

Author(s): , Manish Mittal

Table 19-18. Differnces Between Hard VIAC & VIAC RTL

S.No.	Issues	Behaviour Of Hard VIAC	Behaviour Of VIAC RTL	Resolution/Comments
1.	**Path Choosing Problem**: When both the calculated Path Metrics for a particular sate comes out to be the same then the path chosen by Hard Viac and the Viac RTL sometimes differ.	In hard VIAC,it is not consistant in path choosing and applies a particular pattern to implement this.	In VIAC RTL, VIAC is choosing upper path consistantly.	a) Theoretically, we can choose any of the two paths. It was decided that for current situation in order to meet the schedule, verification team will change the differing VODR values in their test cases. b) But in parallel, design team will be working on its implementation for replicating the hard VIAC's implementation so that in the next release the new design will be merged in the current chain.

Table 19-18. Differnces Between Hard VIAC & VIAC RTL

S.No.	Issues	Behaviour Of Hard VIAC	Behaviour Of VIAC RTL	Resolution/Comments
2.	**VIDR & VBMR Overwrite**: These registers are input register for VIAC.	These registers get overwritten in hard VIAC if we give the next input immedaitely before the VIAC processes that input.	In RTL overwritting operation onto these registers is ignored.	In the specs it is written that over-writting these registers is an erroronous operation. Therefore, we will be continuing with the RTL approach.
3.	**START DMA Problem:**After START DMA command, VIAC is asumed to fetch inputs from the DMA input memory starting from the address specified by VDIBA.	In the hard block VIAC fetches the two inputs with three clock cycles difference in between.	After START DMA command, VIAC RTL fetches 2 inputs from the memory simultaneously	As this does not effect the performance of VIAC and nothing is written in specs, VIAC RTL is assumed to be OK.

Table 19-18. Differnces Between Hard VIAC & VIAC RTL

S.No.	Issues	Behaviour Of Hard VIAC	Behaviour Of VIAC RTL	Resolution/Comments
4.	**DINF Bit Problem:** DINF bit is indicator to DSP that VIAC needs input to be proceesed.	This bit get cleared in the hard VIAC even when only one input is there in the the input register.	But in VIAC RTL, it gets cleared only when both the input buffered registers are full.	The input register is double buffered. According to specs,until the DINF bit gets set,we cannot write into the input register. In case of hard VIAC, this bit will be set when both the input registers will be empty and everytime after each trellis if we poll this bit to see whether VIAC nedds next input or not,the VIAC will go to wait mode due to input stall. Therefore, VIAC RTL has been assumed good.

Table 19-18. Differnces Between Hard VIAC & VIAC RTL

S.No.	Issues	Behaviour Of Hard VIAC	Behaviour Of VIAC RTL	Resolution/Comments
5.	**WEDV bit problem:** This bit indicates that the valid WED data has come into the VWEDR register.	a) In the hard VIAC, WEDV bit is cleared only when we write into the VWADDR register. This bit remains set even when we read the VWEDR register. b) In Hard Viac the WEDV bit is set even though there has been no trellis procedure, WED has not been enabled and only the VWADRR has been written into after bringing Viac into WAIT mode from STOP mode.	a) In VIAC RTL this bit gets cleared when we write into the VWADDR register and also when we read or write the VWEDR register. b) In VIAC RTL this bit gets set only if there has already been a trellis with WED enable command.	This looks to be a problem with hard VIAC as compared to the specifications. In the specs it is written that reading or writting the VWEDR register also clears the WEDV bit. In the second case, the behaviour of hard VIAC is wrong as there has not been any trellis. Therefore VIAC RTL is considered OK.

Table 19-18. Differnces Between Hard VIAC & VIAC RTL

S.No.	Issues	Behaviour Of Hard VIAC	Behaviour Of VIAC RTL	Resolution/Comments
6.	**Interrupt Handling:** In independent mode if VIAC tries to access any memory location beyond its range (2K shared memory), it generates an interrupt.	In independent mode, when there is IERR or OERR an interrupt is raised by the viac. In hard VIAC, VIAC continues to read/write from/to the wrong memory address and performs its internel functioning.	In VIAC RTL, after the interrupt, VIAC suspends the read/write function until the memory address is corrected in the VDOBA register.	In case of hard VIAC, the continues to perform operation and generates the errornous results. After servicing the interrupt routine, we have to restart the whole trellis procedure as VIAC has already done some wrong calculations during the interrupt servicing. Therefore, VIAC RTL is considered OK.

Table 19-18. Differnces Between Hard VIAC & VIAC RTL

S.No.	Issues	Behaviour Of Hard VIAC	Behaviour Of VIAC RTL	Resolution/Comments
7.	**Asserting RESET:** There is one additional RESET command in the VIAC. It resets the VIAC internal logic.	Hard VIAC takes two clock cycles to perform reset operation.	VIAC RTL takes four clock cycles to perform the same operation.	Reset bit in VCSR is to be checked to see if reset is asserted or not. Thus it wont cause problems in application.Therefore VIAC RTL is considered acceptable.

Table 19-18. Differnces Between Hard VIAC & VIAC RTL

S.No.	Issues	Behaviour Of Hard VIAC	Behaviour Of VIAC RTL	Resolution/Comments
8.	**WED ENABLE & WED DISABLE COMMAND:** These commands are given if we want the WED values for a particular trellis.	In the first case, we are enabling the WED before trellis start and complete the first trellis. After that due to no input the VIAC goes in the wait mode. In the wait mode we are giving WED DISABLE + WED ENABLE+WED DISABLE + WED ENABLE command one by one and then give the second part of second input to put the VIAC in the active mode. Now what is happening in the hard VIAC ,it ignores all the WED enable and disable commands and continues the next trellis by carrying the previous WED values instead of reinitializing the WED RAM to FFFF.	But in the VIAC RTL, these commands are executed and the WED RAM values are reinitialized to FFFF for the next trellis.	It seems that the hard VIAC ignores the WED ENABLE & WED DISABLE commands while it is in wait mode. But in the specs it is written that when WED ENABLE commands comes, WED RAM is initialized to FFFF. Therefore, VIAC RTL is considered acceptable.

Table 19-18. Differnces Between Hard VIAC & VIAC RTL

S.No.	Issues	Behaviour Of Hard VIAC	Behaviour Of VIAC RTL	Resolution/Comments
9.	**WED ENABLE & WED DISABLE COMMAND:** These commands are given if we want the WED values for a particular trellis.	Now in the second case, The first trellis is executed without WED enabling. For the second trellis we give the one half of the second input to put the VIAC in the wait mode. After that WED ENABLE command is given. Then we give the second part of the second input to again put VIAC in the active mode. In the hard VIAC, the WED is not getting enabled for that trellis but gets enabled in the next trellis.	In the VIAC RTL, the WED gets enabled in that trellis itself before which the WED ENABLE command is given.	According to the specs,after giving the WED ENABLE command, WED should be enabled for the immediate next trellis. Therefore the VIAC RTL considered OK.

Table 19-18. Differnces Between Hard VIAC & VIAC RTL

S.No.	Issues	Behaviour Of Hard VIAC	Behaviour Of VIAC RTL	Resolution/Comments
10.	**PathMetric RAM access using VPMAR FIFO**: When The DSP access the PM RAM , the RAM is organized as a FIFO. The PM RAM can be read / write using the 22 bits VPMAR VPMARA,VPMARB . Both register must be access consecutively starting with VPMARA and then VPMARB to obtain one fifo entry. Each time VPMARB is accessed both VPMARA and VPMARB are either written into the RAM or loaded with data from the RAM.	In Hard VIAC, the path metric RAM can be initialize by continuos writing of data in VPMAR fifo. There is no one cycle delay required between the successive writing of data to VPMARB. Same is the case for reading the PM values.	In VIAC RTL, There is need of one cycle delay in between two successive writing/reading of data in VPMAR fifo. Other wise corrupted data will be stored in PM RAM.	The VIAC RTL has 2 PM memory bank of 32 location each (22X32). it keep the path values of each state during the trellis. The writing in PM ram is in two stage. Write into register & then write from register to memory.The writing from vbmar_b reg to internal memory is of two cycles. If we want to change the Delay in the RTL, implementation of the Internal memory has to be changed

Chapter 20
DSP Debug Module (PDDM)

Revision History

Revision	Date	Author	Changes
1.0	8/15/2000	Janet King	Initial Release for Hip6w. Change bars represent the changes from the CDR3 ROM device.
1.1	9/15/2000	Brian Einloth	Revised for LCA Neptune
1.2	11/27/2000	Vesy Mou	Added the port list
1.3	2/15/01	Shannon Osgood/Kathy Flories	Update Pin List Table format, updated DDTS Action item, ID# DSPhl10326
1.4	10/16/2001	Rakesh Adhikari	Updated the port list for Patriot Indy.
1.5	11/28/2001	Rakesh Adhikari	To support the amount of RAM needed in the Patriot Bravo with the new DSP memory architecture, the paging scheme has been expanded.
1.6	02/23/2002	Rakesh Adhikari	Fixed the DDTS #DSPhl13154, PDDM will generate watchpoint for program memory read and write.
1.7	04/21/2002	Rakesh Adhikari	Fixed the DDTS # DSPhl13742.
1.8	04/24/2002	Rakesh Adhikari	Fixed the DDTS # DSPhl14149 & DSPhl13910
1.9	08/13/2002	Rakesh Adhikari	Implemented the patch logic for Neptune LTE.
2.0	10/31/2002	Scott King	Clarified that patch logic is in Neptune LTE and Neptune ULS but not in Neptune LTS
2.1	12/04/02	Scott King, Shanon Osgood	Updated per DDTS# DSPhl15751. Updated paged memory address.
2.2	05/07/03		Updated for LTE specification release and DDTS# DSPhl15841.

20.1 Introduction

The DSP Debug Module (DDM) is designed to enhance the existing DSP56600 family OnCE capabilities. It allows developers to debug DSP applications in real time. The DDM is designed as a separate DSP peripheral, which is programmed and controlled by the 56600 Core.

DDM provides the following debug capabilities:

- Program address tracing (16 bit width program address and 3 Paging Bits).
- Program address and paging bits watch point.
- Two data address range watchpoints (X and Y) for any combination of Read/Write accesses.
- Two bit-maskable data watchpoints (X and Y) for any combination of Read/Write accesses.
- Programmable generation of debug event or interrupt upon match.
- Ability to optionally toggle external I/O pin on match detection.
- Paging mechanism control.

20.2 PDDM Module Pin List

Port definitions for the PDDM are provided in Table 20-1.

Table 20-1. PDDM Module Pin List

Pin Name	Direction	Description	signal_name	envelope _pin
mcab_wr[6:0]	Input	core write sector address bit		
gclkw_b	Input	Inverted DSP clock		
mcsely_wr	Input	Y I/O write select		
gdb_wr[15:0]	Input	I/O Space Write Data Bus		
mcab_rd[6:0]	Input	core read sector address bits		
mcsely_rd	Input	Y I/O read select		
mcdis0	Input	clock disable		
mcdis1	Input	clock disable		
pivrd	Input	Peripherals Interrupt Vector Read		
pivack	Input	Peripherals Interrupt Acknowledge		
pires	Input	Software reset		
res	Input	Hardware reset		
sonyxu_arywr	Input	Y Memory Write		
sonyxu_arxwr	Input	X Memory Write		

Table 20-1. PDDM Module Pin List

Pin Name	Direction	Description	signal_name	envelope _pin
sonyxu_aryrd	Input	Y Memory Read		
sonyxu_arxrd	Input	X Memory Read		
sonyxu_apmrpm	Input	Program Memory Read		
sonyxu_apwr	Input	Program Memory Write		
sonyxu_ate	Input	Address Tracing Enable Bit of OMR		
sonyxu_ydb_wr[15:0]	Input	Y Memory Write Data bus		
sonyxu_ydb_rd[15:0]	Input	Y Memory Read Data bus		
sonyxu_yab[15:0]	Input	Y Memory Address bus		
sonyxu_xdb_wr[15:0]	Input	X Memory Write Data bus		
sonyxu_xdb_rd[15:0]	Input	X Memory Read Data bus		
sonyxu_xab[15:0]	Input	X Memory Address bus		
sonyxu_pab[15:0]	Input	Program Memory Address bus		
vab_base_add[7:1]	Input	Vector address bus		
bank0_base_ref[6:3]	Input	Register Bank address bus 0		
bank1_base_ref[6:3]	Input	Register Bank address bus 1		
scan_mode,	Input	Scan Mode select		
scan_en	Input	global scan enable signal		
spmbif_scan_tristate_ enable	input	tristate enable for scan mode, It should be “1” for normal operation		
pivsse	Input	Peripheral Interrupt Vector Select		
gdb_rd[15:0]	Output	I/O Space Read data bus		
mirq_b,	Output	peripheral module interrupt request		
vab_rd[7:1]	Output	interrupt vector bus		
pddm_pw_pb[2:0]	Output	Paging bits		
pddm_pcr[11:0]	Output	PCR bits to memory		
pddm_patch_en	Output	Patch enable, to sonyxu		
pddm_ymatch	Output	Y Memory Match		

Table 20-1. PDDM Module Pin List

Pin Name	Direction	Description	signal_name	envelope _pin
pddm_pmatch	Output	Program Memory Match		
pddm_xmatch	Output	X Memory Match		
pddm_eg_dreq_b	Output	Debug event output		

20.3 DSP Memory Paging

In order to enable the system to use more than 64 KWords Program Memory space, a DSP memory paging mechanism is implemented. The paging mode method defines a common 32Kx24 shared across all address pages from $0000 to $7FFF. The remaining 32Kx24 of program address space is divided into 8Kx24 blocks. Page configuration is defined by the Page Configuration Register (PCR) to select which of the 8Kx24 blocks is contained in the DSP memory map at any given time.

Table 20-2 summarizes how the Page Select bits in the PCR are allocated to the different addresses of the program address space. Please note that, if actual memory space does not exist for a selected page and page section, the actual behavior is not defined. Please refer to Chapter 6, “Chip Configuration and Memory Maps (MEMMAP),” for the exact mapping of the DSP Program Memory in Patriot Indy.

Table 20-2. DSP Program Memory Allocation

Program Address (PA15-PA0)	Page Select Bits (see Table 20-8 "PCR Description")
$0000-$7FFF	Common Memory
Bank A: $8000-$9FFF	{AHSEL,ALSEL[1:0]}
Bank B: $A000-$BFFF	{BHSEL,BLSEL[1:0]}
Bank C: $C000-$DFFF	{CHSEL,CLSEL[1:0]}
Bank D: $E000-$FEFF	{DHSEL,DLSEL[1:0]}

20.4 Data Watchpoint (DW)

The Data Watchpoint (DW) is capable of generating a match event on any combination of partial data **and** data address range **and** read/write. This match event is able to generate a DSP Core interrupt, generate a DSP Core debug event (DE), and/or be routed to an external pin. All masks, match values, address ranges and match event actions are software configurable. The DW allows simultaneous watch of both X and Y data spaces. Routing of match events to external pins is selectable through the GPIO Module. A block diagram of Data Watchpoint is shown in Figure 20-1., "Data Watchpoint (DW) Block Diagram".

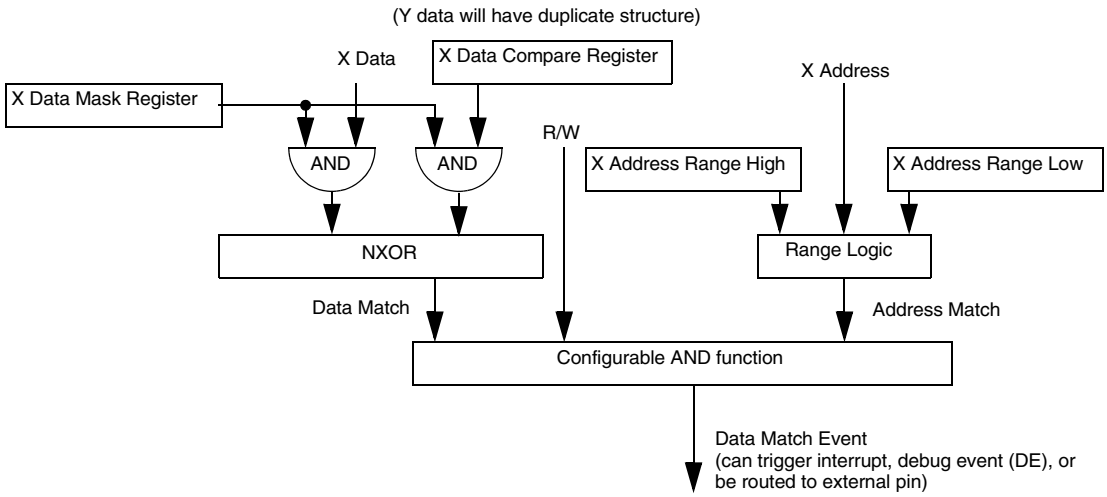


Figure 20-1. Data Watchpoint (DW) Block Diagram

20.5 Program Address Trace (PAT)

The Program Address Trace (PAT) gives the developer the ability to trace DSP program flow by utilizing a logic analyzer that captures the **3** page bits and **16** program address bits routed to externally visible pins. These bits could be multiplexed with other signals; the multiplexer selection is defined in the GPIO Module.

The functionality of the PAT is controlled by the ATE bit of the OMR (DSP Core Mode register). The synchronization of the Program Address Bus with the 3 Paging Bits is also done in this module, so that the Paging Bits may be directly routed to the external I/O and their timing adjusted to the program address appearance on the external pins. DSP_AT pin can be used as a strobe for the Logic Analyzer. The external paging pins, PAGE[2:0], can be decoded according to Table .

Table 20-3. External Paging Pins Decoding

HSEL	LSEL1	LSEL0	DESCRIPTION
0	0	0	page 0 active
0	0	1	page 1 active
0	1	0	page 2 active
0	1	1	page 3 active
1	0	0	page 4 active
1	0	1	page 5 active
1	1	0	page 6 active
1	1	1	page 7 active

20.6 Program Watchpoint (PW)

The Program Watchpoint (PW) is capable of generating a match event on any combination of program address and paging bits during program memory read and write accesses. This match event will be able to generate a DSP core interrupt, or generate a DSP core debug event (DE), as it is defined in DCSR. Routing of this match event to an external pin will be selectable through the GPIO Module.

NOTE:

When using Program Watchpoint feature it must be taken into account that not all instruction whose address appears on program address bus (PAB) are executed by the Core.

20.6.1 Program Watchpoint Controls

Program watch point logic is controlled by the control/status bits of the DCSR. Program address matching is enabled when PAME bit of DCSR is set. The match event is defined by EOMS bits of the DCSR.

20.7 PDDM Patch Logic

PDDM patch logic enables user to generate the patch for the desired page(0 to 7) of program memory. The top half of P memory is pageable, if an address in that area needs to be patched, it can be done by programming the PATCH0 and PATCH1 registers. This patch logic provides the user a way to fix the program code in the on-chip ROM without generating the new mask. The Programmer should program the pddm's patch0 and patch1 register properly before generating the patch for pageable and non-pageable memory.

Table 20-4 describes the BANKSEL[1:0] bits of PATCH1/0 registers.

Table 20-4. BANKSEL[1:0]

BANKSEL[1]	BANKSEL[0]	Description
0	0	Bank A is selected
0	1	Bank B is Selected
1	0	Bank C is Selected
1	1	Bank D is Selected

20.8 Programming Model

20.8.1 Register Summary

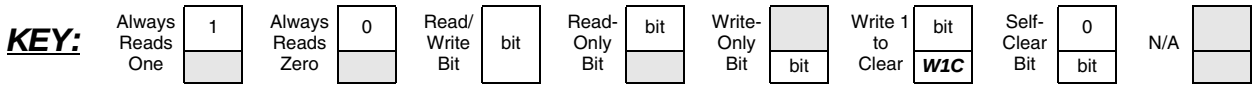


Table 20-5. PDDM Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PATCH0	R	RESERVED		PAGESEL1[2:0]			BANKSEL1[1:0]		EN1	RESERVED		PAGESEL0[2:0]			BANKSEL0[1:0]		EN0
(Y:$FFE3)	W																
PATCH1	R	RESERVED		PAGESEL3[2:0]			BANKSEL3[1:0]		EN3	RESERVED		PAGESEL2[2:0]			BANKSEL2[1:0]		EN2
(Y:$FFE4)	W																
PCR	R	RESERVED			PBH	DHSEL	CHSE	BHSEL	AHSEL	DLSEL[1:0]		CLSEL[1:0]		BLSEL[1:0]		ALSEL[1:0]	
(Y:$FFE5)	W																
DCSR	R	EOMS[1:0]		PMS	YMS	XMS	PB [1:0]		PAME	YATS [1:0]		YDME	YARME	XATS [1:0]		XDME	XARME
(Y:$FFE6)	W			W1C	W1C	W1C											
PACMPR	R	PACMPR [15:0]															
(Y:$FFE7)	W																
YARRL	R	YARRL [15:0]															
(Y:$FFE8)	W																
YARRH	R	YARRH [15:0]															
(Y:$FFE9)	W																
YDCMPR	R	YDCMPR [15:0]															
(Y:$FFEA)	W																
YDMR	R	YDMR [15:0]															
(Y:$FFEB)	W																
XARRL	R	XARRL [15:0]															
(Y:$FFEC)	W																
XARRH	R	XARRH [15:0]															
(Y:$FFED)	W																
XDCMPR	R	XDCMPR [15:0]															
(Y:$FFEE)	W																
XDMR	R	XDMR [15:0]															
(Y:$FFEF)	W																

NOTE:

The clock to the above registers will automatically be disabled during DSP WAIT mode.

20.8.2 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various DDM registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

Patch0 and Patch1 register decides which portion of pageable area is patched. There are four patch registers in sonyxu (PAR0 to PAR3), So PDDM generates four enables for each register to enable patching of pageable area

PATCH0 DDM Patch0 Register **Addr Y:$FFE3**

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	RESERVED		PAGESEL1[2:0]			BANKSEL1[1:0]		EN1	RESERVED		PAGESEL0[2:0]			BANKSEL0[1:0]		EN0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-6. PATCH0 Description

Name	Description	Settings
Bits [15:14]	Reserved Bits	
PAGESEL1 Bit 13-11	These three bit selects the page, for which we have generate a patch using PAR1 register.	Please refer to Table 20-2
BANKSEL1 Bit 10-9	These bits selects the bank from the selected page	Please refer to Table 20-3
EN1 Bit 8	This bit decides whether the patch is in pageable area or not.	0 -patch is in lower half of program memory 1 -patch is in pageable area.
Bits[7:6]	Reserved Bits	
PAGESEL0 Bits 5-3	These three bit selects the page, for which we have generate a patch using PAR0 register.	Please refer to Table 20-2
BANKSEL0 Bit 2-1	These bits selects the bank from the selected page	Please refer to Table 20-3
EN0 Bit 0	This bit decides whether the patch is in pageable area or not.	0 -patch is in lower half of program memory 1 -patch is in pageable area.

PATCH1 DDM Patch1 Register **Addr Y:$FFE4**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	RESERVED		PAGESEL3[2:0]			BANKSEL3[1:0]		EN3	RESERVED		PAGESEL2[2:0]			BANKSEL2[1:0]		EN2
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-7. PATCH Description

Name	Description	Settings
Bits [15:14]	Reserved Bits	
PAGESEL3 Bit 13-11	These three bit selects the page, for which we have generate a patch using PAR3 register.	Please refer to Table 20-2
BANKSEL3 Bit 10-9	These bits selects the bank from the selected page	Please refer to Table 20-3
EN3 Bit 8	This bit decides whether the patch is in pageable area or not.	0 -patch is in lower half of program memory 1 -patch is in pageable area.
Bits[7:6]	Reserved Bits	
PAGESEL2 Bits 5-3	These three bit selects the page, for which we have generate a patch using PAR2 register.	Please refer to Table 20-2
BANKSEL2 Bit 2-1	These bits selects the bank from the selected page	Please refer to Table 20-3
EN2 Bit 0	This bit decides whether the patch is in pageable area or not.	0 -patch is in lower half of program memory 1 -patch is in pageable area.

PCR is a 16-bit read/write register which controls the paging mechanism. The PCR is cleared at reset. PCR[11:0] bits are directly going to the output of the PDDM module.The decoding will be done in the memory itself.

PCR DDM Page Configuration Register **Addr Y:$FFE5**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	RESERVED			PBH	DHSEL	CHSEL	BHSEL	AHSEL	DLSEL[1:0]		CLSEL[1:0]		BLSEL[1:0]		ALSEL[1:0]	
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-8. PCR Description

Name	Description	Settings
Bits [15:13]	Reserved Bits	
PBH Bit 12	This Bit will combine with DCSR[10:9],Paging bits, and Generates match events for different pages.	See the description of DCSR PB[1:0] Bits.
DHSEL Bit 11 **DLSEL[1:0]** Bit 7-6	**Bank D Select** — The Bank D Select bits are used to select which bank will be contained in the DSP memory map from P:$E000 - $FEFF. **Note:** This block is only 7.75Kx24 because the upper 256x24 from P:$FF00 - $FFFF is reserved.	Please refer to Table 20-2 {DHSEL,DLSEL[1:0]}
CHSEL Bit 10 CLSEL[1:0] Bits 5–4	**Bank C Select** — The Bank C Select bits are used to select which bank will be contained in the DSP memory map from P:$C000 - $DFFF.	Please refer to Table 20-2 {CHSEL,CLSEL[1:0]}
BHSEL Bit 9 BLSEL[1:0] Bits 3–2	**Bank B Select** — The Bank B Select bits are used to select which bank will be contained in the DSP memory map from P:$A000 - $BFFF. The table to the right describes the bit encoding.	Please refer to Table 20-2 {BHSEL,BLSEL[1:0]}
AHSEL Bit 8 ALSEL[1:0] Bits 1-0	**Bank A Select** — The Bank A Select bits are used to select which Bank will be contained in the DSP memory map from P:$8000 - $9FFF. The table to the right describes the bit encoding.	Please refer to Table 20-2 {AHSEL,ALSEL[1:0]}

The DCSR is a16-bit read/write register which controls the operation of the DSP debug block. The DCSR is cleared at reset.

DCSR DDM Debug Control/Status Register **Addr Y:$FFE6**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	EOMS[1:0]		PMS	YMS	XMS	PB[1:0]		PAME	YATS[1:0]		YDME	YARME	XATS[1:0]		XDME	XARME
TYPE:	rw	rw	w1c	w1c	w1c	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-9. DCSR Description

Name	Description	Settings
EOMS [1:0] Bit 15-14	**Event On Match Selection**—The Event On Match Select bits are used to select the event that will be generated when data watchpoint or program address watchpoint event occurs. The table to the right describes the bit encoding. **Note:** 1. External pin is asserted for 2 DSP clock cycles.	**EOMS[1] / EOMS[0] / Description** 0 / 0 / External pin assertion only 0 / 1 / Enables DSP Debug Event (DE) on Match + External pin assertion 1 / 0 / Enables DSP Core Interrupt on Match + External pin assertion 1 / 1 / Reserved
PMS Bit 13	**P Match Status**—PMS status bit indicates whether P address watchpoint event had occurred. It is cleared by writing ‘1’ to it.	‘0’ - P address watchpoint event did not occur. ‘1’ - P address watchpoint event did occur.
YMS Bit 12	**Y Match Status**—YMS status bit indicates whether Y space access watchpoint event had occurred. It is cleared by writing ‘1’ to it.	‘0’ - Y space access watchpoint event did not occur. ‘1’ - Y space access watchpoint event did occur.
XMS Bit 11	**X Match Status—**XMS status bit indicates whether X space access watchpoint event had occurred. It is cleared by writing ‘1’ to it.	‘0’ - X space access watchpoint event did not occur. ‘1’ - X space access watchpoint event did occur.

Table 20-9. DCSR Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>PB [1:0]
Bit 9-10</td><td>Paging Bits—The Paging Bits are used to select which page will trigger a Program Address Match event.
Note: The PBH bit of the PCR is combine with the PB[1:0] bits of the DCSR to
form a 3 bit field that select which page will trigger a Program Address Match event. The bit order would be: {PBH, PB[1:0]}</td><td>
<table>
<tr><th>PB2</th><th>PB1</th><th>PB0</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Page 0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Page 1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Page 2</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Page 3</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Page 4</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Page 5</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Page 6</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Page 7</td></tr>
</table>
</td></tr>
<tr><td>PAME
Bit 8</td><td>Program Address Match Enable—The Program Address Match Enable bit is used to enable the program access detection. The value of the P memory address is compared to the value of the PACMPR together with the Paging Bits.</td><td>0’ - program access match is disabled
‘1’ - program access match is enabled</td></tr>
<tr><td>YATS[1:0]
Bits 6-7</td><td>Y Access Type Selection—The Access Type Selection bits are used to select the type of Y memory access that should trigger Data Watchpoint event. The possible access types are Read Access, Write Access and Read or Write Access. The table to the right explains the YATS bit encoding.</td><td>
<table>
<tr><th>YATS[1]</th><th>YATS[0]</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>Read or Write Access</td></tr>
<tr><td>0</td><td>1</td><td>Read Access</td></tr>
<tr><td>1</td><td>0</td><td>Write Access</td></tr>
<tr><td>1</td><td>1</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>YDME
Bit 5</td><td>Y Data Match Enable— YDME enables data match detection for Y memory space accesses. The value of the current access data is compared to the reference value programmed in YDCMPR after it has been masked with the value written in YDMR.</td><td>‘0’ - Y data match is disabled
‘1’ - Y data match is enabled</td></tr>
<tr><td>YARME
Bit 4</td><td>Y Address Range Match Enable—YARME enables address range match detection for Y memory space accesses. The value of the current access address is compared to the reference values programmed into YARRL and YARRH.
Note: In order to match one specific address, both YARRH and YARRL must be initialized with the value of this address.</td><td>‘0’ - Y address range match is disabled
‘1’ - Y address range match is enabled</td></tr>
</table>

Table 20-9. DCSR Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>XATS[1:0]
Bits 2-3</td><td>X Access Type Selection—The Access Type Selection bits are used to select the type of X memory access that should trigger Data Watchpoint event. The possible access types are Read Access, Write Access and Read or Write Access. The table to the right explains the XATS bit encoding.</td><td>
<table>
<tr><th>XATS[1]</th><th>XATS[0]</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>Read or Write Access</td></tr>
<tr><td>0</td><td>1</td><td>Read Access</td></tr>
<tr><td>1</td><td>0</td><td>Write Access</td></tr>
<tr><td>1</td><td>1</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>XDME
Bit 1</td><td>X Data Match Enable—XDME enables data match detection for X memory space accesses. The value of the current access data is compared to the reference value programmed in XDCMPR after it has been masked with the value written in XDMR.</td><td>‘0’ - X data match is disabled
‘1’ - X data match is enabled</td></tr>
<tr><td>XARME
Bit 0</td><td>X Address Range Match Enable—XARME enables address range match detection for X memory space accesses. The value of the current access address is compared to the reference values programmed into XARRL and XARRH.
Note: In order to match one specific address both XARRH and XARRL must be programmed with the same value.</td><td>‘0’ - X address range match is disabled
‘1’ - X address range match is enabled</td></tr>
</table>

PACMPR DDM Program Address Compare Register **Addr Y:$FFE7**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PACMPR [15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-10. PACMPR Description

Name	Description	Settings
PACMPR [15:0] Bits 15-0	**Program Address Compare Register**—A 16-bit read/write register which holds the desired program address to be matched. The Paging Bits of DCSR are used to determine the desired page to be matched. The register is cleared during hardware reset.	N/A

YARRL DDM Y Address Range Low Register **Addr Y:$FFE8**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	YARRL [15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-11. YARRL Description

Name	Description	Settings
YARRL [15:0] Bits 15-0	**Y Address Range Low Register**—A 16-bit read/write register which holds the lower boundary of the desired Y address range to be matched. This register is cleared during hardware reset.	N/A

YARRH DDM Y Address Range High Register **Addr Y:$FFE9**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	YARRH [15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-12. YARRH Description

Name	Description	Settings
YARRH [15:0] Bits 15-0	**Y Address Range High Register**—A 16-bit read/write register which holds the upper boundary of the desired Y address range to be matched. This register is cleared during hardware reset.	N/A

YDCMPR DDM Y Data Compare Register **Addr Y:$FFEA**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	YDCMPR[15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-13. YDCMPR Description

Name	Description	Settings
YDCMPR [15:0] Bits 15-0	**Y Data Compare Register**—A 16-bit read/write register which holds the value to be compared with the Y data value. This register is cleared during hardware reset.	N/A

YDMR DDM Y Data Mask Register **Addr Y:$FFEB**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	YDMR[15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 20-14. YDMR Description

Name	Description	Settings
YDMR [15:0] Bits 15-0	**Y Data Mask Register**—A 16-bit read/write register. The value of YDMR is ANDed bitwise with the current value of the Y Data bus and with the value of the Y Data Compare Register (YDCMPR). The results are the inputs to the Y data comparator which can generate Data Match in case the inputs are equal. The register is set to $FFFF during hardware reset (no bits are masked).	'0' - mask the corresponding data bit '1' - do not mask the corresponding data bit

XARRL DDM X Address Range Low Register **Addr Y:$FFEC**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	XARRL[15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-15. XARRL Description

Name	Description	Settings
XARRL [15:0] Bits 15-0	**X Address Range Low Register**—A 16-bit read/write register which holds the lower boundary of the desired X address range to be matched. This register is cleared during hardware reset.	N/A

XARRH DDM X Address Range High Register **Addr Y:$FFED**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	XARRH[15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-16. XARRH Description

Name	Description	Settings
XARRH [15:0] Bits 15-0	**X Address Range High Register**—A 16-bit read/write register which holds the upper boundary of the desired X address range to be matched. If the X address is less than or equal to XARRH and greater than or equal to XARRL a match event will be issued. This register is cleared during hardware reset.	N/A

XDCMPR DDM X Data Compare Register **Addr Y:$FFEE**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	XDCMPR[15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-17. XDCMPR Description

Name	Description	Settings
XDCMPR [15:0] Bits 15-0	**X Data Compare Register**—A 16-bit read/write register which holds the value to be compared with the X data value. This register is cleared during hardware reset.	N/A

XDMR DDM X Data Mask Register **Addr Y:$FFEF**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	XDMR[15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 20-18. XDMR Description

Name	Description	Settings
XDMAR [15:0] Bits 15-0	**X Data Mask Register**—A 16-bit read/write register. The value of XDMR is ANDed bitwise with the current value of the X Data bus and with the value of the X Data Compare Register (XDCMPR). The results are the inputs to the X data comparator which can generate a Data Match event if the inputs are equal. The register is set to $FFFF during hardware reset (no bits are masked).	'0' - mask the corresponding data bit '1' - do not mask the corresponding data bit

20.9 DDM Interrupt

The DSP Debug Block has the capability of sending one interrupt to the DSP interrupt controller. The interrupt can come from 3 sources:

- Program Address + page bits match
- X space access match
- Y space access match

This interrupt is enabled by setting the EOM[1:0] bits to '10'.

The interrupt routine should read the 3 status bits (XMS,YMS & PMS) from the DCSR register to determine what was the source for the interrupt.

For the specific interrupt vector of this interrupt refer to Chapter 6, "Chip Configuration and Memory Maps (MEMMAP)." .

20.10 Restrictions

20.10.1 Using Read-Modify-Write instructions (bset, bclr)

Using of the RMW instructions on DCSR is restricted, because of the possible status signal may be overwritten. The programmer should take care to not violate this restriction for there will be no error generated by the assembler.

20.10.2 Fetches From Program Memory Matching

When using the Program Watchpoint feature it must be taken in account that due to pipe line logic there may be instructions whose address appears on program address bus (PAB), but which are not executed by the Core. Those address can trigger program watch point event, if they match the value written into PACMPR and PAME bit is set.

20.10.3 Paged Memory Access Delay after PCR Write

Due to DSP pipeline constraints, a delay of at least three instruction cycles after writing the PCR is required before the paged memory can be accessed. This constraint can be satisfied by placing five NOP instructions after a write to the PCR.

20.10.4 Program Watch Point Delay After DCSR Write

Program address watch point must be placed at least six locations after the write to PAME bit in DCSR location. This constraint can be satisfied by placing six NOP instructions after a write to the DCSR instruction.

20.10.5 Paging Bits Delay After OMR Write

Paging bits value for PW match will be valid three cycles after ATE and PTE bits in the OMR are set.

20.10.6 Changing The EOMS Bits in DCSR

Changing the value of EOMS bits can be done only when the PDDM is disabled (the XARME,YARME,XDME,YDME,PAME bits in the DCSR and PTE are cleared).

20.10.7 Changing PACMPR

Program Match is missed if the value of Program Address Compare Register is so programmed so as to change just one clock cycle later to the timestamp at which the address to be compared appears on pab[15:0]. This can be done if the instruction to change the value of PACMPR is given a few cycles earlier to the time stamp at which the address to be compared is expected to arrive at pab[15:0]. PACMPR can only be changed two clock cycles later to the address, for which match is supposed to be generated, actually appears on the address bus(pab). Ideally, PACMPR should not be changed until a few clock cycles later to the event at which the match, supposed to be generated, actually occurs. (See DDTS DSPhl13910)

20.10.8 Patch Generation Restriction

After writing the patch0/1 register at least 6 “nop” delay should be given before generating any patch.This restriction is because of DSP pipeline constraints.

20.11 Mismatch at chip periphery (in Indy)

The pddm module is generating the “pddm_pw_pb” signal which is used for tracing the page. This signal is generated using “pab” (program address bus) and then routed through GPIO at the chip periphery for debug purposes. One cycle delayed version of pab i.e bab[15:0] is actually what is brought at the chip periphery. So logically there is a complete one cycle mismatch between the paging bits(pddm_pw_pb) and the corresponding address bus(bab[15:0]).

Chapter 21
Universal Serial Bus Interface (USB)

Revision History Table

Revision	Date	Author	Changes
0	12/13/2001		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	3/26/2002	John Oakley	Updated Base Address Updated port list to IP interface Removed scan data ports Fixed interrupt register reset values, per DDTS: TLSbo22070
0.2	4/1/2002	John Oakley	Added USB_CPU_CTL register from INDY design Added MCU memory map select Updated memory map per above
0.3	4/4/2002	John Oakley	Updated 3 port names to match IP Added notes about end the ISR for DSP isochronous endpoints 5 & 6 (per DSPhl12506)
0.4	4/22/2002	John Oakley	Updated MCU Mem. Map Diagram Updated register descriptions
0.5	5/3/2002	John Oakley	Updated USBCPUCTL register Review updates to registers
0.6	5/30/2002	Scott Wolf	Updated USB Memory Map added Supported USB Device Speeds section. Reorganized the USB Control Registers and the Detailed Register Descriptions. Updated USB and Tranceiver Pin Lists. Other editorial changes.
0.7	6/14/2002	Scott Wolf	Changed USBCPUCTL register name to USBCPUCR to the name given in the Memory Map chapter. Other editorial changes.
0.8	05/07/03		Updated for LTE specification release.

21.1 Description

The USB peripheral provides the required buffering and protocol to communicate on the Universal Serial Bus. This peripheral incorporates all the logic required to support a USB Device (excluding HUBs) as defined in the USB 1.1 Specification.

21.2 Module Overview

21.2.1 Features

Features of the USB Model include the following:

- Implementation
  - — Support for Twelve (12) Endpoints
  - — Full/Low speed modes (12Mbit/1.5Mbit)
- Serial Bus Interface Engine (SIE)
  - — Packet decoding/generation
  - — CRC generation and checking
  - — STALL, NAK, and ACK handshake generation
  - — NRZI encoding/decoding
  - — Bit-stuffing
- Suspend and resume operations
  - — Remote Wakeup support
- Transaction interrupt driven
- USB generated interrupts
  - — Resume, Suspend
  - — Start of Frame
  - — Transaction complete
  - — Reset

21.2.2 Background

This USB module is designed to support the entire range of USB devices (HUB functionality is not included; furthermore, this module does not act as a Host controller). All four types of USB data transfers are supported: control, isochronous, interrupt, and bulk. Endpoint 0 always functions as a bidirectional control endpoint with an eight (8) byte buffer.

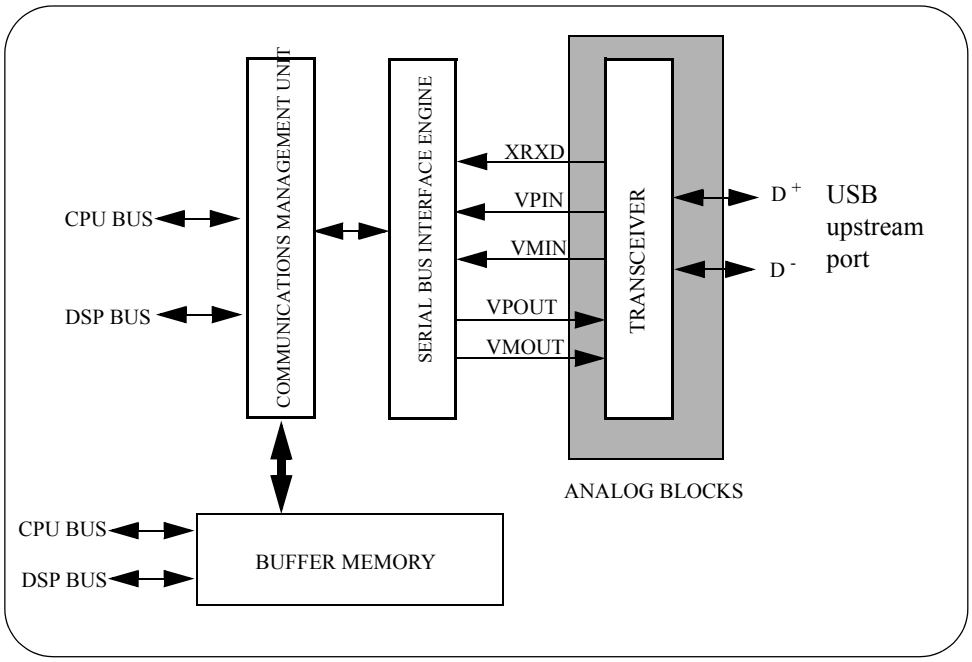


Figure 21-1. USB Block Diagram

A block diagram of the USB model is shown in Figure 21-1. The model is partitioned into three functional blocks. These blocks consist of the serial bus interface engine (SIE), the communications management unit (CMU), and the buffer memory. The analog blocks shown are not part of the model, but are necessary for a complete USB device implementation.

This USB Module contains 12 endpoints. These endpoints are dedicated toward either the CPU or the DSP (all endpoints are unidirectional unless otherwise specified):

- CPU endpoints:
  - Endpoint 0 - Bi-directional Control Endpoint
  - Endpoint 1 - Programmable Bulk or Interrupt Receive Endpoint
  - Endpoint 2 - Programmable Bulk or Interrupt Transmit Endpoint
  - Endpoint 3 - Programmable Bulk or Interrupt Receive Endpoint
  - Endpoint 4 - Programmable Bulk or Interrupt Transmit Endpoint
  - Endpoint 10 - Interrupt Transmit Endpoint
- DSP Endpoints:
  - Endpoint 5 - Isochronous Receive Endpoint
  - Endpoint 6 - Isochronous Transmit Endpoint
  - Endpoint 7 - Programmable Bulk or Interrupt Receive Endpoint
  - Endpoint 8 - Programmable Bulk or Interrupt Transmit Endpoint
  - Endpoint 9 - Interrupt Transmit Endpoint
  - Endpoint 11 - Interrupt Transmit Endpoint

The USB Module also provides Neptune the buffering required for these Endpoints. The buffering requirements are as follows:

- CPU Endpoints
  - Endpoint 0 - 16 bytes (8 bytes receive, 8 bytes transmit)
  - Endpoint 1 - 16 bytes (or 32 bytes, selectable via the MEMMAP bit)
  - Endpoint 2 - 16 bytes (or 32 bytes, selectable via the MEMMAP bit)
  - Endpoint 3 - 16 bytes
  - Endpoint 4 - 32 bytes
  - Endpoint 10 - 8 bytes
- DSP Endpoints
  - Endpoint 5 - 32 bytes (16 bytes double buffered)
  - Endpoint 6 - 32 bytes (16 bytes double buffered)
  - Endpoint 7 - 16 bytes
  - Endpoint 8 - 16 bytes
  - Endpoint 9 - 16 bytes
  - Endpoint 11 - 8 bytes

Endpoint 0, by USB definition, is always a bidirectional endpoint. This has been allocated to the CPU. It is expected that the CPU will provide the proper messaging to/from the DSP to handle Standard and non-Standard USB Device requests. The CPU also contains two pairs of RX and TX endpoints. These endpoints are programmable to be either Bulk or Interrupt endpoints. The TX endpoint 4 has a 32 byte buffer for additional bandwidth capability on the USB bus. TX endpoint 10 has an 8 byte buffer.

The DSP contains 6 endpoints. Endpoints 5 and 6 have been defined to be isochronous endpoints. Isochronous endpoints are required to be unidirectional and double-buffered. The DSP contains two unidirectional endpoints, one TX and one RX, for bulk or interrupt endpoints. There are also two unidirectional interrupt TX endpoints for feedback data to the host.

Endpoints are not shared between the CPU or DSP. The two processor interfaces do not interact within the USB other than to share the USB bus resource. Buffers are not shared and are not accessible from one domain to the other.

For the Neptune LTE IC, the CPU is the master of the IC. That is, the DSP will be controlled at the top of the IC by registers set by the CPU

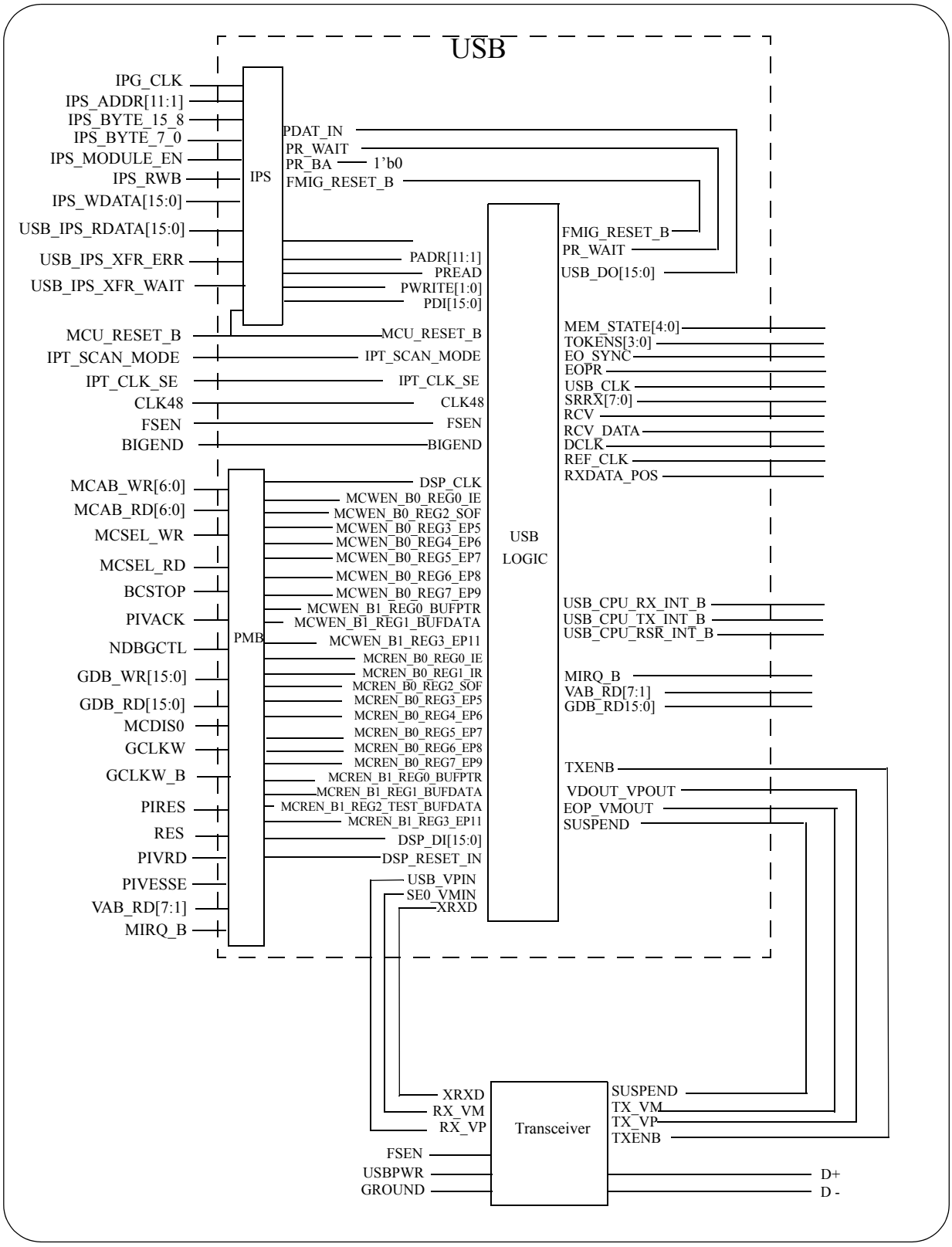


Figure 21-2. USB Peripheral

Shown in Figure is the top level of the USB block. This peripheral block contains the logic and buffers needed to implement the USB function (excluding the transceiver functions).

21.3 USB Module Pin List

Table 21-1 is a list of all the pins in the USB module.

Table 21-1. USB Module Pin List

Pin Name	Direction	Description
Configuration Signals		
bigend	Input	Endian mode; Big or Little Endian select. A “1” indicates Big endian, a “0” indicates Little endian buffer accesses for the CPU. This line is tied high in the Neptune.
fsen	Input	Full Speed Enable. When set to 1, the USB operates as a full speed device (12MHZ). When set to a 0, the USB operates as a low speed device(1.5MHZ). This is tied high in the Neptune allowing high speed mode only.
IP BUS Interface Signals		
ips_wdata[15:0]	Input	Write data bus
ips_module_en	Input	Module select
ips_addr[4:2]	Input	Address bus
ips_byte_15_8	Input	Byte [15:8] enable
ips_byte_7_0	Input	Byte [7:0] enable
ips_rwb	Input	Module read/write signal
usb_ips_rdata[15:0]	Output	Read data bus from USB
usb_ips_xfr_wait	Output	USB transfer wait output
usb_ips_xfr_err	Output	USB transfer error output
CPU Clock and Reset Signals		
ipg_clk	Input	IP bus clock; Chip level MCU clocks. These clocks are in phase with the MCU core.
clk48	Input	USB clock; Four times oversampling clock used by the digital PLL for clock recovery from the data stream. This is a 48Mhz input. The USB logic will divide this by 8 if FSEN = 0.

Table 21-1. USB Module Pin List

Pin Name	Direction	Description
mcu_reset_b	Input	Asynchronous reset; When active, all registers are reset to their defined initial values. The USB Buffers are unaffected. This includes both the DSP and the CPU registers. Active low signal.
CPU Interruipt Signals		
usb_cpu_rx_int_b	Output	Receive interrupt; Receive endpoint interrupt flag for endpoints assigned to the CPU. This signal is an interrupt input to the CPU whenever valid data has been received by the USB.
usb_cpu_tx_int_b	Output	Transmit interrupt; Transmit endpoint interrupt flag for endpoints assigned to the CPU. This signal is an interrupt input to the CPU whenever valid data has been successfully transmitted by the USB.
usb_cpu_rsr_int_b	Output	Reset, Suspend, Resume interrupt; USB Reset and Suspend/Resume interrupt. This is a composite interrupt for interrupts due to suspend or resume actions on the USB bus. It also includes USB Reset conditions detected on the USB data lines.
DSP Bus Interface Signals		
mcab_wr[6:0]	Input	Module Address bus write
mcab_rd[6:0]	Input	Module Address bus read
mcsel_wr	Input	Module Select write
mcsel_rd	Input	Module Select read
bcstop	Input	Clock disable
pivack	Input	Interrupt vector acknowledge
ndbgctl	Input	
gdb_wr[15:0]	Input	Module data bus write
gdb_rd[15:0]	Output	Module data bus read; Data Bus output from the USB module to the DSP.
mcdis0	Input	Clock disable
mcdis1	Input	Clock disable
DSP Clock and Reset SIgnals		
gclkw	Input	DSP clock

Table 21-1. USB Module Pin List

<table>
<tr><th>Pin Name</th><th>Direction</th><th>Description</th></tr>
<tr><td>gclkw_b</td><td>Input</td><td>DSP clock inverted</td></tr>
<tr><td>pires</td><td>Input</td><td>Asynchronous reset</td></tr>
<tr><td>res</td><td>Input</td><td>Asynchronous reset</td></tr>
<tr><td colspan="3">DSP Interrupt Signals</td></tr>
<tr><td>pivrd</td><td>Input</td><td>Interrupt vector read strobe</td></tr>
<tr><td>pivesse</td><td>Input</td><td>Interrupt vector select</td></tr>
<tr><td>mirq_b</td><td>Output</td><td>Module interrupt; All inclusive interrupt signal for all endpoints assigned to the DSP and all USB related interrupts. Used in conjunction with the DSP interrupt vector.</td></tr>
<tr><td>vab_rd[7:1]</td><td>Output</td><td>Interrupt vector address bus; Interrupt vector provided for MOT DSPs. This interrupt vector will provide the address for branches to the DSP ISR for the USB. Different vectors are provided tor the different interrupt sources.
<table>
<tr><th>Vector</th><th>Interrupt</th></tr>
<tr><td>0000</td><td>Endpoint 5</td></tr>
<tr><td>0001</td><td>Endpoint 6</td></tr>
<tr><td>0010</td><td>Endpoint 7</td></tr>
<tr><td>0011</td><td>Endpoint 8</td></tr>
<tr><td>0100</td><td>Endpoint 9</td></tr>
<tr><td>0101</td><td>SOF</td></tr>
<tr><td>0110</td><td>Suspend / Resume</td></tr>
<tr><td>0111</td><td>Reset</td></tr>
<tr><td>1001</td><td>Endpoint 11</td></tr>
</table></td></tr>
<tr><td colspan="3">Transceiver Signals</td></tr>
<tr><td>xrxd</td><td>Input</td><td>Differential Data; Differential Received Data. This is the output of the differential receiver connected to the VM and VP pins in the USB transceiver block.</td></tr>
<tr><td>se0_vmin</td><td>Input</td><td>Single ended USB data; Voltage Negative Input. This is a CMOS level input of the D- signal on the USB. Used for single ended zero detection.</td></tr>
</table>

Table 21-1. USB Module Pin List

Pin Name	Direction	Description
eop_vmout	Output	Single ended USB data; Voltage Negative Output. This signal is the negative input to the USB differential driver which drives the USB D+ and D- wires.
vdout_vpout	Output	Single ended USB data; Voltage Positive Output. This signal is the positive input to the USB differential driver which drives the USB D+ and D- wires.
usb_vpin	Input	Single ended USB data; Voltage Positive Input. This is a CMOS level input of the D+ signal on the USB. Used for single ended zero detection.
txenb	Output	Transmit enable; Active Low Transmit Enable. This signal is an input to the USB transceiver module to signal when to transmit data on the USB bus.
suspend	Output	Suspend; This signal is active high and signifies the USB device is suspended. This signal is intended to be used to deactivate system circuitry (including the USB transceiver) in order to enter a low power state.
Scan Signals		
lpt_scan_mode	Input	Scan mode
lpt_clk_se	Input	Scan enable (serial shift mode)
Debug Signals		
msm_state[4:0]	Output	Debug - Master state machine state
tokens[3:0]	Output	Debug - Encoded token type
eo_sync	Output	Debug - Start of Packet detected
eopr	Output	Debug - End of Packet detected
usb_clk	Output	Debug - USB clock
srrx[7:0]	Output	Debug - Decoded data
rcv	Output	Debug - Receive active
rcv_data	Output	Debug - Raw recovered data
dclk	Output	Debug - Recovered clock
ref_clk	Output	Debug - Reference clock
rxdata_pos	Output	Debug - Sampled input data

21.4 Transceiver Module Pin List

Table 21-2. Transceiver Module Pin List

Pin Name	Direction	Description
fsen	Input	Full Speed enable.
usbpwr	Input	Cable power supplied by USB bus.
ground	Input	Cable ground suppolied by USB bus.
tx_vp	Input	Digital positive input for the differential USB driver.
tx_vm	Input	Digital negative input for the differential USB driver.
suspend	Input	Places dirvers in transceiver to low power state.
txenb	Input	Output enable for transceiver drivers.
rx_vp	Output	Digital output of received USB D+ signal.
rx_vm	Output	Digital output of received USB D- signal.
xrxd	Output	CMOS logic value of differential received data from USB wires.
D-	Bidirectional	Negative value of differential USB bus pair.
D+	Bidirectional	Positive value of differential USB bus pair.

21.5 USB Module Operation

Figure 21-3 shows the transaction types supported by the USB model. The transactions are portrayed as error free. The effect of errors in the data flow are discussed later.

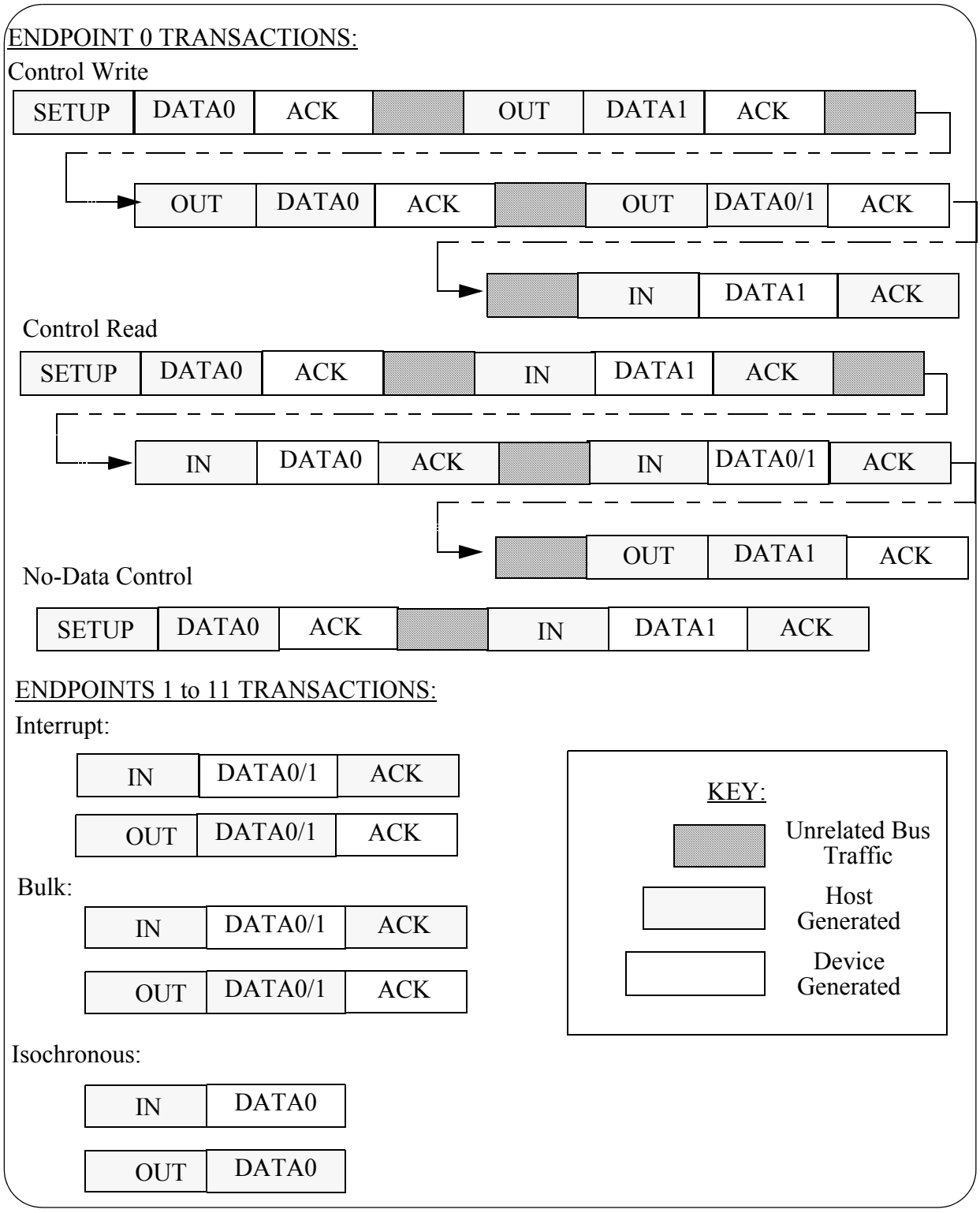


Figure 21-3. Supported Transaction Types per Endpoint

Each USB transaction is comprised of a series of packets. The USB model supports the packet types shown in Figure 21-4. Token packets are generated by the USB host and decoded by the USB device. Data and Handshake packets are both decoded and generated by the USB device depending on the type of transaction.

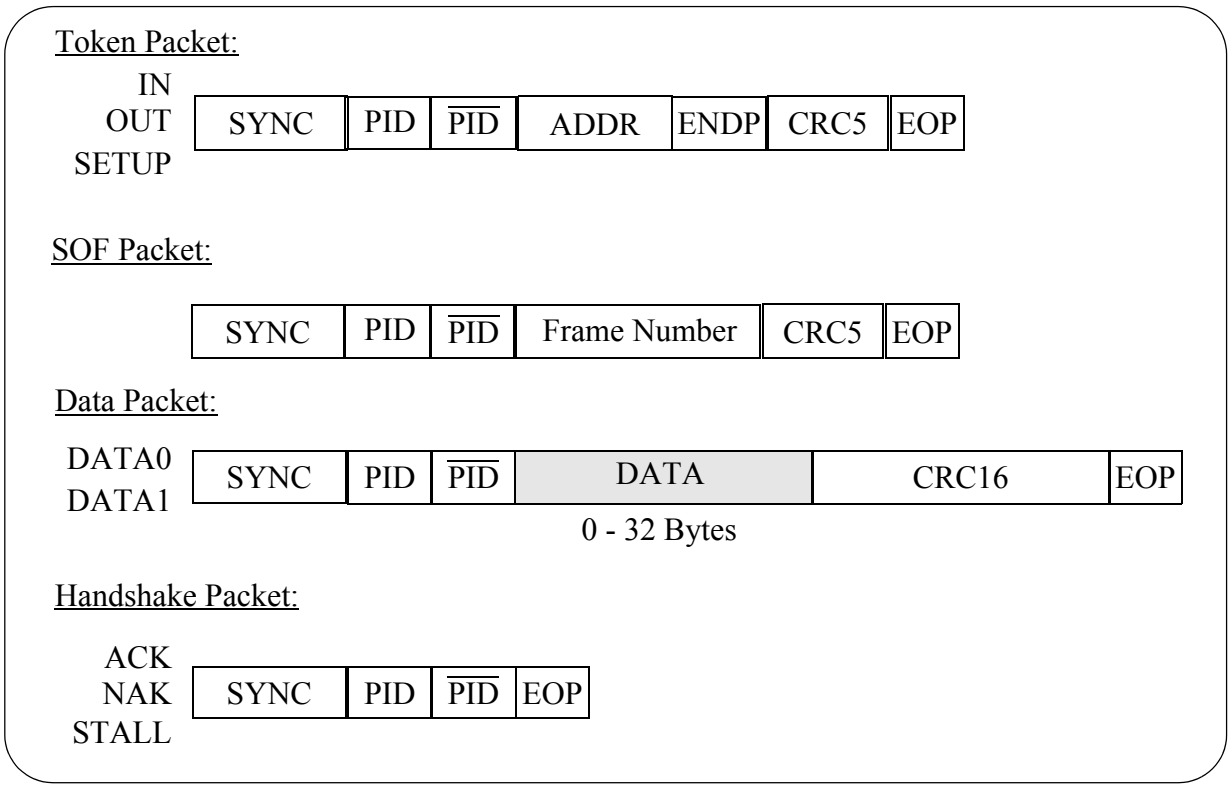


Figure 21-4. Supported USB Packet Types

21.6 Packet Fields

The following sections will give some detail on each field used to form a complete USB transaction packet.

21.6.1 Sync Pattern

The NRZI (see Section 21.7.1) bit pattern shown in Figure 21-5 is used as a synchronization pattern and is prefixed to each packet. This pattern is equivalent to a data pattern of seven 0's followed by a 1 (0x80).

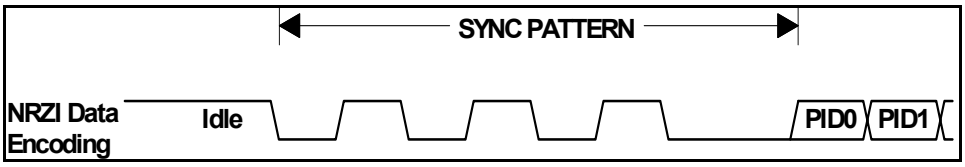


Figure 21-5. Sync Pattern

The start of a packet (SOP) is signaled by the originating port by driving the D+ and D- lines from the idle state (also referred to as the "J" state) to the opposite logic level (also referred to as the "K" state). This switch in levels represents the first bit of the Sync field. Figure 21-5 shows the data signaling and voltage levels for the start of packet and the sync pattern.

21.6.2 Packet Identifier Field

The Packet Identifier field is an eight bit number comprised of the four bit packet identification (PID) and its complement. The field follows the sync pattern and determines the direction and type of transaction on the bus. Table 21-3 shows the PID values for the supported packet types.

Table 21-3. Supported Packet Identifier

PID Value	PID Type	Description
1001	IN Token	Device to Host transaction
0001	OUT Token	Host to Device transaction
1101	SETUP Token	Host to Device transaction. Control endpoints only
0101	SOF Token	Host to Device. Signals start of 1 Ms Frame
1100	PRE Token	Host to Device. Pre-cursor to Low Speed data.
0011	DATA0 Token	Bi-directional. Data packet with sequence 0.
1011	DATA1 Token	Bi-directional. Data packet with sequence 1
0010	ACK Token	Bi-directional. Acknowledge handshake signal.
1010	NAK Token	Device to Host. Not Acknowledge handshake signal.
1110	STALL Token	Device to Host. Endpoint Stalled.

21.6.3 Address Field (ADDR)

The Address field is a seven bit number that is used to select a particular USB device. This field is compared to the lower seven bits of the FADDR register to determine if a given transaction is targeting the USB device.

21.6.4 Endpoint Field (ENDP)

The Endpoint field is a four bit number that is used to select a particular endpoint within a USB device.

21.6.5 Cyclic Redundancy Check (CRC)

Cyclic Redundancy Checks are used to verify the address and data stream of a USB transaction. This field is five bits wide for token packets and sixteen bits wide for data packets. CRCs are generated in the transmitting device and sent on the USB data lines after both the endpoint field and the data field. Figure 21-6 shows how the five bit CRC value is calculated from the data stream and verified for the address and endpoint fields of a token packet and Figure 21-7 shows how the sixteen bit CRC value is calculated and either transmitted or verified for the data packet of a given transaction.

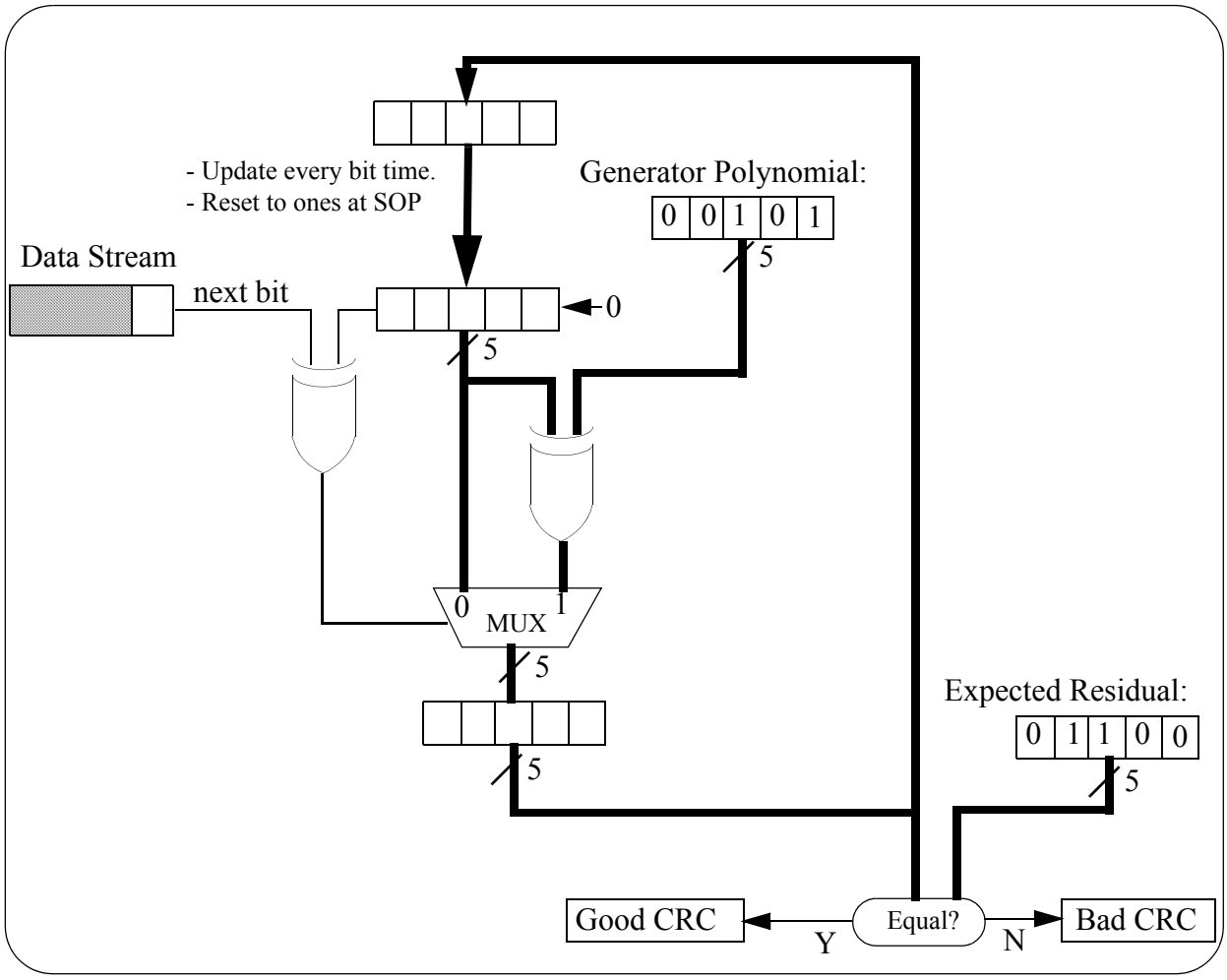


Figure 21-6. 5 bit CRC Calculation for Address and Endpoint Fields

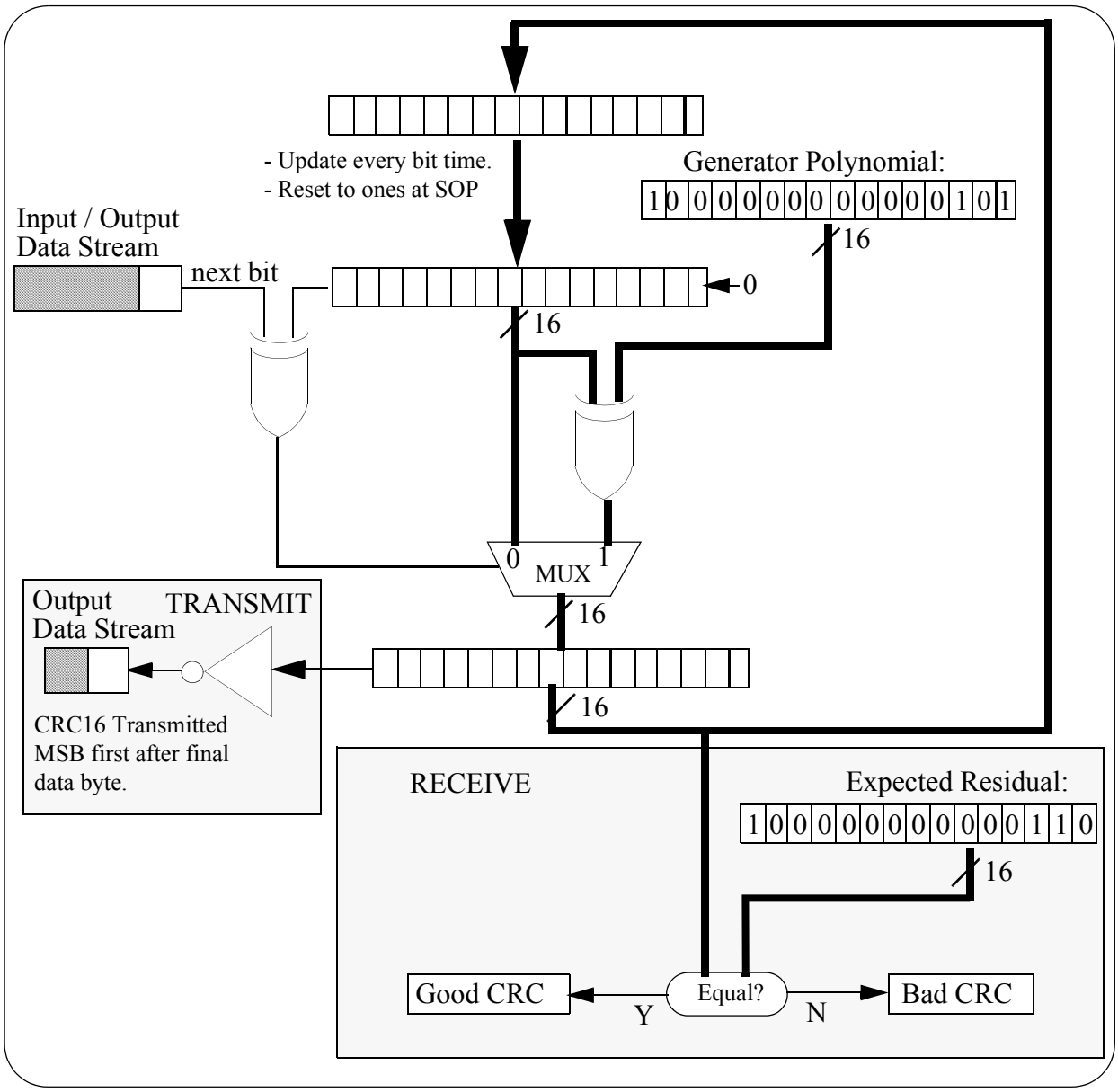


Figure 21-7. 16 bit CRC Calculation for Data Field

21.6.6 End Of Packet (EOP)

The single-ended 0 (SE0) state is used to signal an end of packet (EOP). The single-ended 0 state is indicated by both D+ and D- being below 0.8 V. EOP will be signaled by driving D+ and D- to the single-ended 0 state for two bit times followed by driving the lines to the idle state for one bit time. The transition from the single-ended 0 to the idle state defines the end of the packet. The idle state is asserted for one bit time and then both the D+ and D- output drivers are placed in their high-impedance state. The bus termination resistors hold the bus in the idle state. The width of the SE0 in the EOP is about two bit times. The EOP width is measured with the same capacitive load used for maximum rise and fall times and is measured at the same level as the differential signal crossover points of the data lines.

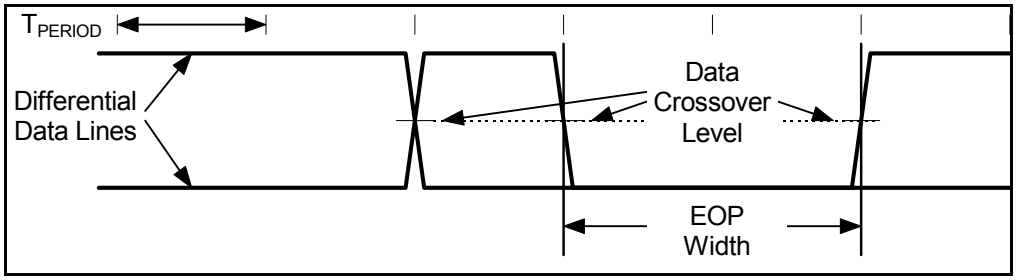


Figure 21-8. EOP Width Timing

21.7 Signaling Description

The following sections detail the signaling conventions for USB transactions. Additionally, signaling that does not constitute a transaction, but yields hardware response is defined.

21.7.1 Data Encoding/Decoding

The USB employs NRZI data encoding when transmitting packets. In NRZI encoding, a 1 is represented by no change in level and a 0 is represented by a change in level. Figure 21-9 shows a data stream and the NRZI equivalent and Figure 21-10 is a flow diagram for NRZI. The high level represents the J state on the data lines in this and subsequent figures showing NRZI encoding. A string of zeros causes the NRZI data to toggle each bit time. A string of ones causes long periods with no transitions in the data.

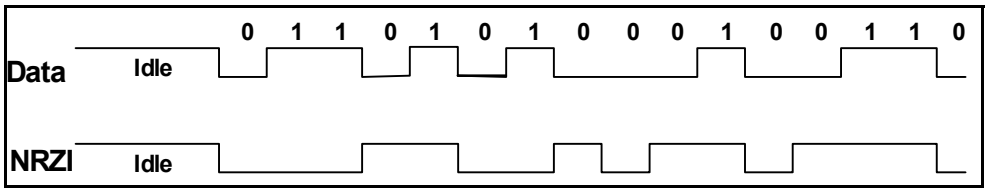


Figure 21-9. NRZI Data Encoding

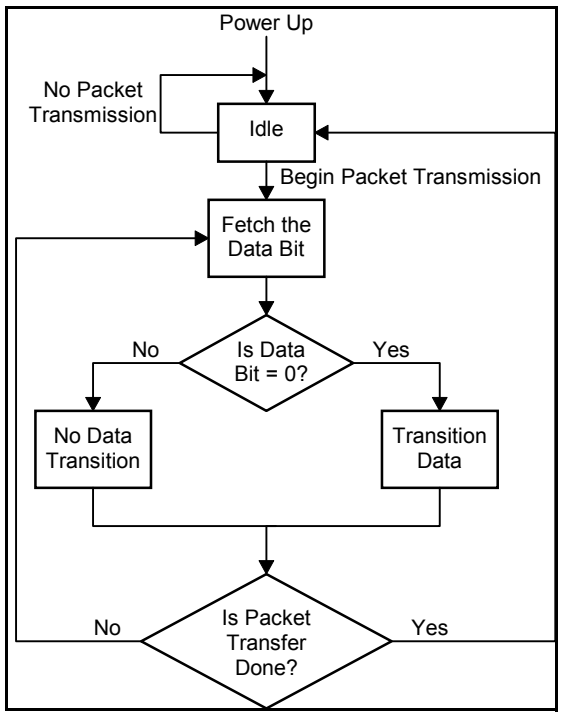


Figure 21-10. Flow Diagram for NRZI

21.7.2 Bit Stuffing

In order to ensure adequate signal transitions, bit stuffing is employed by the transmitting device when sending a packet on the USB (see Figure 21-11 and Figure 21-12). A 0 is inserted after every six consecutive 1's in the data stream before the data is NRZI encoded to force a transition in the NRZI data stream. This gives the receiver logic a data transition at least once every seven bit times to guarantee the data and clock lock. The receiver must decode the NRZI data, recognize the stuffed bits, and discard them. Bit stuffing is enabled beginning with the Sync Pattern and throughout the entire transmission. The data "one" that ends the Sync Pattern is counted as the first one in a sequence. Bit stuffing is always enforced, without exception. If required by the bit stuffing rules, a zero bit will be inserted even if it is the last bit before the end-of-packet (EOP) signal.

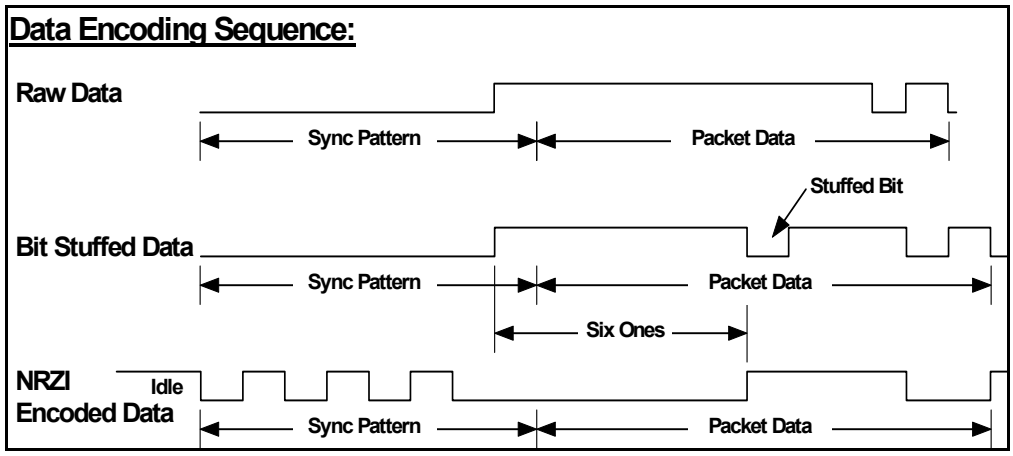


Figure 21-11. Bit Stuffing

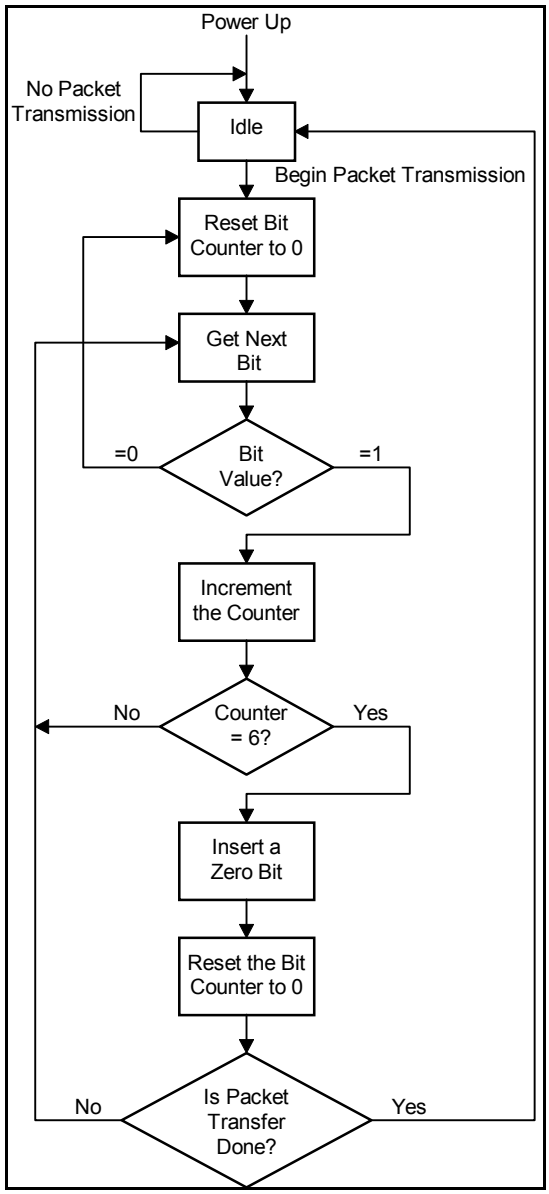


Figure 21-12. Flow Diagram for Bit Stuffing

21.7.3 Reset Signaling

The USB employs a signaling condition that can reset the entire bus to a given state. A reset is signaled on the bus by the presence of an extended SE0 at the USB data pins of a device. The reset signaling is specified to be present for a minimum of 10 ms. An active device (powered and not in the suspend state) seeing a single-ended zero on its USB data inputs for more than 2.5µs may treat that signal as a reset, but must have interpreted the signaling as a reset within 5.5 µs. For a Full speed device, this translates to between 32 and 64 full speed bit times for a valid USB reset. For a Low speed device, an SE0 condition for between 4 and 8 low speed bit times represents a valid USB reset. After the reset is removed, the device will be in the attached, but not yet addressed or configured state (refer to Section 21.1, “Description.”). The device must be able to accept device address via a SET_ADDRESS command (refer to Section 21.7, “Signaling Description,” on page 21-16 of the USB spec) no later than 10 ms after the reset is removed.

The default configuration of the USB model allows the USB generated reset to set a status bit to note that a RESET has occurred on the USB bus The firmware must set the RST_en bit in the Interrupt enable register (see Table 21-7, USBIER Description) to see this interrupt.

Reset can wake a device from the suspended mode. A device may take up to 10 ms to wake up from the suspended state.

21.7.4 Suspend / Resume

USB employs a low power mode termed suspend. Suspend mode is entered when the USB data lines are in the idle state for more than 4.0 ms. Any bus activity will keep the device out of the suspend state. The SOF packet is guaranteed to occur once a frame (1 frame/ms) to keep full speed devices awake during normal bus operation. Low speed devices are kept awake by a low speed EOP since they do not receive the SOF packet (refer to Section 21.7.5.1, “Low Speed Device.”). This is referred to as Low speed keep alive.

The USB model can be awakened from the suspend state by switching the bus state to the resume state (“K” state), by normal bus activity, or by signaling a reset. The USB model also supports the remote wakeup feature. This gives the firmware the ability to exit suspend mode by signaling a resume state to the upstream Host or Hub. Refer to the register definitions (see Section 21.10.1) specifically the Force Resume (FRESUM) bit for more information about how to initiate the remote wakeup feature.

When using the remote wake-up capability, the firmware must wait for at least 5 ms after the bus is in the idle state before sending the remote wake-up resume signaling. This allows the upstream devices to get into their suspend state and prepare for propagating resume signaling. The resume signaling will be held by the hardware for at least 10 ms and no more than 15 ms. At the end of the resume signaling, the data lines will return to their high impedance state.

21.7.5 Supported USB Device Speeds

USB devices can operate at one of three signaling frequencies. The low speed signaling frequency is 1.5MHz and the full speed signalling frequency is 12Mhz. The high speed signaling frequency is not supported in the Neptune LTE configuration.

The default USB signaling frequency is determinted by the fsen port of the USB module. The USB speed is configurable by setting the SPEEDSEL bits in the USBCPUCTL register (refer to Table 21-26, “USBCPUCTL Description” for the setting of the SPEEDSEL bits).

NOTE:

Only change the SPEEDSEL bits when the USB clock is disabled and the USB is in idle or reset state.

21.7.5.1 Low Speed Device

The low speed data rate is nominally 1.5 Mbs. The permitted frequency tolerance for low speed functions is approximately ±1.5% (15000 ppm). This tolerance includes inaccuracies from all sources: initial frequency accuracy, crystal capacitive loading, supply voltage on the oscillator, temperature, and aging. The jitter in the low speed data rate must be less than 10ns. This tolerance allows the use of resonators in low cost, low speed devices.

For low speed transmissions, the transmitter’s EOP width must be between 1.25µs and 1.50µs. These ranges include timing variations due to differential buffer delay and rise/fall time mismatches and to noise and other random effects. A low speed receiver must accept a 670 ns wide SE0 followed by a J transition

as a valid EOP. An SE0 narrower than 330 ns or an SE0 not followed by a J transition must be rejected as an EOP. An EOP between 330 ns and 670 ns may be rejected or accepted as above. Any SE0 that is 2.5µs or wider is automatically a reset.

21.7.5.2 Full Speed Device

The full speed data rate is nominally 12 Mbs. The data rate tolerance for full speed functions is ±0.25% (2500 ppm). The accuracy of the host controller's data rate must be known to better than ±0.05% (500 ppm) in order to meet the frame interval accuracy. This tolerance includes inaccuracies from all sources: initial frequency accuracy, crystal capacitive loading, supply voltage on the oscillator, temperature, and aging.

For full speed transmissions, the EOP width from the transmitter must be between 160ns and 175ns. These ranges include timing variations due to differential buffer delay and rise/fall time mismatches and to noise and other random effects. The full speed receiver must accept a 82 ns wide SE0 followed by a J transition as a valid EOP. An SE0 narrower than 40ns or any SE0 not followed by a J transition must be rejected as an EOP.

21.7.5.3 High Speed Device

Currently not supported in the Neptune LTE configuration.

21.8 Hardware/Software Interaction

This USB design provides accessibility to a number of registers and buffers that comprise the control and data for the USB device interface. To transmit and receive data properly, certain controls are in place and some guidelines must be followed.

The USB provides two methods of flow control for non-Isochronous endpoints. The first is through interrupt processing. The USB module is interrupt driven. The registers and buffer values for a particular endpoint are subject to change until its interrupt has been set. From this point, the registers and buffer associated with this endpoint cannot change until the interrupt is cleared.

While an interrupt for a given non-isochronous endpoint is set, any further transactions received on the USB from the Host are “nak’ed”. That is, the NAK response is returned as the handshake by USB interface. This provides flow control by ensuring that the processor can access the USB data without concern about over/under-running of data.

There is a second method of flow control implemented on non-isochronous endpoints. This is the XFREN (transfer enable) bit in the endpoint control registers. If this bit is cleared, any transactions received by the USB interface for this endpoint will be “nak’ed”. This allows control over the endpoint so that it is enabled, but transactions are held off until the software would like to start transactions.

Isochronous endpoints are also interrupt driven, but do not provide the same safeguards as the other type of endpoints. However, these endpoints are only updated once every 1 millisecond. As long as the software does not delay in accessing these endpoints upon the reception of the interrupt, there should not be any problems with updates occurring during accesses of the endpoint registers. The data, if double-buffered, normally alternates data buffers so there should not be any overlap in accesses during normal operation.

The SOF register updates fall under the same category as the isochronous endpoints. This register is updated every millisecond and should read before the next update.

21.9 Memory Maps

Shown below is the memory map for the USB device. The access to the USB peripheral is 16 bit. The organization is broken up into two separate groups. One, registers dedicated to a DSP and two, registers dedicated to a CPU. These registers are further broken down into control registers and endpoint registers.

The two processor domains do not share any common registers, nor do they share buffer space. Each will see interrupts for global type USB operations, such as USB reset, Suspend, and Resume. The transactional interrupts will be seen on endpoints that are dedicated to that specific domain. The SOF interrupt is seen only by the DSP since it has the isochronous endpoints. The CPU is responsible for providing control messaging to the DSP so that endpoints are managed appropriately.

The USB design is an interrupt driven device that must be programmed to configure and enable its endpoints. However, the device, after a valid USB reset, will accept the default USB address for the Control endpoint 0, so that communication can be established with the USB host.

This allows the host to go through the process of enumeration which will provide a Function address to the USB device. This value should be placed into the Function address register to establish a valid communication link between the host and the USB device.

An interrupt register and an interrupt enable register for each processor are used to determine which interrupt flags are actually presented.

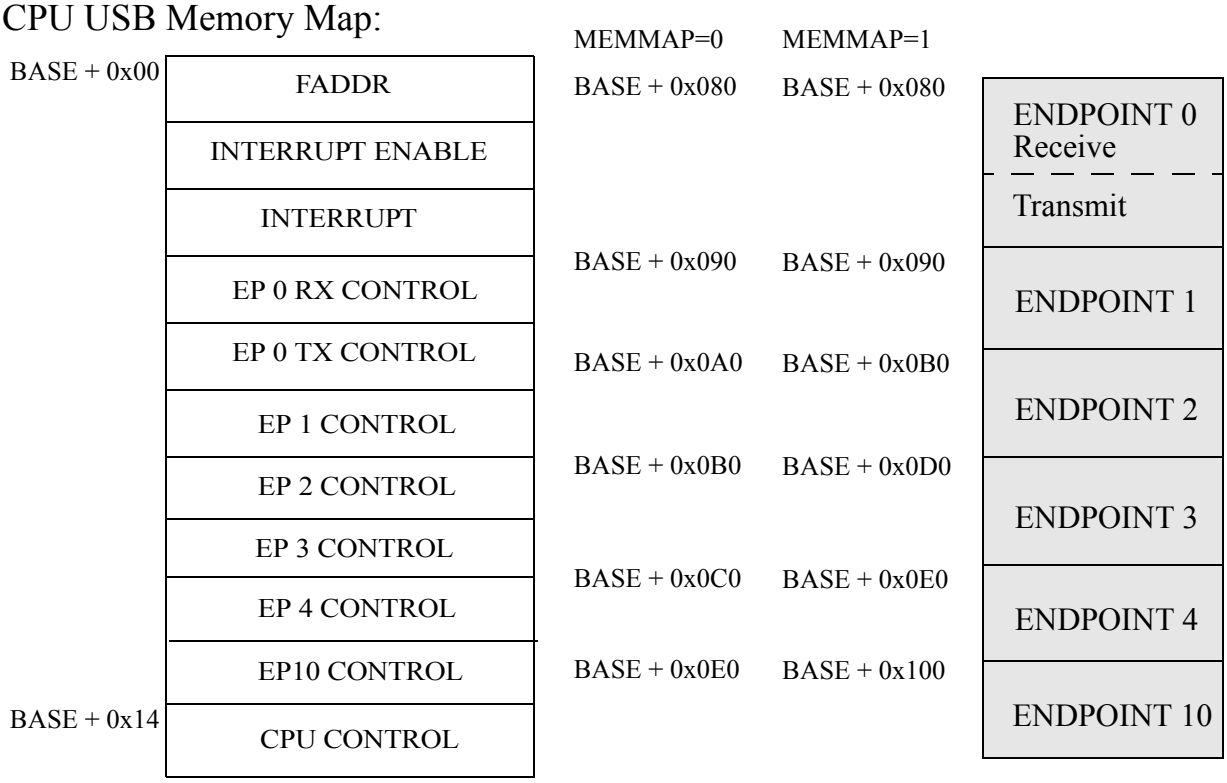
21.9.1 Buffer Connections

The USB peripheral uses a RAM as a buffer for storage of data being received or transmitted on the various endpoints.

There are two logical RAM buffers for the USB module. One is for the endpoints associated with the CPU and contains 136 bytes for the endpoints assigned to the CPU. The other logical RAM is associated with the DSP and contains 120 bytes for buffering data for the DSP endpoints

Physically, the USB module uses four custom RAM blocks. There are two dedicated to the CPU and two dedicated to the DSP. For each pair, there is one buffer for receive endpoints and one for transmit endpoints.

CPU USB Memory Map:



DSP USB Memory Map:

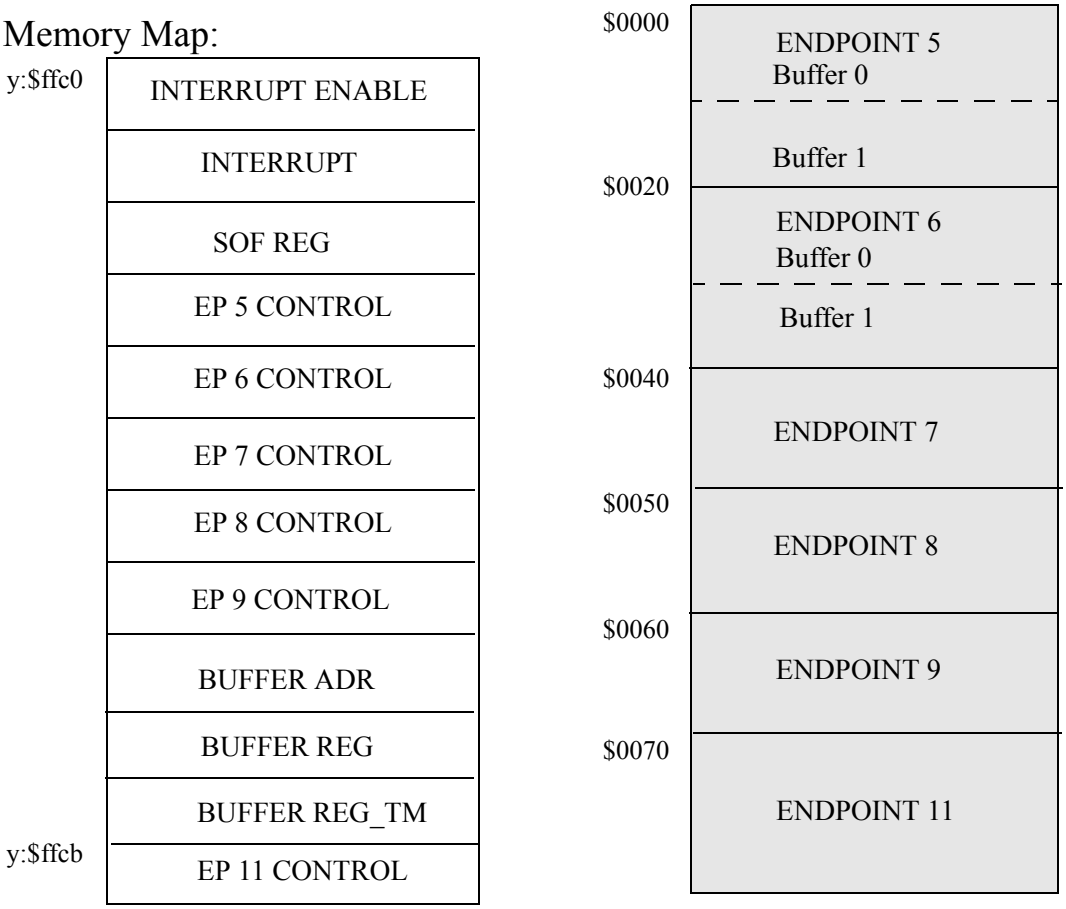


Figure 21-13. USB Memory Maps (With CPU Memory Map Selection)

21.10 USB Control Registers

Figure 21-14. Register Summary Legend

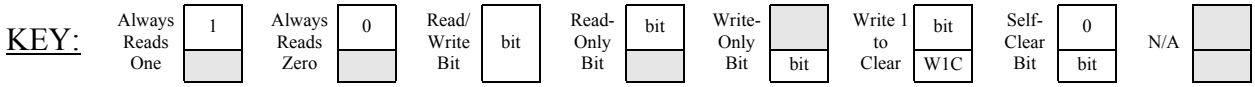


Table 21-4. USB CPU Control Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBFAD ($2485_2000)	R	0	0	0	0	0	0	0	0	0	FAD6	FAD5	FAD4	FAD3	FAD2	FAD1	FAD0
	W																
USBIER ($2485_2002)	R	FRESUM	RSRFC	ERRC	ERR_en	0	0	EP10_en	EP4_en	EP3_en	EP2_en	EP1_en	EP0_TX_en	EP0_RX_en	0	SR_en	RST_en
	W																
USBINT ($2485_2004)	R	ERR	0	0	0	0	0	EP10_int	EP4_int	EP3_int	EP2_int	EP1_int	EP0_TX_int	EP0_RX_int	SUSP_int	RES_int	RST_int
	W																
USBRXCR ($2485_2006)	R	EPTYP1	EPTYP0	0	0	FSTALL	XFREN	EPEN	SEQ	OVF	ERRF	SETUPF	0	B3	B2	B1	B0
	W			IEOSFC	SETSEQ												
USBTXCR ($2485_2008)	R	EPTYP1	EPTYP0	0	0	FSTALL	XFREN	EPEN	SEQ	ACKF	ERRF	0	0	B3	B2	B1	B0
	W			IAEFC	SETSEQ												
USBE1CR ($2485_200A)	R	EPTYP1	EPTYP0	0	0	FSTALL	XFREN	EPEN	SEQ	OVF	ERRF	B5	B4	B3	B2	B1	B0
	W			IEOFC	SET												
USBE2CR ($2485_200C)	R	EPTYP1	EPTYP0	0	0	FSTALL	XFREN	EPEN	SEQ	ACKF	ERRF	B5	B4	B3	B2	B1	B0
	W			IAEFC	SET												
USBE3CR ($2485_200E)	R	EPTYP1	EPTYP0	0	0	FSTALL	XFREN	EPEN	SEQ	ACKF	ERRF	0	B4	B3	B2	B1	B0
	W			IEOFC	SET												
USBE4CR ($2485_2010)	R	EPTYP1	EPTYP0	0	0	FSTALL	XFREN	EPEN	SEQ	ACKF	ERRF	B5	B4	B3	B2	B1	B0
	W			IAEFC	SET												
USBE10CR ($2485_2012)	R	EPTYP1	EPTYP0	0	0	FSTALL	XFREN	EPEN	SEQ	ACKF	ERRF	0	0	B3	B2	B1	B0
	W			IAEFC	SET												
USBCPUCR ($2485_2014)	R	0	0	0	0	0	0	0	0	0	0	SPEEDSEL		0	0	0	MEMMAP
	W																

Table 21-5. USB DSP Control Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBINTE (Y:$FFC0)	R	0	0	SOF_en	EP9_en	EP8_en	EP7_en	EP6_en	EP5_en	EP11_en	0	0	0	0	0	SR_en	RST_en
	W		RSRFC														
USBDIR (Y:$FFC1)	R	0	0	SOF_int	EP9_int	EP8_int	EP7_int	EP6_int	EP5_int	EP11_int	0	0	0	0	SUSP_int	RES_int	RST_int
	W																
USBSOF (Y:$FFC2)	R	0	0	ACKF	0	FLOCK	FN10	FN9	FN8	FN7	FN6	FN5	FN4	FN3	FN2	FN1	FN0
	W				RSRFC												
USBE5CR (Y:$FFC3)	R	EPTYP1	EPTYP0	0	DBUF	DPM	0	EPEN	0	OVF	ERRF	0	B4	B3	B2	B1	B0
	W			IEOFC													
USBE6CR (Y:$FFC4)	R	EPTYP1	EPTYP0	0	DBUF	DPM	0	EPEN	0	0	0	0	B4	B3	B2	B1	B0
	W			IAFC													
USBE7CR (Y:$FFC5)	R	EPTYP1	EPTYP0	0	0	FSTALL	XFREN	EPEN	SEQ	OVF	ERRF	0	B4	B3	B2	B1	B0
	W			IEOFC	SET												
USBE8CR (Y:$FFC6)	R	EPTYP1	EPTYP0	0	0	FSTALL	XFREN	EPEN	SEQ	ACKF	ERRF	0	B4	B3	B2	B1	B0
	W			IAEFC	SET												
USBE9CR (Y:$FFC7)	R	EPTYP1	EPTYP0	0	0	FSTALL	XFREN	EPEN	SEQ	ACKF	ERRF	0	B4	B3	B2	B1	B0
	W			IAEFC	SETSEQ												
USBBPR (Y:$FFC8)	R	0	0	0	0	0	0	0	0	PTR7	PTR6	PTR5	PTR4	PTR3	PTR2	PTR1	PTR0
	W																
USBDBR (Y:$FFC9)	R	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
	W																
USBDRAT (Y:$FFCA)	R	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
	W																
USBE11CR (Y:$FFCB)	R	EPTYP1	EPTYP0	0	0	FSTALL	XFREN	EPEN	SEQ	ACKF	ERRF	0	0	B3	B2	B1	B0
	W			IAEFC	SETSEQ												

21.10.1 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various USB CPU Control Registers. The following definitions serve as a key for these figures:

- Grey bit: unimplemented bit. Always reads as zero; Writing has no effect.
- TYPE: Type of register bit. Defines register bit's behavior. Possible values:
  - r: read only. Writing this bit has no effect.
  - w: write only.
  - rw: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - rwm: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - w1c: A status bit that can be read, and it cleared by writing a logic 1.
  - slfclr: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- RESET: Gives the reset value of the bit. Possible values:
  - 0: Will reset to a logic 0
  - 1: Will reset to a logic 1
  - ?: The reset state is unknown.
  - u: Unaffected by reset

USBFAD USB Function Address Register **Addr $2485_2000**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
										FAD6	FAD5	FAD4	FAD3	FAD2	FAD1	FAD0
TYPE	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-6. USBFAD Description

Name	Description	Settings
Bits 15–7	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A
FAD [6:0] Bits 6–0	**Function Address**—This16-bit read/write register specifies the USB address of the device. Reset clears these bits.	N/A

USBIER USB Interrupt Enable Register **Addr $2485_2002**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	FRESUM	RSRFC	ERRC	ERR_en	XVR type		EP10_en	EP4_en	EP3_en	EP2_en	EP1_en	EP0_TX_en	EP0_RX_en		SR_en	RST_en
TYPE	slfclr	slfclr	slfclr	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	r	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-7. USBIER Description

Name	Description	Settings
FRESUM Bit 15	**Force Resume**—This bit forces a resume state (“K” or non-idle state) onto the USB data lines to initiate a remote wakeup. This bit will always read as a “0”. Hardware controls the duration of the forced resume to be greater than 1ms.	1 = Force data lines to “K” state 0 = Default
RSRFC Bit 14	**Resume/Suspend/Reset Interrupt Flag Clear**—This bit clears the RES_int, SUSP_int, and RST_int flags in the Interrupt enable register. This bit always reads as zero.	1 = Clear Resume/Suspend, USB reset interrupts 0 = Default
ERRC Bit 13	**Halfword Access Error Flag Clear**—This bit clears the ERR flag in the Interrupt register. This bit always reads as zero.	1 = Clear ERR flag 0 = Default
ERR_en Bit 12	**Write Error Interrupt Enable**—This bit controls interrupt generation caused by byte write accesses.	1 = Write Error interrupt enable 0 = Write Error interrupt disabled
XVR type Bit 11	**Transceiver Type** —This bit controls the interface type for the transceiver.	1 = Differential transmit 0 = Single ended transmit (default)
Bits 10	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A
EP10_en Bit 8	**Endpoint 10 Interrupt Enable**—This bit enables an interrupt to be generated whenever a transaction on this endpoint is completed.	1 = EP10 transaction interrupt enabled 0 = EP10 transaction interrupt disabled
EP4_en Bit 8	**Endpoint 4 Interrupt Enable**—This bit enables an interrupt to be generated whenever a transaction on this endpoint is completed.	1 = EP4 transaction interrupt enabled 0 = EP4 transaction interrupt disabled
EP3_en Bit 7	**Endpoint 3 Interrupt Enable**—This bit enables an interrupt to be generated whenever a transaction on this endpoint is completed.	1 = EP3 transaction interrupt enabled 0 = EP3 transaction interrupt disabled
EP2_en Bit 6	**Endpoint 2 Interrupt Enable**—This bit enables an interrupt to be generated whenever a transaction on this endpoint is completed.	1 = EP2 transaction interrupt enabled 0 = EP2 transaction interrupt disabled
EP1_en Bit 5	**Endpoint 1 Interrupt Enable**—This bit enables an interrupt to be generated whenever a transaction on this endpoint is completed.	1 = EP1 transaction interrupt enabled 0 = EP1 transaction interrupt disabled

Table 21-7. USBIER Description (Continued)

Name	Description	Settings
EP0_TX_en Bit 4	**Endpoint 0 Transmit Interrupt Enable**—This bit enables an interrupt to be generated whenever a transaction on this endpoint is completed.	1 = EP0 Transmit transaction interrupt enabled 0 = EP0 Transmit transaction interrupt disabled
EP0_RX_en Bit 3	**Endpoint 0 Receive Interrupt Enable**—This bit enables an interrupt to be generated whenever a transaction on this endpoint is completed.	1 = EP0 Receive transaction interrupt enabled 0 = EP0 Receive transaction interrupt disabled
Bit 2	Reserved bit. Read as zero. Must be written with zero for future compatibility.	N/A
SR_en Bit 1	**Resume / Suspend Interrupt Enable**—This read/write bit enables an interrupt to occur when either RSM_IF or SUSP_IF flag is set. Reset clears this bit.	1 = Resume / Suspend interrupt enabled 0 = Resume / Suspend interrupt disabled
RST_en Bit 0	**USB Reset Interrupt Enable**—This read/write bit enables an interrupt to occur when the RST_IF bit is set. Reset clears this bit.	1 = USB Reset interrupt enabled 0 = USB Reset interrupt disabled

USBINT USB CPU Interrupt Register **Addr $2485_2004**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	ERR						EP10_int	EP4_int	EP3_int	EP2_int	EP1_int	EP0_TX_int	EP0_RX_int	SUSP_int	RES_int	RST_int
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-8. USBINT Description

Name	Description	Settings
ERR Bit 15	**Error Flag**—This bit signifies that a non-halfword access occurred from the CPU. This bit will be cleared by a write to the ERRC bit in Interrupt Enable register. Reset also clears this bit.	1 = Non-halfword access occurred. 0 = Default
Bits 14–10	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A
EP10_int Bit 9	**Endpoint 10 Interrupt Flag**—This bit signifies that a transaction has completed on Endpoint 10. This will generate an interrupt if the corresponding EP10 interrupt enable bit is set in the IE register. This bit will be cleared by a write to the IAEFC bit in the EP10 Status/Ctl register. Reset also clears this bit.	1 = EP10 transaction done. 0 = Default
EP4_int Bit 8	**Endpoint 4 Interrupt Flag**—This bit signifies that a transaction has completed on Endpoint 4. This will generate an interrupt if the corresponding EP4 interrupt enable bit is set in the IE register. This bit will be cleared by a write to the IAEFC bit in the EP4 Status/Ctl register. Reset also clears this bit.	1 = EP4 transaction done. 0 = Default
EP3_int Bit 7	**Endpoint 3 Interrupt Flag**—This bit signifies that a transaction has completed on Endpoint 3. This will generate an interrupt if the corresponding EP3 interrupt enable bit is set in the IE register. This bit will be cleared by a write to the IAEFC bit in the EP3 Status/Ctl register. Reset also clears this bit.	1 = EP3 transaction done. 0 = Default
EP2_int Bit 6	**Endpoint 2 Interrupt Flag**—This bit signifies that a transaction has completed on Endpoint 2. This will generate an interrupt if the corresponding EP2 interrupt enable bit is set in the IE register. This bit will be cleared by a write to the IAEFC bit in the EP2 Status/Ctl register. Reset also clears this bit.	1 = EP2 transaction done. 0 = Default

Table 21-8. USBINT Description (Continued)

Name	Description	Settings
EP1_int Bit 5	**Endpoint 1 Interrupt Flag**—This bit signifies that a transaction has completed on Endpoint 1. This will generate an interrupt if the corresponding EP1 interrupt enable bit is set in the IE register. This bit will be cleared by a write to the IEOFC bit in the EP1 Status/Ctl register. Reset also clears this bit.	1 = EP1 transaction done. 0 = Default
EP0_TX_int Bit 4	**Endpoint 0 Transmit Interrupt Flag**—This bit signifies that a transmit transaction has completed on Endpoint 0. This will generate an interrupt if the corresponding EP0 interrupt enable bit is set in the IE register. This bit will be cleared by a write to the IAEFC bit in the EP0 Status/Ctl register. Reset also clears this bit.	1 = EP0 transaction done. 0 = Default
EP0_RX_int Bit 3	**Endpoint 0 Receive Interrupt Flag**—This bit signifies that a receive transaction has completed on Endpoint 0. This will generate an interrupt if the corresponding EP0 interrupt enable bit is set in the IE register. This bit will be cleared by a write to the IEOSFC bit in the EP0 Status/Ctl register. Reset also clears this bit.	1 = EP0 transaction done. 0 = Default
SUSP_int Bit 2	**Suspend Interrupt Flag**—This bit is set when the USB SIE detects a suspend state on the USB data lines. This bit will be clear by a write to the RSFRC bit in the IE register. Reset clears this bit.	1 = Suspend detected 0 = Default
RES_int Bit 1	**Resume Interrupt Flag**—This bit is set when the USB SIE detects a resume condition on the USB data lines (J to K transition). This bit will be clear by a write to the RSFRC bit in the IE register. Reset clears this bit.	1 = Resume detected 0 = default
RST_int Bit 0	**Reset Interrupt Flag**—This bit is set when the USB SIE detects a reset condition on the USB data lines (SE0 > 2.5 us). This bit will be clear by a write to the RSFRC bit in the IE register. Reset clears this bit.	1 = USB Reset detected 0 = default

USBRXCR USB Endpoint 0 RX Control Register **Addr $2485_2006**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	EPTYP 1	EPTYP 0	IEO SFC	SET SEQ	FST ALL	XFRE N	EPEN	SEQ	OVF	ERRF	SETUP F		B3	B2	B1	B0
TYPE	r	r	slfclr	slfclr	rw	rw	rw	rw	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-9. USBRXCR Description

Name	Description	Settings
EPTYP [1:0] Bits 15–14	**Endpoint Type**—These read only bits will always read as 00b to denote a Control Endpoint.	<table><tr><th>EPTYP1</th><th>EPTYP0</th><th>Endpoint Type</th></tr><tr><td>0</td><td>0</td><td>Control</td></tr><tr><td>0</td><td>1</td><td>Reserved</td></tr><tr><td>1</td><td>0</td><td>Reserved</td></tr><tr><td>1</td><td>1</td><td>Reserved</td></tr></table>
IEOSFC Bit 13	**Interrupt, ERR, Overflow, and Setup Flag Clear**—Writing a one to this bit will clear the OVF and ERRF bits The EP0_RX_int bit in the Interrupt register (USBINT) will also be cleared. This bit always reads as a zero. Writing a one to the IEOSFC bit will clear the SETUPF bit.	1 = Not-applicable 0 = Always reads as a zero.
SETSEQ Bit 12	**Set SEQ bit**—This write only bit controls the setting of the SEQ bit. This bit will read as zero when read.	1 = Set SEQ to value from data bus 0 = Do not allow SEQ to be updated from data bus.
FSTALL Bit 11	**Force STALL Response**—This read/write bit causes the endpoint to return a STALL handshake when polled by an OUT token by the USB Host Controller. Control endpoints are not allowed to respond with a STALL handshake to a SETUP token, so this bit will have no effect in that case. A valid SETUP token on an endpoint configured as a control endpoint clears the FSTALL bit in both the transmit and receive EPCTL registers for that endpoint. Reset clears this bit.	1 = Send STALL handshake 0 = Default
XFREN Bit 10	**Transfer Enable** — This read/write bit enables a transfer to occur when the endpoint is polled by the USB Host controller. Reset clears this bit.	1 = Data is ready to be received. 0 = If (EPEN = 1), the endpoint is enabled, but busy. Respond with NAK. (Applies to only data and status phases.)

Table 21-9. USBRXCR Description (Continued)

Name	Description	Settings
EPEN Bit 9	**Endpoint Enable** —This read/write bit enables the endpoint to be seen by the USB Host controller. If this bit is clear, any transactions addressing this endpoint will be ignored. Reset clears this bit for all endpoints.	1 = Endpoint is available to USB Host controller 0 = Endpoint is not available to USB Host controller
SEQ Bit 8	**Sequence**—This bit determines which type of data packet (DATA0 or DATA1) is expected during the next OUT transaction. The toggling of this bit is controlled by hardware (upon reception of a valid ACK handshake), but can be written by firmware to force a particular data token. Reset clears this bit. Since this endpoint is configured as a control endpoint (EPTYP = 00), the reception of a valid SETUP token on the corresponding Receive endpoint will set SEQ to 1.	1 = DATA1 Token active for next transaction 0 = DATA0 Token active for next transaction.
OVF Bit 7	**Receive Buffer Overflow** —This bit is set when a receive endpoint receives more than its maximum number of bytes. The RX0_int will be set on an overflow event. The firmware is responsible for determining whether an overflow event occurred. This bit cleared by a write to the IEOSFC bit. reset clears this bit.	1 = Overflow occurred 0 = No overflow occurred
ERRF Bit 6	**Error flag**—This bit indicates an Error occurred during the last transmit operation. Errors include: bit-stuff, CRC, and time-out. This bit is cleared by a write to the IEOSFC bit. Reset also clears this bit.	1 = Error occurred 0 = No error detected
SETUPF Bit 5	**Setup Token flag**—This bit indicates that a valid SETUP token has been received for this endpoint. This bit is cleared by the IEOSFC bit. Reset also clears this bit.	1 = Setup token detected 0 = Default
Bit 4	Reserved. Reads as zero. Must be written with zero for future compatibility.	N/A
B[3:0] Bits 3–0	**Byte Count**—These bits are read only and reflect the number of bytes received during the last transaction. Reset clears these bits.	N/A

USBTXCR USB Endpoint 0 TX Control Register **Addr $2485_2008**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	EPTYP 1	EPTYP 0	IAEFC	SET SEQ	FST ALL	XFRE N	EPEN	SEQ	ACKF	ERRF			B3	B2	B1	B0
TYPE	r	r	slfclr	slfclr	rw	rw	rw	rw	r	r	r	r	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-10. USBTXCR Description

Name	Description	Settings
EPTYP [1:0] Bits 15–14	**Endpoint Type**—These read-only bits will always read as 00b to denote a Control Endpoint.	<table><tr><th>EPTYP1</th><th>EPTYP0</th><th>Endpoint Type</th></tr><tr><td>0</td><td>0</td><td>Control</td></tr><tr><td>0</td><td>1</td><td>Reserved</td></tr><tr><td>1</td><td>0</td><td>Reserved</td></tr><tr><td>1</td><td>1</td><td>Reserved</td></tr></table>
IAEFC Bit 13	**Interrupt, ACK, and ERR Flag Clear**—Writing a one to this bit will clear the ACKF and ERRF bits The EP0_TX_int bit in the Interrupt register (USBINT) will also be cleared. This bit always reads as a zero.	1 = Not-applicable 0 = Always reads as a zero.
SETSEQ Bit 12	**Set SEQ bit** — This write only bit controls the setting of the SEQ bit. This bit will read as zero when read.	1 = Set SEQ to value from data bus 0 = Do not allow SEQ to be updated from data bus.
FSTALL Bit 11	**Force STALL Response**—This read/write bit causes the endpoint to return a STALL handshake when polled by an IN token by the USB Host Controller. Control endpoints are not allowed to respond with a STALL handshake to a SETUP token, so this bit will have no effect in that case. A valid SETUP token on an endpoint configured as a control endpoint clears the FSTALL bit in both the transmit and receive EPCTL registers for that endpoint. Reset clears this bit.	1 = Send STALL handshake 0 = Default
XFREN Bit 10	**Transfer Enable** — This read/write bit enables a transfer to occur when the endpoint is polled by the USB Host controller. Reset clears this bit.	1 = Data is ready to be sent. 0 = If (EPEN = 1), the endpoint is enabled, but busy. Respond with NAK.
EPEN Bit 9	**Endpoint enabled** —This read/write bit enables the endpoint to be seen by the USB Host controller. If this bit is clear, any transactions addressing this endpoint will be ignored. Reset clears this bit for all endpoints.	1 = Endpoint is available to USB Host controller 0 = Endpoint is not available to USB Host controller

Table 21-10. USBTXCR Description (Continued)

Name	Description	Settings
SEQ Bit 8	**Sequence** — This bit determines which type of data packet (DATA0 or DATA1) will be sent during the next IN transaction. The toggling of this bit is controlled by hardware (upon reception of a valid ACK handshake), but can be written by firmware to force a particular data token. Reset clears this bit.	1 = DATA1 Token active for next transaction 0 = DATA0 Token active for next transaction.
ACKF Bit 7	**Acknowledge flag**—This bit is set when a valid ACK handshake is received after the USB device endpoint data is successfully transmitted over the USB lines. This bit is cleared by a write of the IAEFC bit. Reset also clears this bit.	1 = ACK handshake received 0 = No ACK received
ERRF Bit 6	**Error flag**—This bit indicates an Error occurred during the last transmit operation. Errors include: bit-stuff, CRC, and time-out. This bit is cleared by a write to the IAEFC bit. Reset also clears this bit.	1 = Error occurred 0 = No error detected
Bits 5–4	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A
B[3:0] Bits 3–0	**Byte Count**—These bits are read/writable and contain the number of bytes desired to be transferred at the next IN token request. Reset clears these bits.	N/A

USBE1CR Endpoint 1 Control Register **Addr $2485_200A**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	EPTYP 1	EPTYP 0	IEOFC	SET	FSTALL	XFREN	EPEN	SEQ	OVF	ERRF	B5	B4	B3	B2	B1	B0
TYPE	rw	r	slfclr	slfclr	rw	rw	rw	rw	r	r	r	r	r	r	r	r
RESET	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-11. USBE1CR Description

Name	Description	Settings
EPTYP [1:0] Bits 15-14	**Endpoint Type**—These bits describe the type of endpoint that is programmed. This endpoint can be either a Bulk or Interrupt RX endpoint. So only values of 01b or 11b are valid. Therefore bit EPTYP1 is a read/write bit and bit EPTYP0 is read-only and always read as one.	**EPTYP1** / **EPTYP0** / **Endpoint Type**: 0 / 0 / Reserved; 0 / 1 / Bulk; 1 / 0 / Reserved; 1 / 1 / Interrupt RX
IEOFC Bit 13	**Interrupt, ERR, and Overflow Flag Clear**—Writing a one to this bit will clear the OVF and ERRF bits The EP1_int bit in the Interrupt register (USBINT) will also be cleared. This bit always reads as a zero.	1 = Not-applicable 0 = Always reads as a zero.
SET Bit 12	**Set SEQ and EPTYP1 bit**—This write only bit controls the setting of the SEQ and EPTYP1 bits. This bit will read as zero when read.	1 = Set SEQ and EPTYP1 to values from data bus 0 = Do not allow SEQ or EPTYP1 to be updated from data bus.
FSTALL Bit 11	**Force STALL Response**—This read/write bit causes the endpoint to return a STALL handshake when polled by an OUT token by the USB Host Controller. Reset clears this bit.	1 = Send STALL handshake 0 = Default
XFREN Bit 10	**Transfer Enable**—This read/write bit enables a transfer to occur when the endpoint is polled by the USB Host controller. Reset clears this bit.	1 = Data is ready to be received. 0 = If (EPEN = 1), the endpoint is enabled, but busy. Respond with NAK.
EPEN Bit 9	**Endpoint enabled**—This read/write bit enables the endpoint to be seen by the USB Host controller. If this bit is clear, any transactions addressing this endpoint will be ignored. Reset clears this bit for all endpoints.	1 = Endpoint is available to USB Host controller 0 = Endpoint is not available to USB Host controller

Table 21-11. USBE1CR Description (Continued)

Name	Description	Settings
SEQ Bit 8	**Sequence**—This bit determines which type of data packet (DATA0 or DATA1) is expected during the next OUT transaction. The toggling of this bit is controlled by hardware (upon reception of a valid ACK handshake), but can be written by firmware to force a particular data token. Reset clears this bit.	1 = DATA1 Token active for next transaction 0 = DATA0 Token active for next transaction.
OVF Bit 7	**Receive Buffer Overflow**—This bit is set when a receive endpoint receives more than its maximum number of bytes. The EP1_int flag will be set on an overflow event. The firmware is responsible for determining whether an overflow event occurred. This bit cleared by a write to the IEOFC bit. reset clears this bit.	1 = Overflow occurred 0 = No overflow occurred
ERRF Bit 6	**Error flag**—This bit indicates an Error occurred during the last receive operation. Errors include: bit-stuff, time-out, and CRC. This bit is cleared by a write to the IEOFC bit. Reset also clears this bit.	1 = Error occurred 0 = No error detected
B[5:0] Bits 5–0	**Byte Count**—These bits are read only and reflect the number of bytes received during the last transaction. Reset clears these bits.	N/A

USBE2CR USB Endpoint 2 Control Register **Addr $2485_200C**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	EPTYP1	EPTYP0	IAEFC	SET	FSTALL	XFREN	EPEN	SEQ	ACKF	ERRF	B5	B4	B3	B2	B1	B0
TYPE	rw	r	slfclr	slfclr	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw
RESET	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-12. USBE2CR Description

Name	Description	Settings
EPTYP [1:0} Bits 15–14	**Endpoint Type**—These bits describe the type of endpoint that is programmed. This endpoint can be either a Bulk or Interrupt TX endpoint. So only values of 01b or 11b are valid. Therefore bit EPTYP1 is a read/write bit and bit EPTYP0 is read-only and always read as one.	**EPTYP1** / **EPTYP0** / **Endpoint Type**: 0 / 0 / Reserved; 0 / 1 / Bulk; 1 / 0 / Reserved; 1 / 1 / Interrupt TX
IAEFC Bit 13	**Interrupt, ACK, and ERR Flag Clear**—Writing a one to this bit will clear the ACKF and ERRF bits The EP2_int bit in the Interrupt register (USBINT) will also be cleared. This bit always reads as a zero.	1 = Not-applicable 0 = Always reads as a zero.
SET Bit 12	**Set SEQ and EPTYP1 bit**—This write only bit controls the setting of the SEQ and the EPTYP1 bits. This bit will read as zero when read.	1 = Set SEQ and EPTYP1 to values from data bus 0 = Do not allow SEQ or EPTYP1 to be updated from data bus.
FSTALL Bit 11	**Force STALL Response**—This read/write bit causes the endpoint to return a STALL handshake when polled by an IN token by the USB Host Controller. Reset clears this bit.	1 = Send STALL handshake 0 = Default
XFREN Bit 10	**Transfer Enable**—This read/write bit enables a transfer to occur when the endpoint is polled by the USB Host controller. Reset clears this bit.	1 = Data is ready to be sent. 0 = If (EPEN = 1), the endpoint is enabled, but busy. Respond with NAK.
EPEN Bit 9	**Endpoint enabled**—This read/write bit enables the endpoint to be seen by the USB Host controller. If this bit is clear, any transactions addressing this endpoint will be ignored. Reset clears this bit for all endpoints.	1 = Endpoint is available to USB Host controller 0 = Endpoint is not available to USB Host controller

Table 21-12. USBE2CR Description (Continued)

Name	Description	Settings
SEQ Bit 8	**Sequence**—This bit determines which type of data packet (DATA0 or DATA1) will be sent during the next IN transaction. The toggling of this bit is controlled by hardware (upon reception of a valid ACK handshake), but can be written by firmware to force a particular data token. Reset clears this bit.	1 = DATA1 Token active for next transaction 0 = DATA0 Token active for next transaction.
ACKF Bit 7	**Acknowledge flag**—This bit is set when a valid ACK handshake is received after the USB device endpoint data is successfully transmitted over the USB lines. This bit is cleared by a write of the IAEFC bit. Reset also clears this bit.	1 = ACK handshake received 0 = No ACK received
ERRF Bit 6	**Error flag**—This bit indicates an Error occurred during the last transmit operation. Errors include: bit-stuff, time-out, and CRC. This bit is cleared by a write to the IAEFC bit. Reset also clears this bit.	1 = Error occurred 0 = No error detected
B[5:0] Bits 5–0	**Byte Count**—These bits are read/writable and contain the number of bytes desired to be transferred at the next IN token request. Reset clears these bits.	N/A

USBE3CR USB Endpoint 3 Control Register **Addr $2485_200E**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	EPTYP1	EPTYP0	IEOFC	SET	FSTALL	XFREN	EPEN	SEQ	OVF	ERRF		B4	B3	B2	B1	B0
TYPE	rw	r	slfclr	sflclr	rw	rw	rw	rw	r	r	r	r	r	r	r	r
RESET	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-13. USBE3CR Description

Name	Description	Settings
EPTYP [1:0] Bits 15–14	**Endpoint Type**—These bits describe the type of endpoint that is programmed. This endpoint can be either a Bulk or Interrupt RX endpoint. So only values of 01b or 11b are valid. Therefore bit EPTYP1 is a read/write bit and bit EPTYP0 is read-only and always read as one.	**EPTYP1** / **EPTYP0** / **Endpoint Type**: 0 / 0 / Reserved; 0 / 1 / Bulk; 1 / 0 / Reserved; 1 / 1 / Interrupt RX
IEOFC Bit 13	**Interrupt, ERR, and Overflow Flag Clear**—Writing a one to this bit will clear the ACKF and ERRF bits The EP3_int bit in the Interrupt register (USBINT) will also be cleared. This bit always reads as a zero.	1 = Not-applicable 0 = Always reads as a zero.
SET Bit 12	**Set SEQ and EPTYP1 bit**—This write only bit controls the setting of the SEQ and EPTYP1 bits. This bit will read as zero when read.	1 = Set SEQ and EPTYP1 to values from data bus 0 = Do not allow SEQ or EPTYP1 to be updated from data bus.
FSTALL Bit 11	**Force STALL Response**—This read/write bit causes the endpoint to return a STALL handshake when polled by an OUT token by the USB Host Controller. Reset clears this bit.	1 = Send STALL handshake 0 = Default
XFREN Bit 10	**Transfer Enable**—This read/write bit enables a transfer to occur when the endpoint is polled by the USB Host controller. Reset clears this bit.	1 = Data is ready to be sent. 0 = If (EPEN = 1), the endpoint is enabled, but busy. Respond with NAK.
EPEN Bit 9	**Endpoint enabled**—This read/write bit enables the endpoint to be seen by the USB Host controller. If this bit is clear, any transactions addressing this endpoint will be ignored. Reset clears this bit for all endpoints.	1 = Endpoint is available to USB Host controller 0 = Endpoint is not available to USB Host controller

Table 21-13. USBE3CR Description (Continued)

Name	Description	Settings
SEQ Bit 8	**Sequence**—This bit determines which type of data packet (DATA0 or DATA1) will be sent during the next IN transaction. The toggling of this bit is controlled by hardware (upon reception of a valid ACK handshake), but can be written by firmware to force a particular data token. Reset clears this bit.	1 = DATA1 Token active for next transaction 0 = DATA0 Token active for next transaction.
OVF Bit 7	**Receive Buffer Overflow**—This bit is set when a receive endpoint receives more than its maximum number of bytes. The EP3_int flag will be set on an overflow event. The firmware is responsible for determining whether an overflow event occurred. This bit cleared by a write to the IEOFC bit. reset clears this bit.	1 = Overflow occurred 0 = No overflow occurred
ERRF Bit 6	**Error flag**—This bit indicates an Error occurred during the last receive operation. Errors include: bit-stuff, time-out, and CRC. This bit is cleared by a write to the IEOFC bit. Reset also clears this bit.	1 = Error occurred 0 = No error detected
Bit 5	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A
B[4:0] Bits 4–0	**Byte Count**—These bits are read only and reflect the number of bytes received during the last transaction. Reset clears these bits.	N/A

USBE4CR USB Endpoint 4 Control Register **Addr $2485_2010**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	EPTYP1	EPTYP0	IAEFC	SET	FSTALL	XFREN	EPEN	SEQ	ACKF	ERRF	B5	B4	B3	B2	B1	B0
TYPE	rw	r	slfclr	slfclr	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw
RESET	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-14. USBE4CR Description

Name	Description	Settings
EPTYP [1:0] Bits 15–14	**Endpoint Type**—These bits describe the type of endpoint that is programmed. This endpoint can be either a Bulk or Interrupt TX endpoint. So only values of 01b or 11b are valid. Therefore bit EPTYP1 is a read/write bit and bit EPTYP0 is read-only and always read as one.	<table><tr><th>EPTYP1</th><th>EPTYP0</th><th>Endpoint Type</th></tr><tr><td>0</td><td>0</td><td>Reserved</td></tr><tr><td>0</td><td>1</td><td>Bulk</td></tr><tr><td>1</td><td>0</td><td>Reserved</td></tr><tr><td>1</td><td>1</td><td>Interrupt TX</td></tr></table>
IAEFC Bit 13	**Interrupt, ACK, and ERR Flag Clear**—Writing a one to this bit will clear the ACKF and ERRF bits The EP4_int bit in the Interrupt register (USBINT) will also be cleared. This bit always reads as a zero.	1 = Not-applicable 0 = Always reads as a zero.
SET Bit 12	**Set SEQ and EPTYP1 bit**—This write only bit controls the setting of the SEQ and EPTYP1 bits. This bit will read as zero when read.	1 = Set SEQ and EPTYP1 to values from data bus 0 = Do not allow SEQ or EPTYP1 to be updated from data bus.
FSTALL Bit 11	**Force STALL Response**—This read/write bit causes the endpoint to return a STALL handshake when polled by an IN token by the USB Host Controller. Reset clears this bit.	1 = Send STALL handshake 0 = Default
XFREN Bit 10	**Transfer Enable**—This read/write bit enables a transfer to occur when the endpoint is polled by the USB Host controller. Reset clears this bit.	1 = Data is ready to be sent. 0 = If (EPEN = 1), the endpoint is enabled, but busy. Respond with NAK.
EPEN Bit 9	**Endpoint enabled**—This read/write bit enables the endpoint to be seen by the USB Host controller. If this bit is clear, any transactions addressing this endpoint will be ignored. Reset clears this bit for all endpoints.	1 = Endpoint is available to USB Host controller 0 = Endpoint is not available to USB Host controller

Table 21-14. USBE4CR Description (Continued)

Name	Description	Settings
SEQ Bit 8	**Sequence**—This bit determines which type of data packet (DATA0 or DATA1) will be sent during the next IN transaction. The toggling of this bit is controlled by hardware (upon reception of a valid ACK handshake), but can be written by firmware to force a particular data token. Reset clears this bit.	1 = DATA1 Token active for next transaction 0 = DATA0 Token active for next transaction.
ACKF Bit 7	**Acknowledge flag**—This bit is set when a valid ACK handshake is received after the USB device endpoint data is successfully transmitted over the USB lines. This bit is cleared by a write of the IAEFC bit. Reset also clears this bit.	1 = ACK handshake received 0 = No ACK received
ERRF Bit 6	**Error flag**—This bit indicates an Error occurred during the last transmit operation. errors include: bit-stuff, CRC, and time-out. This bit is cleared by a write to the IAEFC bit. Reset also clears this bit.	1 = Error occurred 0 = No error detected
B[5:0] Bits 5–0	**Byte Count**—These bits are read/writable and contain the number of bytes desired to be transferred at the next IN token request. Reset clears these bits.	N/A

USBE10CR USB Endpoint 10 Control Register **Addr $2485_2012**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	EPTYP1	EPTYP0	IAEFC	SET	FSTALL	XFREN	EPEN	SEQ	ACKF	ERRF			B3	B2	B1	B0
TYPE	r	r	slfclr	slfclr	rw	rw	rw	rw	r	r	r	r	rw	rw	rw	rw
RESET	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-15. USBE10CR Description

Name	Description	Settings
EPTYP [1:0] Bits 15–14	**Endpoint Type**—These bits describe the type of endpoint that is programmed. This is an Interrupt TX endpoint.	2'b11 is the fixed value
IAEFC Bit 13	**Interrupt, ACK, and ERR Flag Clear**—Writing a one to this bit will clear the ACKF and ERRF bits The EP10_int bit in the Interrupt register (USBINT) will also be cleared. This bit always reads as a zero.	1 = Not-applicable 0 = Always reads as a zero.
SET Bit 12	**Set SEQ and EPTYP1 bit**—This write only bit controls the setting of the SEQ and EPTYP1 bits. This bit will read as zero when read.	1 = Set SEQ and EPTYP1 to values from data bus 0 = Do not allow SEQ or EPTYP1 to be updated from data bus.
FSTALL Bit 11	**Force STALL Response**—This read/write bit causes the endpoint to return a STALL handshake when polled by an IN token by the USB Host Controller. Reset clears this bit.	1 = Send STALL handshake 0 = Default
XFREN Bit 10	**Transfer Enable**—This read/write bit enables a transfer to occur when the endpoint is polled by the USB Host controller. Reset clears this bit.	1 = Data is ready to be sent. 0 = If (EPEN = 1), the endpoint is enabled, but busy. Respond with NAK.
EPEN Bit 9	**Endpoint enabled**—This read/write bit enables the endpoint to be seen by the USB Host controller. If this bit is clear, any transactions addressing this endpoint will be ignored. Reset clears this bit for all endpoints.	1 = Endpoint is available to USB Host controller 0 = Endpoint is not available to USB Host controller
SEQ Bit 8	**Sequence**—This bit determines which type of data packet (DATA0 or DATA1) will be sent during the next IN transaction. The toggling of this bit is controlled by hardware (upon reception of a valid ACK handshake), but can be written by firmware to force a particular data token. Reset clears this bit.	1 = DATA1 Token active for next transaction 0 = DATA0 Token active for next transaction.
ACKF Bit 7	**Acknowledge flag**—This bit is set when a valid ACK handshake is received after the USB device endpoint data is successfully transmitted over the USB lines. This bit is cleared by a write of the IAEFC bit. Reset also clears this bit.	1 = ACK handshake received 0 = No ACK received

Table 21-15. USBE10CR Description

Name	Description	Settings
ERRF Bit 6	**Error flag**—This bit indicates an Error occurred during the last transmit operation. errors include: bit-stuff, CRC, and time-out. This bit is cleared by a write to the IAEFC bit. Reset also clears this bit.	1 = Error occurred 0 = No error detected
Bits 5-4	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A
B[3:0] Bits 3–0	**Byte Count**—These bits are read/writable and contain the number of bytes desired to be transferred at the next IN token request. Reset clears these bits.	N/A

USBCPUCR USB CPU Control Register **Addr $2485_2014**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
											SPEEDSEL					MEM MAP
TYPE	r	r	r	r	r	r	r	r	r	r	rw	rw	r	r	r	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-16. USBCPUCR Description

Name	Description	Settings
Bits 15–6	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A
SPEED SEL Bits 5-4	**USB Speed Select**—These read/write bits select the speed of the USB device. The default setting of these bits select the speed via the fsen port of the USB module. Reset clears these bits. For a low speed USB device, a pull-up is expected on the D- pin of the USB transceiver. For a full speed USB device, a pull-up is expected on the D+ pin of the USB transceiver. If these bits are not programmed, then the USB will respond exactly like Neptune LT. **Note:** Only change these bits when USB clock is disabled and the USB is in idle or reset state. **Note:** "J" and "K" bit polarity is reversed between low and full speed devices.	**SPEEDSEL** / **USB Speed** 0 0 — speed determined via fsen 0 1 — low speed (1.5 Mb/s) 1 0 — full speed (12 Mb/s) 1 1 — full speed (12 Mb/s)
Bits 3–1	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A

Table 21-16. USBCPUCR Description

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>MEMMAP
Bit 0</td><td>CPU Memory Map Select—This read/write bit selects between the two memory maps supported on the CPU. Reset clears this bit.

Setting this bit will increase the memory depth of EP1 and EP2 from 16 bytes to 32 bytes.

If this bit is not programmed, then the USB will respond exactly like Neptune LT.</td><td>1 = EP1 and EP2 are 32 bytes long instead of normal 16 byte depth

<table>
<tr><th>Endpoint</th><th>Size</th><th colspan="2">Address</th></tr>
<tr><td>EP0 RX</td><td>8 bytes</td><td>0x80</td><td>0x87</td></tr>
<tr><td>EP0 TX</td><td>8 bytes</td><td>0x88</td><td>0x8F</td></tr>
<tr><td>EP1 RX</td><td>32 bytes</td><td>0x90</td><td>0xAF</td></tr>
<tr><td>EP2 TX</td><td>32 bytes</td><td>0xB0</td><td>0xCF</td></tr>
<tr><td>EP3 RX</td><td>16 bytes</td><td>0xD0</td><td>0xDF</td></tr>
<tr><td>EP4 TX</td><td>32 bytes</td><td>0xE0</td><td>0xFF</td></tr>
<tr><td>EP10 TX</td><td>8 bytes</td><td>0x100</td><td>0x107</td></tr>
</table>

0 = Normal memory map (default)

<table>
<tr><th>Endpoint</th><th>Size</th><th colspan="2">Address</th></tr>
<tr><td>EP0 RX</td><td>8 bytes</td><td>0x80</td><td>0x87</td></tr>
<tr><td>EP0 TX</td><td>8 bytes</td><td>0x88</td><td>0x8F</td></tr>
<tr><td>EP1 RX</td><td>16 bytes</td><td>0x90</td><td>0x9F</td></tr>
<tr><td>EP2 TX</td><td>16 bytes</td><td>0xA0</td><td>0xAF</td></tr>
<tr><td>EP3 RX</td><td>16 bytes</td><td>0xB0</td><td>0xBF</td></tr>
<tr><td>EP4 TX</td><td>32 bytes</td><td>0xC0</td><td>0xDF</td></tr>
<tr><td>EP10 TX</td><td>8 bytes</td><td>0xE0</td><td>0xE7</td></tr>
</table>
</td></tr>
</table>

USBDINTE USB DSP Interrupt Enable Register **Addr Y:$FFC0**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
		RSRFC	SOF_en	EP9_en	EP8_en	EP7_en	EP6_en	EP5_en	EP11_en						SR_en	RST_en
TYPE	r	slfclr	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-17. USBDINTE Description

Name	Description	Settings
Bit 15	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A
RSRFC Bit 14	**Resume/Suspend/Reset Interrupt Flag Clear**—This bit clears the RES_int, SUSP_int, and RST_int flags in the DSP Interrupt enable register. This bit always reads as zero.	1 = Clear Resume/Suspend, USB reset interrupts 0 = Default
SOF_en Bit 13	**Start of Frame Interrupt Enable**—This bit enables an interrupt to be generated when the SOF interrupt flag is set.	1 = SOF interrupt enabled 0 = SOF interrupt disabled
EP9_en Bit 12	**Endpoint 9 Interrupt Enable**—This bit enables an interrupt to be generated whenever a transaction on this endpoint is completed.	1 = EP9 transaction interrupt enabled 0 = EP9 transaction interrupt disabled
EP8_en Bit 11	**Endpoint 8 Interrupt Enable**—This bit enables an interrupt to be generated whenever a transaction on this endpoint is completed.	1 = EP8 transaction interrupt enabled 0 = EP8 transaction interrupt disabled
EP7_en Bit 10	**Endpoint 7 Interrupt Enable**—This bit enables an interrupt to be generated whenever a transaction on this endpoint is completed.	1 = EP7 transaction interrupt enabled 0 = EP7 transaction interrupt disabled
EP6_en Bit 9	**Endpoint 6 Interrupt Enable**—This bit enables an interrupt to be generated whenever a transaction on this endpoint is completed.	1 = EP6 transaction interrupt enabled 0 = EP6 transaction interrupt disabled
EP5_en Bit 8	**Endpoint 5 Interrupt Enable**—This bit enables an interrupt to be generated whenever a transaction on this endpoint is completed.	1 = EP5 transaction interrupt enabled 0 = EP5 transaction interrupt disabled
EP11_en Bit 7	**Endpoint 11 Interrupt Enable**—This bit enables an interrupt to be generated whenever a transaction on this endpoint is completed.	1 = EP11 transaction interrupt enabled 0 = EP11 transaction interrupt disabled
Bits 6–2	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A
SR_en Bit 1	**Suspend / Resume Interrupt Enable**—This read/write bit enables an interrupt to be generated whenever RSM_int or the SUSP_int is set. Reset clears this bit.	1 = Suspend/Resume interrupt enabled 0 = Suspend/Resume interrupt disabled

Table 21-17. USBDINTE Description (Continued)

Name	Description	Settings
RST_en Bit 0	**Reset Interrupt Enable**—This read/write bit enables an interrupt to occur when the RST_en bit is set. Reset clears this bit.	1 = USB Reset interrupt enabled 0 = USB Reset interrupt disabled

USBDIR USB DSP Interrupt Register **Addr Y:$FFC1**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
			SOF_int	EP9_int	EP8_int	EP7_int	EP6_int	EP5_int	EP11_int					SUSP_int	RES_int	RST_int
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-18. USBDIR Description

Name	Description	Settings
Bits 15–14	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A
SOF_int Bit 13	**Start of Frame Interrupt Flag**—This bit is set when a SOF token PID is detected regardless of any CRC or time-out errors that may occur, or when a synthesized SOF occurs(i.e. a SOF packet was expected, but was not received). Reset also clears this bit.	1 = SOF event 0 = Default condition
EP9_int Bit 12	**Endpoint 9 Interrupt Flag**—This bit signifies that a transaction has completed on Endpoint 9. This will generate an interrupt if the corresponding EP9 interrupt enable bit is set in the IE register. This bit will be cleared by a write to the IAEFC bit in the EP9 Status/Ctl register. Reset also clears this bit.	1 = EP9 transaction done. 0 = Default
EP8_int Bit 11	**Endpoint 8 Interrupt Flag**—This bit signifies that a transaction has completed on Endpoint 8. This will generate an interrupt if the corresponding EP8 interrupt enable bit is set in the IE register. This bit will be cleared by a write to the IAEFC bit in the EP8 Status/Ctl register. Reset also clears this bit.	1 = EP8 transaction done. 0 = Default
EP7_int Bit 10	**Endpoint 7 Interrupt Flag**—This bit signifies that a transaction has completed on Endpoint 7. This will generate an interrupt if the corresponding EP7 interrupt enable bit is set in the IE register. This bit will be cleared by a write to the IEOFC bit in the EP7 Status/Ctl register. Reset also clears this bit.	1 = EP7 transaction done. 0 = Default
EP6_int Bit 9	**Endpoint 6 Interrupt Flag**—This bit signifies that a transaction has completed on Endpoint 6. This will generate an interrupt if the corresponding EP6 interrupt enable bit is set in the IE register. This bit will be cleared by a write to the IAFC bit in the EP6 Status/Ctl register. Reset also clears this bit.	1 = EP6 transaction done. 0 = Default

Table 21-18. USBDIR Description (Continued)

Name	Description	Settings
EP5_int Bit 8	**Endpoint 5 Interrupt Flag**—This bit signifies that a transaction has completed on Endpoint 5. This will generate an interrupt if the corresponding EP5 interrupt enable bit is set in the IE register. This bit will be cleared by a write to the IEOFC bit in the EP5 Status/Ctl register. Reset also clears this bit.	1 = EP5 transaction done. 0 = Default
EP11_int Bit 7	**Endpoint 11 Interrupt Flag**—This bit signifies that a transaction has completed on Endpoint 11. This will generate an interrupt if the corresponding EP11 interrupt enable bit is set in the IE register. This bit will be cleared by a write to the IAEFC bit in the EP11 Status/Ctl register. Reset also clears this bit.	1 = EP11 transaction done. 0 = Default
Bits 6–3	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A
SUSP_int Bit 2	**Suspend Interrupt Flag**—This bit is set when the USB SIE detects a suspend state on the USB data lines. Reset clears this bit.	1 = Suspend detected 0 = Default
RES_int Bit 1	**Resume Interrupt Flag**—This bit is set when the USB SIE detects a resume condition on the USB data lines (J to K transition). Reset clears this bit.	1 = Resume detected 0 = default
RST_int Bit 0	**Reset Interrupt Flag**—This bit is set when the USB SIE detects a reset condition on the USB data lines (SE0 > 2.5 us). Reset clears this bit.	1 = USB Reset detected 0 = default

USBSOF USB Start Of Frame Register **Addr Y:$FFC2**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
			ACKF	IAFC	FLOCK	FN[10:0]										
TYPE	r	r	rw	slfclr	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-19. USBSOF Description

Name	Description	Settings
Bits 15–14	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A
ACKF Bit 13	**Acknowledge Flag**—This bit is set when a valid SOF token packet is received. This bit is not set when a synthesized SOF event occurs.	1 = Valid SOF token packet 0 = No SOF token packet detected
IAFC Bit 12	**Interrupt and ACK Flag Clear**—Writing a one to this bit will clear the ACKF bits The SOF_int bit in the Interrupt register (USBDIR) will also be cleared. This bit always reads as a zero.	1 = Not-applicable 0 = Always reads as a zero.
FLOCK Bit 11	**Frame Timer Locked**—This read only bit signifies that the Host generated SOF is synchronized to the frame timer used for SOF synthesis. To be synchronized, the USB spec requires that an SOF token arrives within 2 clock cycles of the previous SOF token spaced between 11,985 and 12,015 clock cycles.	1 = Frame timer locked 0 = Frame timer not locked
FN[10:0] Bits 10–0	**Frame Number**—These bits represent the last frame number received during a SOF packet. Reset clears these bits.	N/A

USBE5CR USB Endpoint 5 (Isochronous Receive) Control Register **Addr Y:$FFC3**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	EPTYP 1	EPTYP 0	IEOFC	DBUF	DPM		EPEN		OVF	ERRF		B4	B3	B2	B1	B0
TYPE	r	r	slfclr	r	rw	r	rw	r	r	r	r	r	r	r	r	r
RESET	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-20. USBE5CR Description

Name	Description	Settings
EPTYP [1:0] Bits 15–14	**Endpoint Type**—These read only bits describe the type of endpoint. Endpoint 5 will always read as an isochronous endpoint, 10b.	<table><tr><th>EPTYP1</th><th>EPTYP0</th><th>Endpoint Type</th></tr><tr><td>0</td><td>0</td><td>Reserved</td></tr><tr><td>0</td><td>1</td><td>Reserved</td></tr><tr><td>1</td><td>0</td><td>Isochronous</td></tr><tr><td>1</td><td>1</td><td>Reserved</td></tr></table>
IEOFC Bit 13	**Interrupt, ERR, and Overflow Clear**—Writing a one to this bit will clear the OVF and ERRF bits The EP5_int bit in the Interrupt register (USBDIR) will also be cleared. This bit always reads as a zero. **Note:** DSP isochronous endpoints 5 and 6 have a interrupt clearing delay not present in other non-isochronous endpoints. Therefore, the following code can not be used to end the ISR: BSET #13, USBE5CR ; clear the interrupt flag RTI ; return from ISR A NOP is required between these two commands to guarantee that the ISR is not re-executed. For example: BSET #13, USBE5CR ; clear the interrupt flag NOP ; wait for clear to propagate RTI ; return from ISR	1 = Not-applicable 0 = Always reads as a zero.

Table 21-20. USBE5CR Description

Name	Description	Settings
DBUF Bit 12	**Current Data Buffer**—This read only bit determines which data buffer is currently active. The hardware will set or clear this bit depending on which data buffer has been received. In the event that both data buffers are received before the CPU can service either transaction interrupt, the DBUF bit will reflect the first data buffer received (i.e. the oldest data). At the clearing of the Interrupt flag for the active data buffer, the DBUF bit will switch to the other data buffer. If the current endpoint is not double buffered or the Dual Packet Mode bit is clear (DPM=0), this bit will read as zero. Reset clears this bit.	1 = Data buffer 1 active. 0 = Data Buffer 0 active.
DPM Bit 11	**Dual Packet Mode**—This bit controls whether a double-buffered endpoint operates in dual packet mode. When set, the endpoint toggles between data buffers zero and one for subsequent transactions. Reset clears this bit.	1 = Dual Packet Mode enabled. 0 = Dual Packet Mode disabled.
Bit 10	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A
EPEN Bit 9	**EPEN**—This read/write bit enables the endpoint to be seen by the USB Host controller. If this bit is clear, any transactions addressing this endpoint will be ignored. Reset clears this bit for all endpoints.	1 = Endpoint is available to USB Host controller 0 = Endpoint is not available to USB Host controller
Bit 8	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A
OVF Bit 7	**Receive Buffer Overflow**—This bit is set when a receive endpoint receives more than its maximum number of bytes. The EP5_int will be set on an overflow event. The firmware is responsible for determining whether an overflow event occurred. This bit cleared by a write to the IEOFC bit. reset clears this bit.	1 = Overflow occurred 0 = No overflow occurred
ERRF Bit 6	**Error flag**—This bit indicates an Error occurred during the last transmit operation. Errors include: bit-stuff, CRC, and time-out. This bit is cleared by a write to the IEOFC bit. Reset also clears this bit.	1 = Error occurred 0 = No error detected
Bit 5	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A
B[4:0] Bits 4–0	**Byte Count**—These bits are read only and reflect the number of bytes received during the last transaction. Reset clears these bits.	N/A

USBE6CR — USB Endpoint 6 (Isochronous Transmit) Control Register — **Addr Y:$FFC4**

BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
EPTYP1	EPTYP0	IAFC	DBUF	DPM		EPEN					B4	B3	B2	B1	B0
TYPE	r	r	slfclr	r	rw	r	rw	r	r	r	r	rw	rw	rw	rw	rw
RESET	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-21. USBE6CR Description

Name	Description	Settings
EPTYP [1:0] Bits 15–14	**Endpoint Type**—These read only bits describe the type of endpoint. Endpoint 6 will always read as an isochronous endpoint, 10b.	<table><tr><th>EPTYP1</th><th>EPTYP0</th><th>Endpoint Type</th></tr><tr><td>0</td><td>0</td><td>Reserved</td></tr><tr><td>0</td><td>1</td><td>Reserved</td></tr><tr><td>1</td><td>0</td><td>Isochronous</td></tr><tr><td>1</td><td>1</td><td>Reserved</td></tr></table>
IAFC Bit 13	**Interrupt and Acknowledge Flag Clear**—Writing a one to this bit will clear the acknowledge flag and the EP6_int bit in the Interrupt register (USBDIR) will also be cleared. This bit always reads as a zero. **Note:** DSP isochronous endpoints 5 and 6 have a interrupt clearing delay not present in other non-isochronous endpoints. Therefore, the following code can not be used to end the ISR: BSET #13, USBE6CR ; clear the interrupt flag RTI ; return from ISR A NOP is required between these two commands to guarantee that the ISR is not re-executed. For example: BSET #13, USBE6CR ; clear the interrupt flag NOP ; wait for clear to propagate RTI ; return from ISR	1 = Not-applicable 0 = Always reads as a zero

Table 21-21. USBE6CR Description (Continued)

Name	Description	Settings
DBUF Bit 12	**Current Data Buffer**—This read only bit determines which data buffer is currently active. The hardware will set or clear this bit depending on which data buffer has been sent. In the event that both data buffers are sent before the CPU can service either transaction interrupt, the DBUF bit will reflect the first data buffer sent (i.e. the oldest data). At the clearing of the Interrupt flag for the active data buffer, the DBUF bit will switch to the other data buffer. If the current endpoint is not double buffered or the Dual Packet Mode bit is clear (DPM=0), this bit will read as zero. Reset clears this bit.	1 = Data buffer 1 active. 0 = Data Buffer 0 active.
DPM Bit 11	**Dual Packet Mode**—This bit controls whether a double-buffered endpoint operates in dual packet mode. When set, the endpoint toggles between data buffers zero and one for subsequent transactions. Reset clears this bit.	1 = Dual Packet Mode enabled. 0 = Dual Packet Mode disabled.
Bit 10	Reserved bit. Read as zero. Must be written with zero for future compatibility.	N/A
EPEN Bit 9	**Endpoint enabled**—This read/write bit enables the endpoint to be seen by the USB Host controller. If this bit is clear, any transactions addressing this endpoint will be ignored. Reset clears this bit for all endpoints.	1 = Endpoint is available to USB Host controller 0 = Endpoint is not available to USB Host controller
Bits 8–5	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A
B[4:0] Bits 4–0	**Byte Count**—These bits are read/writable and contain the number of bytes desired to be transferred at the next SOF token received. Reset clears these bits.	N/A

USBE7CR — USB Endpoint 7 (Bulk or Interrupt Receive) Control Register — **Addr Y:$FFC5**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	EPTYP1	EPTYP0	IEOFC	SET	FSTALL	XFREN	EPEN	SEQ	OVF	ERRF		B4	B3	B2	B1	B0
TYPE	rw	r	slfclr	slfclr	rw	rw	rw	rw	r	r	r	r	r	r	r	r
RESET	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-22. USBE7CR Description

Name	Description	Settings
EPTYP [1:0] Bits 15–14	**Endpoint Type**—These bits describe the type of endpoint that is programmed. This endpoint can be either a Bulk or Interrupt RX endpoint. So only values of 01b or 11b are valid. Therefore bit EPTYP1 is a read/write bit and bit EPTYP0 is read-only and always read as one.	**EPTYP1** / **EPTYP0** / **Endpoint Type** 0 / 0 / Reserved 0 / 1 / Bulk 1 / 0 / Reserved 1 / 1 / Interrupt RX
IEOFC Bit 13	**Interrupt, ERR, and Overflow Flag Clear**—Writing a one to this bit will clear the OVF and ERRF bits. The EP7_int bit in the Interrupt register (USBDIR) will also be cleared. This bit always reads as a zero.	1 = Not-applicable 0 = Always reads as a zero.
SET Bit 12	**Set SEQ and EPTYP1 bits**—This write only bit controls the setting of the SEQ and EPTYP1 bits. This bit will read as zero when read.	1 = Set SEQ and EPTYP1 to values from data bus 0 = Do not allow SEQ or EPTYP1 to be updated from data bus.
FSTALL Bit 11	**Force STALL Response**—This read/write bit causes the endpoint to return a STALL handshake when polled by an OUT token by the USB Host Controller. Reset clears this bit.	1 = Send STALL handshake 0 = Default
XFREN Bit 10	**Transfer Enable**—This read/write bit enables a transfer to occur when the endpoint is polled by the USB Host controller. Reset clears this bit.	1 = Data is ready to be received. 0 = If (EPEN = 1), the endpoint is enabled, but busy. Respond with NAK.
EPEN Bit 9	**Endpoint enabled**—This read/write bit enables the endpoint to be seen by the USB Host controller. If this bit is clear, any transactions addressing this endpoint will be ignored. Reset clears this bit for all endpoints.	1 = Endpoint is available to USB Host controller 0 = Endpoint is not available to USB Host controller

Table 21-22. USBE7CR Description (Continued)

Name	Description	Settings
SEQ Bit 8	**Sequence bit**—This bit determines which type of data packet (DATA0 or DATA1) is expected during the next OUT transaction. The toggling of this bit is controlled by hardware (upon reception of a valid ACK handshake), but can be written by firmware to force a particular data token. Reset clears this bit.	1 = DATA1 Token active for next transaction 0 = DATA0 Token active for next transaction.
OVF Bit 7	**Receive Buffer Overflow**—This bit is set when a receive endpoint receives more than its maximum number of bytes. The EP7_int will be set on an overflow event. The firmware is responsible for determining whether an overflow event occurred. This bit cleared by a write to the IEOFC bit. reset clears this bit.	1 = Overflow occurred 0 = No overflow occurred
ERRF Bit 6	**Error flag**—This bit indicates an Error occurred during the last receive operation. Errors include: bit-stuff, CRC, and time-out. This bit is cleared by a write to the IEOFC bit. Reset also clears this bit.	1 = Error occurred 0 = No error detected
Bit 5	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A
B[4:0] Bits 4–0	**Byte Count**—These bits are read only and reflect the number of bytes received during the last transaction. Reset clears these bits.	N/A

USBE8CR — USB Endpoint 8 (Bulk or Interrupt Transmit) Control Register — **Addr Y:$FFC6**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	EPTYP 1	EPTYP 0	IAEFC	SET	FST ALL	XFRE N	EPEN	SEQ	ACKF	ERRF		B4	B3	B2	B1	B0
TYPE	rw	r	slfclr	slfclr	rw	rw	rw	rw	r	r	r	rw	rw	rw	rw	rw
RESET	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-23. USBE8CR Description

Name	Description	Settings
EPTYP [1:0] Bits 15–14	**Endpoint Type**—These bits describe the type of endpoint that is programmed. This endpoint can be either a Bulk or Interrupt TX endpoint. So only values of 01b or 11b are valid. Therefore bit EPTYP1 is a read/write bit and bit EPTYP0 is read-only and always read as one.	**EPTYP1** / **EPTYP0** / **Endpoint Type** 0 / 0 / Reserved 0 / 1 / Bulk 1 / 0 / Reserved 1 / 1 / Interrupt TX
IAEFC Bit 13	**Interrupt, ACK, and ERR Flag Clear**—Writing a one to this bit will clear the ACKF and ERRF bits The EP8_int bit in the Interrupt register (USBDIR) will also be cleared. This bit always reads as a zero.	1 = Not-applicable 0 = Always reads as a zero.
SET Bit 12	**Set SEQ and EPTYP1 bits**—This write only bit controls the setting of the SEQ and EPTYP1 bits. This bit will read as zero when read.	1 = Set SEQ and EPTYP to values from data bus 0 = Do not allow SEQ or EPTYP1 to be updated from data bus.
FSTALL Bit 11	**Force STALL Response**—This read/write bit causes the endpoint to return a STALL handshake when polled by an IN token by the USB Host Controller. Reset clears this bit.	1 = Send STALL handshake 0 = Default
XFREN Bit 10	**Transfer Enable**—This read/write bit enables a transfer to occur when the endpoint is polled by the USB Host controller. Reset clears this bit.	1 = Data is ready to be sent. 0 = If (EPEN = 1), the endpoint is enabled, but busy. Respond with NAK.
EPEN Bit 9	**Endpoint enabled**—This read/write bit enables the endpoint to be seen by the USB Host controller. If this bit is clear, any transactions addressing this endpoint will be ignored. Reset clears this bit for all endpoints.	1 = Endpoint is available to USB Host controller 0 = Endpoint is not available to USB Host controller

Table 21-23. USBE8CR Description (Continued)

Name	Description	Settings
SEQ Bit 8	**Sequence**—This bit determines which type of data packet (DATA0 or DATA1) will be sent during the next IN transaction. The toggling of this bit is controlled by hardware (upon reception of a valid ACK handshake), but can be written by firmware to force a particular data token. Reset clears this bit.	1 = DATA1 Token active for next transaction 0 = DATA0 Token active for next transaction.
ACKF Bit 7	**Acknowledge flag**—This bit is set when a valid ACK handshake is received after the USB device endpoint data is successfully transmitted over the USB lines. This bit is cleared by a write of the IAEFC bit. Reset also clears this bit.	1 = ACK handshake received 0 = No ACK received
ERRF Bit 6	**Error flag**—This bit indicates an Error occurred during the last transmit operation. Errors include: bit-stuff, time-out, and CRC. This bit is cleared by a write to the IAEFC bit. Reset also clears this bit.	1 = Error occurred 0 = No error detected
Bit 5	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A
B[4:0] Bits 4–0	**Byte Count**—These bits are read/writable and contain the number of bytes desired to be transferred at the next IN token request. Reset clears these bits.	N/A

USBE9CR — USB Endpoint 9 (Interrupt Transmit) Control Register — **Addr Y:$FFC7**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	EPTYP 1	EPTYP 0	IAEFC	SET SEQ	FST ALL	XFRE N	EPEN	SEQ	ACKF	ERRF		B4	B3	B2	B1	B0
TYPE	r	r	slfclr	slfclr	rw	rw	rw	rw	r	r	r	rw	rw	rw	rw	rw
RESET	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-24. USBE9CR Description

Name	Description	Settings
EPTYP [1:0] Bits 15-14	**Endpoint Type**—These read only bits describe the type of endpoint. Endpoint 9 will always read as an interrupt endpoint, 11b.	<table><tr><th>EPTYP1</th><th>EPTYP0</th><th>Endpoint Type</th></tr><tr><td>0</td><td>0</td><td>Reserved</td></tr><tr><td>0</td><td>1</td><td>Reserved</td></tr><tr><td>1</td><td>0</td><td>Reserved</td></tr><tr><td>1</td><td>1</td><td>Interrupt</td></tr></table>
IAEFC Bit 13	**Interrupt, ACK, and ERR Flag Clear**—Writing a one to this bit will clear the ACKF and ERRF bits The EP11_int bit in the Interrupt register (USBDIR) will also be cleared. This bit always reads as a zero.	1 = Not-applicable 0 = Always reads as a zero.
SETSEQ Bit 12	**Set SEQ bit**—This write only bit controls the setting of the SEQ bit. This bit will read as zero when read.	1 = Set SEQ to value from data bus 0 = Do not allow SEQ to be updated from data bus.
FSTALL Bit 11	**Force STALL Response**—This read/write bit causes the endpoint to return a STALL handshake when polled by an IN token by the USB Host Controller. Reset clears this bit.	1 = Send STALL handshake 0 = Default
XFREN Bit 10	**Transfer Enable**—This read/write bit enables a transfer to occur when the endpoint is polled by the USB Host controller. Reset clears this bit.	1 = Data is ready to be sent. 0 = If (EPEN = 1), the endpoint is enabled, but busy. Respond with NAK.
EPEN Bit 9	**Endpoint enabled**—This read/write bit enables the endpoint to be seen by the USB Host controller. If this bit is clear, any transactions addressing this endpoint will be ignored. Reset clears this bit for all endpoints.	1 = Endpoint is available to USB Host controller 0 = Endpoint is not available to USB Host controller
SEQ Bit 8	**Sequence**—This bit determines which type of data packet (DATA0 or DATA1) will be sent during the next IN transaction. The toggling of this bit is controlled by hardware (upon reception of a valid ACK handshake), but can be written by firmware to force a particular data token. Reset clears this bit.	1 = DATA1 Token active for next transaction 0 = DATA0 Token active for next transaction.

Table 21-24. USBE9CR Description (Continued)

Name	Description	Settings
ACKF Bit 7	**Acknowledge flag**—This bit is set when a valid ACK handshake is received after the USB device endpoint data is successfully transmitted over the USB lines. This bit is cleared by a write of the IAEFC bit. Reset also clears this bit.	1 = ACK handshake received 0 = No ACK received
ERRF Bit 6	**Error flag**—This bit indicates an Error occurred during the last transmit operation. Errors include: bit-stuff, CRC, and time-out. This bit is cleared by a write to the IAEFC bit. Reset also clears this bit.	1 = Error occurred 0 = No error detected
Bit 5	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A
B[4:0] Bits 4–0	**Byte Count**—These bits are read/writable and contain the number of bytes desired to be transferred at the next IN token request. Reset clears these bits.	N/A

USBBPR USB DSP Buffer Pointer Register **Addr Y:$FFC8**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									PTR7	PTR6	PTR5	PTR4	PTR3	PTR2	PTR1	PTR0
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-25. USBBPR Description

Name	Description	Settings
Bits 15–8	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A
PTR[7:0]	**Buffer Pointer**—This16-bit read/write register is used as a pointer to the buffer memory. This value will be incremented by 2 upon each access into the DSP buffer. It is the responsibility of the user to ensure that this address is initialized properly prior to reading the buffer. This means the user must programed to a byte address where the initial value must be even.	N/A

USBDBR DSP Buffer Register **Addr Y:$FFC9**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-26. USBDBR Description

Name	Description	Settings
D[15:0]	**Buffer Register**—This16-bit read/write register is used as a data port into the DSP buffer memory. All reads and writes to the DSP buffer will occur from this register.	N/A

USBDRAT DSP Buffer Register for Array Test Mode **Addr Y:$FFCA**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-27. USBDRAT Description

Name	Description	Settings
D[15:0]	**Buffer Register**—This16-bit read/write register is used alternatively to read data from the buffer that normally in only accessible to the USB Serial Interface engine. When Array Test mode is active, the buffer addresses and data lines are adjusted to give visibility to the USB side of the dual address buffers within the module.	N/A

USBE11CR — USB Endpoint 11 (Interrupt Transmit) Control Register — **Addr Y:$FFCB**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	EPTYP 1	EPTYP 0	IAEFC	SET SEQ	FST ALL	XFRE N	EPEN	SEQ	ACKF	ERRF			B3	B2	B1	B0
TYPE	r	r	slfclr	slfclr	rw	rw	rw	rw	r	r	r	r	rw	rw	rw	rw
RESET	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-28. USBE11CR Description

Name	Description	Settings
EPTYP [1:0] Bits 15-14	**Endpoint Type**—These read only bits describe the type of endpoint. Endpoint 9 will always read as an interrupt endpoint, 11b.	<table><tr><th>EPTYP1</th><th>EPTYP0</th><th>Endpoint Type</th></tr><tr><td>0</td><td>0</td><td>Reserved</td></tr><tr><td>0</td><td>1</td><td>Reserved</td></tr><tr><td>1</td><td>0</td><td>Reserved</td></tr><tr><td>1</td><td>1</td><td>Interrupt</td></tr></table>
IAEFC Bit 13	**Interrupt, ACK, and ERR Flag Clear**—Writing a one to this bit will clear the ACKF and ERRF bits. The EP9_int bit in the Interrupt register (USBDIR) will also be cleared. This bit always reads as a zero.	1 = Not-applicable 0 = Always reads as a zero.
SETSEQ Bit 12	**Set SEQ bit**—This write only bit controls the setting of the SEQ bit. This bit will read as zero when read.	1 = Set SEQ to value from data bus 0 = Do not allow SEQ to be updated from data bus.
FSTALL Bit 11	**Force STALL Response**—This read/write bit causes the endpoint to return a STALL handshake when polled by an IN token by the USB Host Controller. Reset clears this bit.	1 = Send STALL handshake 0 = Default
XFREN Bit 10	**Transfer Enable**—This read/write bit enables a transfer to occur when the endpoint is polled by the USB Host controller. Reset clears this bit.	1 = Data is ready to be sent. 0 = If (EPEN = 1), the endpoint is enabled, but busy. Respond with NAK.
EPEN Bit 9	**Endpoint enabled**—This read/write bit enables the endpoint to be seen by the USB Host controller. If this bit is clear, any transactions addressing this endpoint will be ignored. Reset clears this bit for all endpoints.	1 = Endpoint is available to USB Host controller 0 = Endpoint is not available to USB Host controller
SEQ Bit 8	**Sequence**—This bit determines which type of data packet (DATA0 or DATA1) will be sent during the next IN transaction. The toggling of this bit is controlled by hardware (upon reception of a valid ACK handshake), but can be written by firmware to force a particular data token. Reset clears this bit.	1 = DATA1 Token active for next transaction 0 = DATA0 Token active for next transaction.

Table 21-28. USBE11CR Description

Name	Description	Settings
ACKF Bit 7	**Acknowledge flag**—This bit is set when a valid ACK handshake is received after the USB device endpoint data is successfully transmitted over the USB lines. This bit is cleared by a write of the IAEFC bit. Reset also clears this bit.	1 = ACK handshake received 0 = No ACK received
ERRF Bit 6	**Error flag**—This bit indicates an Error occurred during the last transmit operation. errors include: bit-stuff, CRC, and time-out. This bit is cleared by a write to the IAEFC bit. Reset also clears this bit.	1 = Error occurred 0 = No error detected
Bits 5-4	Reserved. Read as zero. Must be written with zero for future compatibility.	N/A
B[3:0] Bits 3–0	**Byte Count**—These bits are read/writable and contain the number of bytes desired to be transferred at the next IN token request. Reset clears these bits.	N/A

Chapter 22
Universal Asynchronous Receiver/Transmitter (UART)

Revision History Table

Revision	Date	Author	Changes
1.0	10/8/99		Initial Release.
1.1	03/20/2002	Sarvesh Verma	Updated specs for Neptune LTS.
1.2	05/09/2002	Sarvesh Verma	Updated specs as per ddts TSLbo23381
1.3	06/19/02	Shannon Osgood	Added base address variables to register descriptions.
1.4	05/07/03		Updated for LTE specification release.

This section describes Universal Asynchronous Receiver Transmitter (UART) module instantiated in the Neptune LTE IC. They are capable of standard RS-232 type Non Return to Zero (NRZ) encoding format and IrDA-compatible infrared modes. The UART provides serial communication with external devices via RS-232 cable or external circuitry which converts infrared signals to electrical signals (for receptions) or transforms electrical signals to signals that drive an infrared LED (for transmissions). This functionality allows for low speed IrDA compatibility.

This UART transmits and receives a character length of 7 or 8 bits. For transmission, data is passed to a Transmitter FIFO, of 64 bytes in depth, from the Peripheral Data Bus. This data is passed to the shift register and shifted serially out on the TXD pin. For reception, data is received serially from the RXD pin and stored in a Receiver FIFO of 64 bytes in depth. The received data is retrieved from the Receiver FIFO on the Peripheral Data Bus. The Receiver and Transmitter FIFO contain a maskable interrupt that can be configured to interrupt when it reaches a certain level.

The UART generated baud rate is based upon a configurable divisor and input clock. The UART also contains autobaud detection circuitry. It can be configured to send one or two stop bits as well as odd, even, or no parity. The receiver detects framing errors, idle conditions, break characters, parity errors, and overrun errors.

The transmitter TXD pin can be configured for open-drain at the chip boundary. This module uses a software interface for the control of modem operations. This module also has a Serial IR module which can decode and encode IrDA-compatible serial infrared data.

22.1 Features

- 7 or 8 data bits
- 1 or 2 Stop bits
- Programmable Parity (even, odd, and none)
- Full 8-wire serial DCE interface[1] for Modem Flow Control
  - — Hardware Flow control support for RTS, and CTS signals
  - — Software Flow control support for DSR, DCD, and RI signals
  - — RTS and DTR edge detect interrupt (edge selectable)
  - — Status Flags for various Flow Control and FIFO states
  - — Serial Infrared Interface (low speed - IrDA-compatible) enable via the UART Control Register 1
  - — Voting logic for improved noise immunity (16X oversampling)
  - — Transmitter FIFO Empty Interrupt Suppression
  - — Wakeup indication to the UART_CLK generator
  - — Auto Baud Rate Feature
  - — Receiver and Transmitter Enable/Disable
  - — RTS, DTR, IrDA RX, and Receive WAKE interrupts will wake MCU from STOP
  - — 13 maskable interrupts
  - — Low power modes
  - — Escape Character Sequence Detection
  - — Software Reset

NOTE:

If the 13MHz, 16.8MHz, and 19.44MHz reference clock frequency bits are cleared, the Escape Character Sequence Detection, BRM Preset Registers, and SIR low speed features (IRSC = 1) are disabled.

1. UART GPIO pins should be used for DTR,DSR,DCD,RI functions.

22.2 UART Module Interface Signals

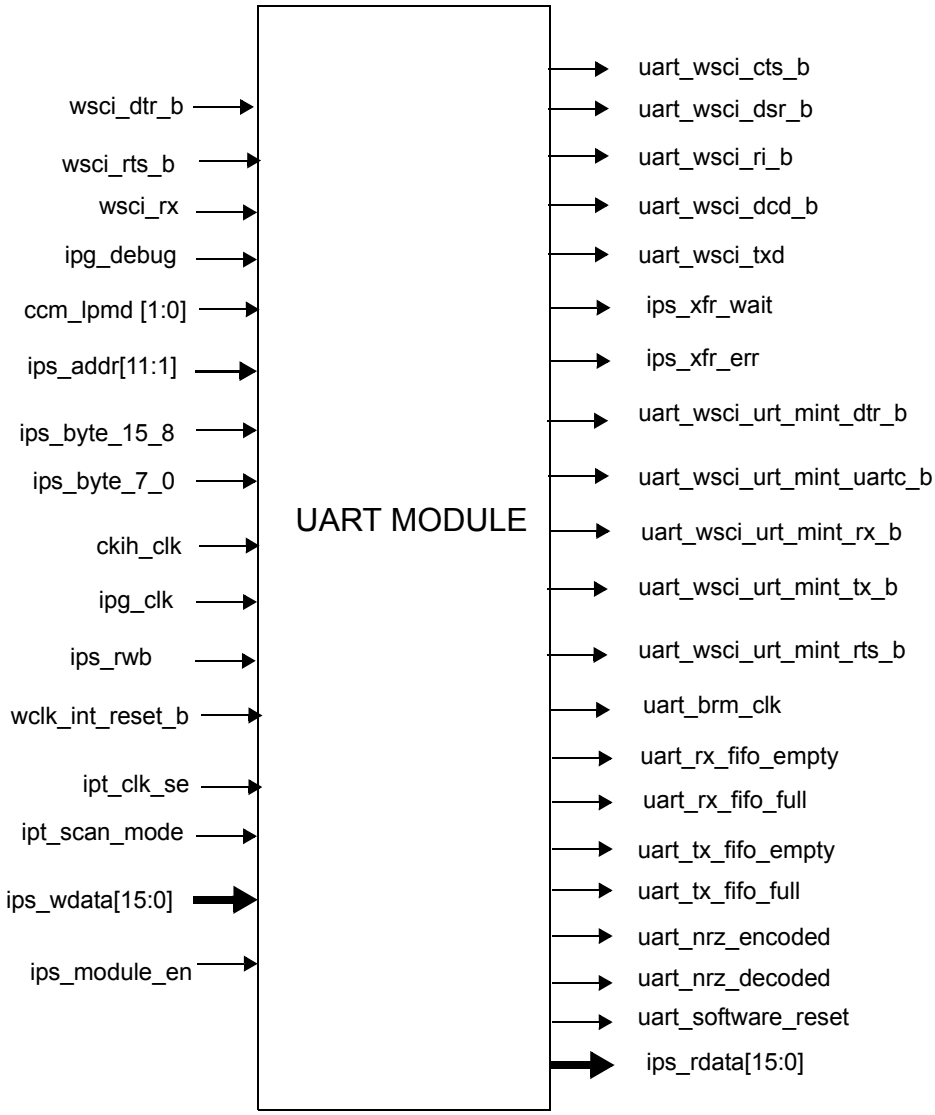


Figure 22-1. UART Module Interface Signals

22.2.1 UART Module Pin List

Table 22-1 is a list of all the UART Module Interface signals.

Table 22-1. UART Module Pin List

Pin Names	Direction	Description	Active
Peripheral Bus Signals:			
ips_addr[11:1]	Input	Address bus .	HIGH
ips_wdata[15:0]	Input	Data bus to/from IP bus.	
ips_byte_15_8	Input	Byte enable signals.	LOW
ips_byte_7_0	Input	Byte enable signals.	LOW
ips_module_en	Input	Module select.	LOW
ips_xfr_wait	Output	Module transfer acknowledge	LOW
ips_xfr_err	Output	Module transfer error acknowledge	LOW
ips_rwb	Input	Module read/write	HIGH
ccm_lpmd[1:0]	Input	Low power signals	HIGH
ipg_debug	Input	Module Debug signals	LOW
Reset:			
wclk_int_reset_b	Input	This active low input signal resets the UART block	LOW
Interrupts:			
uart_wsci_urt_mint_rx_b	Output	Receive Ready, IDLE Detect Interrupts, Receive Data Ready (RDR), Receiver IDLE (RXDS) (OR of 4)	LOW
uart_wsci_urt_mint_tx_b	Output	TX FIFO Empty, TX FIFO Ready, Transmit Complete Interrupts (OR of 3)	LOW
uart_wsci_urt_mint_rts_b	Output	RTS Delta interrupt, RTS edge programmable interrupt (OR of 2)	LOW
uart_wsci_urt_mint_dtr_b	Output	DTR Interrupt	LOW
uart_wsci_urt_mint_uartc_b	Output	Common interrupt. RX Overrun, Break, WAKE, Autobaud, ESCAPE, IrDA, AWAKE (asynchronous WAKE for NRZ data), AIRINT (asynchronous IR pulse detect) (OR of 8)	LOW
Serial Signals:			
uart_wsci_txd	Output	Transmitter serial output (mux'ed IR and NRZ data format)	HIGH
wsci_rx	Input	Receiver serial input (mux'ed IR and NRZ data format)	HIGH

Table 22-1. UART Module Pin List (Continued)

Pin Names	Direction	Description	Active
Modem Control Signals:			
wsci_dtr_b	Input	Data terminal ready signal for modem control	LOW
uart_wsci_dsr_b	Output	Data set ready signal for modem control	LOW
uart_wsci_ri_b	Output	Ring Indicator signal for modem control	LOW
uart_wsci_dcd_b	Output	Data Carrier Detect signal for modem control	LOW
wsci_rts_b	Input	This input is used to control the transmitter. By asserting RTS, the far-end device signals to the UART it's ready to receive. Normally, the transmitter waits until this signal is active (low) before a character is transmitted. If the IRTS (Ignore RTS) bit is set, the transmitter sends a character whenever a character is ready to transmit. This pin can then be used as a general purpose input whose status is read in the RTSS bit. This pin can post an interrupt on any transition of this pin and wake up the RCE core from STOP on its assertion. If transmitting when $\overline{\text{RTS}}$ is negated, the UART transmitter will complete the transmission of the current character, then shut off. The contents of the FIFO (characters to be transmitted) will remain undisturbed.	LOW
uart_wsci_cts_b	Output	This output pin serves two purposes. Normally, the receiver indicates that it is ready to receive data by asserting this pin (low). When the receiver detects a pending overrun, it negates this pin. For other applications, this pin can be a general purpose output controlled by the CTS bit in the UART Control Register 2.	LOW
Clocks:			
ckih_clk	Input	This is the clock input to the BRM. The serial clock is generated by multiplying this input according to the ratio written to the BRM registers. This clock can have one of the following frequencies: 13 Mhz, 16.8 Mhz and 19.44 Mhz. In LTE we will be using 13Mhz frequency only.	
ipg_clk	Input	Mcore clock	
Testing:			
ipt_clk_se	Input	This signal enables the scan logic for testing purposes.	HIGH
ipt_scan_mode	Input	This signal puts the module in SCAN mode	HIGH
Debug Signals:			

Table 22-1. UART Module Pin List (Continued)

Pin Names	Direction	Description	Active
uart_tx_fifo_empty	Output	This signal indicates the TX FIFO is empty	HIGH
uart_rx_fifo_empty	Output	This signal indicates the RX FIFO is empty	HIGH
uart_tx_fifo_full	Output	This signal indicates the TX FIFO is full	HIGH
uart_rx_fifo_full	Output	This signal indicates the RX FIFO is full	HIGH
uart_nrz_encoded	Output	Received IR data converted to NRZ format	HIGH
uart_nrz_decoded	Output	Transmit data decoded from NRZ format to IR format	HIGH
uart_brm_clk	Output	This is the clock output from the BRM.	
uart_software_reset	Output	This signal is the software reset. It indicates the status of the software reset.	HIGH

22.3 UART Memory Map

This section shows the registers of the UART module and their mapped address(es). All the registers can be accessed either as a halfword or as a byte. Table 22-2 gives the register map of the UART. The RX and TX registers are mapped to 16 word addresses in order to support Load Multiple Registers (LDM) instruction and Store Multiple Registers (STM) instruction, respectively.

Table 22-2. Primary UART Memory Map

Address	Register
$2484_D000	UART Receiver Register (URXD) (16 multiple addresses)
$2484_D004	
$2484_D000 + (4*n)	
$2484_D03C	
$2484_D040	UART Transmitter Register (UTXD) (16 multiple addresses)
$2484_D044	
$2484_D000 + (40+4*n)	
$2484_D07C	
$2484_D080	UART Control Register 1 (UCR1)
$2484_D082	UART Control Register 2 (UCR2)
$2484_D084	UART Control Register 3 (UCR3)
$2484_D086	UART Control Register 4 (UCR4)
$2484_D088	UART FIFO Control Register (UFCR)

Table 22-2. Primary UART Memory Map (Continued)

Address	Register
$2484_D08A	UART Status Register 1 (USR1)
$2484_D08C	UART Status Register 2(USR2)
$2484_D08E	UART Escape Detection Register (UESC)
$2484_D090	UART Escape Timer Register (UTIM)
$2484_D092	UART BRM INC Register (UBIR)
$2484_D094	UART BRM MOD Register (UBMR)
$2484_D096	UART Baud Rate Count Register (UBRC)
$2484_D098	UART BRM INC Preset Register 1 (BIPR1)
$2484_D09A	UART BRM MOD Preset Register 1 (BMPR1)
$2484_D09C	UART BRM INC Preset Register 2 (BIPR2)
$2484_D09E	UART BRM MOD Preset Register 2 (BMPR2)
$2484_D0A0	UART BRM INC Preset Register 3 (BIPR3)
$2484_D0A2	UART BRM MOD Preset Register 3 (BMPR3)
$2484_D0A4	UART BRM INC Preset Register 4 (BIPR4)
$2484_D0A6	UART BRM MOD Preset Register 4 (BMPR4)
$2484_D0A8	UART Test Register (UTS)
$2484_D0AA - $2484_DFFF	Reserved

Table 22-3. Secondary UART Memory Map

Address	Register
$2485_7000	UART Receiver Register (URXD) (16 multiple addresses)
$2485_7004	
$2485_7000 + (4*n)	
$2485_703C	
$2485_7040	UART Transmitter Register (UTXD) (16 multiple addresses)
$2485_7044	
$2485_7000 + (40+4*n)	
$2485_707C	
$2485_7080	UART Control Register 1 (UCR1)
$2485_7082	UART Control Register 2 (UCR2)

Table 22-3. Secondary UART Memory Map

Address	Register
$2485_7084	UART Control Register 3 (UCR3)
$2485_7086	UART Control Register 4 (UCR4)
$2485_7088	UART FIFO Control Register ()
$2485_708A	UART Status Register 1 (USR1)
$2485_708C	UART Status Register 2(USR2)
$2485_708E	UART Escape Detection Register (UESC)
$2485_7090	UART Escape Timer Register (UTIM)
$2485_7092	UART BRM INC Register (UBIR)
$2485_7094	UART BRM MOD Register (UBMR)
$2485_7096	UART Baud Rate Count Register (UBRC)
$2485_7098	UART BRM INC Preset Register 1 (BIPR1)
$2485_709A	UART BRM MOD Preset Register 1 (BMPR1)
$2485_709C	UART BRM INC Preset Register 2 (BIPR2)
$2485_709E	UART BRM MOD Preset Register 2 (BMPR2)
$2485_70A0	UART BRM INC Preset Register 3 (BIPR3)
$2485_70A2	UART BRM MOD Preset Register 3 (BMPR3)
$2485_70A4	UART BRM INC Preset Register 4 (BIPR4)
$2485_70A6	UART BRM MOD Preset Register 4 (BMPR4)
$2485_70A8	UART Test Register (UTS)
$2485_70AA - $2485_7FFF	Reserved

22.4 Interrupts

This table illustrates all the interrupts that are output on each interrupt output. Refer to the registers for explanation of enables and status flags.

Table 22-4. Interrupts

INTERRUPT OUTPUT	Interrupt Enable	Enable Register Location	Interrupt Flag	Flag Register Location
uart_wsci_urt_mint_dtr_b	DTREN	UCR3 (bit13)	DTRF	USR2 (bit 13)
uart_wsci_urt_mint_rts_b	RTSDEN RTSEN	UCR1 (bit5) UCR2 (bit 4)	RTSD RTSF	USR1 (bit 12) USR2 (bit 4)

Table 22-4. Interrupts (Continued)

INTERRUPT OUTPUT	Interrupt Enable	Enable Register Location	Interrupt Flag	Flag Register Location
uart_wsci_urt_mint_rx_b	RRDYEN IDEN DREN RXDSEN	UCR1 (bit 9) UCR1 (bit 12) UCR4 (bit 0) UCR3 (bit 6)	RRDY IDLE RDR RXDS	USR1 (bit 9) USR2 (bit 12) USR2 (bit 0) USR2(bit 6)
uart_wsci_urt_mint_tx_b	TXMPTYEN TRDYEN TCEN	UCR1 (bit 6) UCR1 (bit 13) UCR4 (bit 3)	TXFE TRDY TXDC	USR2 (bit 14) USR1 (bit 13) USR2 (bit 3)
uart_wsci_urt_mint_uartc_b	OREN BKEN WKEN ADEN ESCI ENIRI AIRINTEN AWAKEN	UCR4 (bit 1) UCR4 (bit 2) UCR4 (bit 7) UCR1 (bit 15) UCR2 (bit 15) UCR4 (bit 8) UCR3 (bit 5) UCR3 (bit 4)	ORE BRCD WAKE ADET ESCF IRINT AIRINT AWAKE	USR2 (bit 1) USR2 (bit 2) USR2 (bit 7) USR2 (bit 15) USR1 (bit 11) USR2 (bit 8) USR1 (bit 5) USR1 (bit 4)

22.5 General UART Definitions

A few definitions will help in understanding the following discussions.

- Bit Time - A bit time is the period of time required to serially transmit or receive one bit of data and is equal to one cycle of the baud rate frequency.
- Start Bit - A Start bit is a bit time of logic “zero” which indicates the beginning of a data “Frame”. A start bit must begin with a “one” to “zero” transition and is preceded by at least one bit time of logic “one”.
- Stop Bit - A Stop bit is one bit time of logic “one” which indicates the end of a data “Frame”.
- Break - A “Frame” in which all the data bits, including the stop bit, are logic “zero”. This kind of a “Frame” is generally sent to signal the end of a message or the beginning of a new message.
- Frame - A “Frame” consists of a start bit followed by a specified number of data or information bits terminated by a stop bit. The number of data or information bits depends on the format specified and must agree between the transmitting device and the receiving device. The most common “Frame” format is one start bit followed by eight data bits (LSB first) terminated by one stop bit. An additional stop bit and parity bit may also be included.
- Framing Error - An error condition in which the stop bit of the received frame was missing. A framing error results when the frame boundaries in the received bit stream are not synchronized with the receiver bit counter. Framing errors are not always detected: if a data bit in the expected stop bit time happens to be a logic one, the framing error may go undetected. A framing error is always present on the receiver side, when the transmitter is sending breaks. However, if the UART is setup to expect two stop bits, and only the first stop bit is received, then this is not a framing error by definition.
- Parity Error - An error condition in which the calculated parity of the received data bits in the “Frame” is different from the parity bit received on the RXD line. Parity error is only calculated after an entire “Frame” is received.

- Overrun Error - An error condition in which the latest character received is ignored to prevent overwriting an already existing character in the UART Receiver Buffer. An overrun error indicates that the software reading the Buffer is not keeping up with the actual reception of characters on the RXD line.

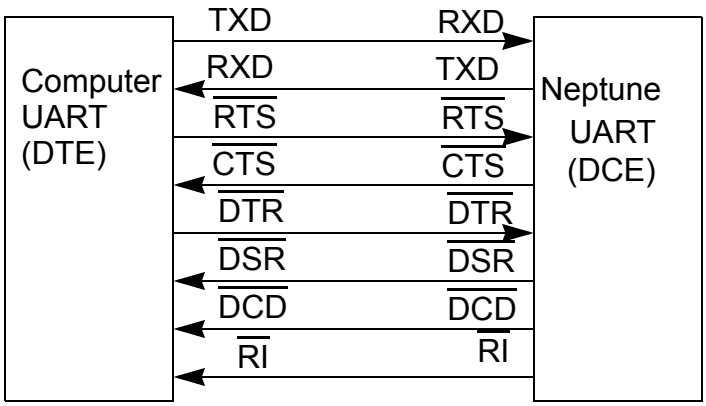


Figure 22-2. General Connections for a UART with a Modem

22.6 $\overline{\text{RTS}}$ - UART Request To Send

This input is used to control the transmitter. By asserting $\overline{\text{RTS}}$, the far-end device signals to the UART it is ready to receive. Normally, the transmitter waits until this signal is active (low) before a character is transmitted. If the IRTS (Ignore RTS) bit is set, the transmitter sends a character whenever a character is ready to transmit. This pin can post an interrupt on any transition of this pin and wake up the M•CORE from STOP on its assertion. If transmitting when $\overline{\text{RTS}}$ is negated, the UART transmitter will complete the transmission of the current character, then shut off. The contents of the FIFO (characters to be transmitted) will remain undisturbed.

22.6.1 RTS Edge Triggered Interrupt

The $\overline{\text{RTS}}$ pin can be configured to cause an interrupt on a selectable edge. To enable the $\overline{\text{RTS}}$ pin to cause an interrupt, set RTSEN = 1 in the UART Control Register 2 (UCR2). Write a “one” to the RTSF bit of the UART Status Register to clear the interrupt flag.The interrupt can be configured to occur on either the rising, falling, or either edge of the $\overline{\text{RTS}}$ input. Write to bits RTEC[1:0] in the UART Control Register 2 (UCR2) to program which edge will cause an interrupt. If the bits are set to 00b and RTSEN = 1, the interrupt will occur on the rising edge (default). If the bits are set to 01b and RTSEN = 1, the interrupt will occur on the falling edge. If the bits are set to 1Xb and RTSEN = 1, the interrupt will occur on either edge.This is an asynchronous interrupt. The RTSF interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect. This flag can not cause an interrupt when uart clock is gated in STOP mode. So to wake up the MCU from STOP mode this interrupt can not be used.

Table 22-5. RTS Edge Triggered Interrupt Truth Table

RTS	RTSEN	RTEC[1]	RTEC[0]	RTSF	Interrupt occurs on:	WSCI_URT_MINT_UARTC_B
X	0	X	X	0	turned off	1

Table 22-5. RTS Edge Triggered Interrupt Truth Table (Continued)

RTS	RTSEN	RTEC[1]	RTEC[0]	RTSF	Interrupt occurs on:	WSCI_URT_MINT_UARTC_B
1->0	1	0	0	0	rising edge	1
0->1	1	0	0	1	rising edge	0
1->0	1	0	1	1	falling edge	0
0->1	1	0	1	0	falling edge	1
1->0	1	1	X	1	either edge	0
0->1	1	1	X	1	either edge	0

There is also another RTS interrupt which is not programmable but will assert the RTSD (RTS Delta) flag when the RTS pin changes states. This interrupt will assert the WSCI_URT_MINT_RTS_B output when RTSDEN = 1. This is an asynchronous interrupt. The RTSD interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect. This interrupt causes the MCU to come out of STOP mode.

22.7 DTR - Data Terminal Ready

This signal to the UART indicates the general readiness of the Data Terminal Equipment (DTE). If the connection between the UART and the DTE is established once, the DTR signal must remain active throughout the whole connection time. In general the DTR and DSR signals are responsible for establishing the connection. RTS and CTS are responsible for the data transfer and the transfer direction in the case of a half-duplex configuration. The DTR signal is like a “main switch”. If the DTR signal is inactive the RTS and CTS signals have no effect. The modem doesn’t respond in any way to the control signals (generally). This functionality is not implemented in hardware.

22.7.1 DTR Edge Triggered Interrupt

The DTR pin can be configured to cause an interrupt on a selectable edge. To enable the DTR pin to cause an interrupt, set the DTREN bit in the UART Control Register 3(UCR3). The DTRF interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect.

The interrupt can be configured to occur on either the rising, falling, or either edge of the DTR input. Write to the DTEC[1:0] bits in the UART Control Register 3 (UCR3) to program which edge will cause an interrupt. If the bits are set to 00b and DTREN = 1, the interrupt will occur on the rising edge (default). If the bits are set to 01b and DTREN = 1, the interrupt will occur on the falling edge. If the bits are set to 1Xb and DTREN = 1, the interrupt will occur on either edge. This is an asynchronous interrupt.

Table 22-6. DTR Edge Triggered Interrupt Truth Table

DTR	DTREN	DTEC[1]	DTEC[0]	DPSF	Interrupt occurs on:	WSCI_URT_MINT_UARTC_B
X	0	X	X	0	turned off	1
1->0	1	0	0	0	rising edge	1
0->1	1	0	0	1	rising edge	0
1->0	1	0	1	1	falling edge	0
0->1	1	0	1	0	falling edge	1
1->0	1	1	X	1	either edge	0
0->1	1	1	X	1	either edge	0

22.8 DSR - Data Set Ready

The UART uses this signal to inform the DTE that it is switched on, has completed all preparations and can communicate with the DTE.

22.9 DCD - Data Carrier Detect

The UART uses this signal to inform the DTE that it has detected the carrier signal and the connection will be set up. This signal remains active while the connection remains established.

22.10 RI - Ring Indicator

The UART uses this signal to inform the DTE that a ring just occurred.

22.11 $\overline{\text{CTS}}$ - Clear To Send

This output pin serves two purposes. Normally, the receiver indicates that it is ready to receive data by asserting this pin (low). When the CTS trigger level is programmed to trigger at 63 characters received and the receiver detects the valid start bit of the 64 character, it de-asserts this pin.

22.11.1 Programmable CTS de-assertion

The CTS pin can also be programmed to de-assert when the RX FIFO reaches a certain level. Setting the CTS trigger level at any value less than will de-assert the CTS pin upon the detection of the valid start bit of the N + 1 character (N being the trigger level setting). The receiver will continue to receive characters until the FIFO is full.

22.12 TXD - UART Transmit

This pin is the transmitter serial output. While in normal mode, NRZ data is output. While in Infrared mode, a 3/16 bit-period pulse is output for each “zero” bit transmitted and no pulse for each “one” bit transmitted. For RS-232 applications this pin must be connected to an RS-232 transmitter.

22.13 RXD - UART Receive

This pin is the receiver serial input. While in normal operation, NRZ data is expected. While in Infrared mode, a narrow pulse is expected for each “zero” bit received and no pulse for a “one” bit received. External circuitry must be used to convert the Infrared signal to an electrical signal. RS-232 applications need an external RS-232 receiver to convert voltage levels.

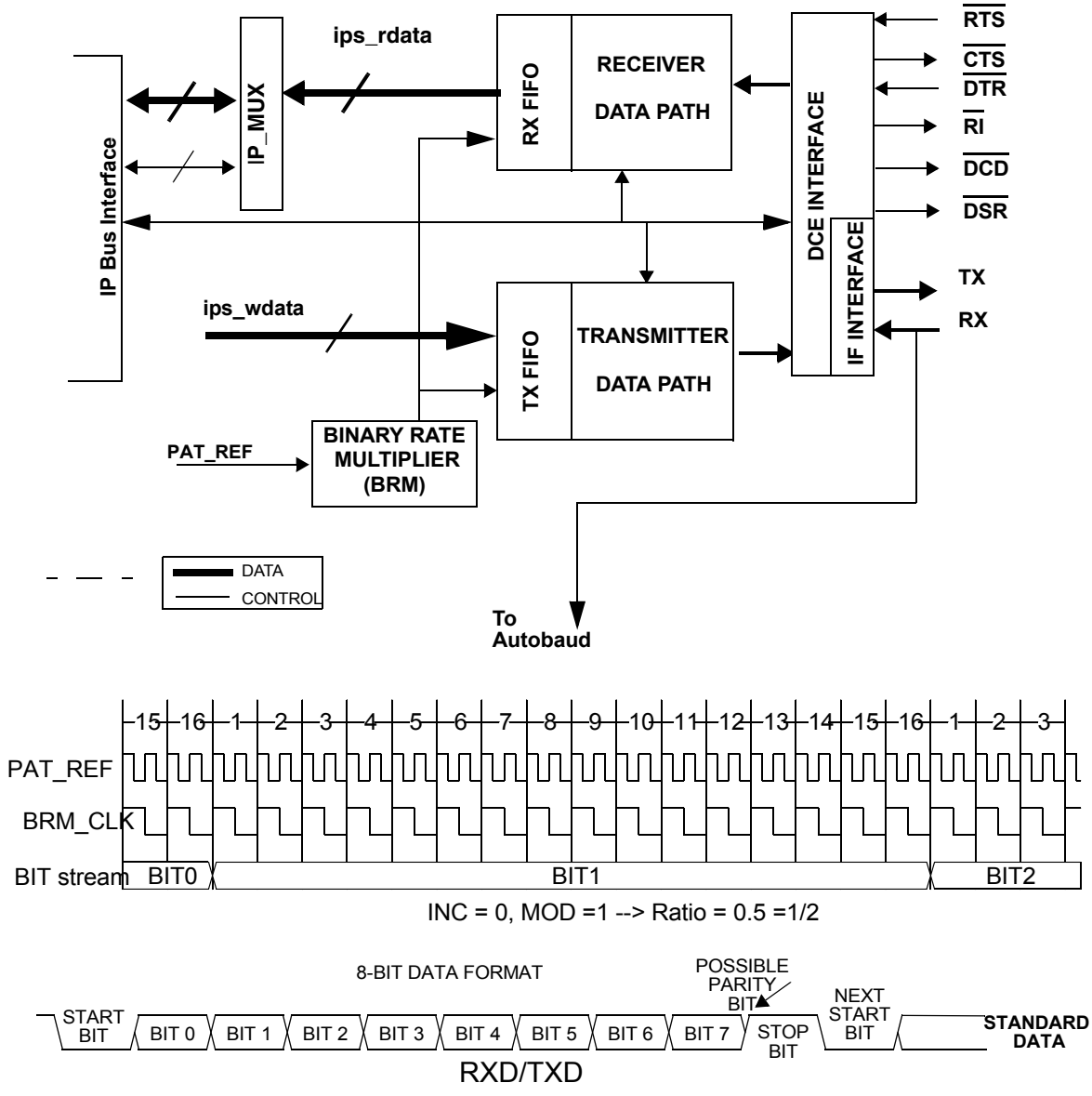


Figure 22-3. UART Block Diagram and Clock Generation Diagram

22.14 Peripheral Bus Interface

The UART interfaces to the bus via the IP bus interface. All accesses to the UART are asynchronous (except URXD register). All writes to registers are updated with respect to the rising edge of PDSN. All registers drive data asynchronously when a read is being performed, except the URXD register. When a read is being performed from the URXD register, data is driven on the bus synchronously using PAT_REF. Once address, module select, and read strobe are presented to the UART, data is present on the bus after the following rising edge of PAT_REF.

22.15 Sub-block Description

This UART module is easy to use from both a software perspective. There are four working registers that provide all status and control functions. A separate test register is provided for those applications that need it. The BRM registers control the UART bit rate.

There is also a Transmit register and a Receive register. The registers are optimized for a 16-bit bus. For example, all status bits associated with the received data are available along with the data in a single read. All register bits are readable (except TX DATA field in the UART Transmit Register) and most are read/write. There is a Baud Rate Count register for automatic baud rate detection. There are also two registers for the Escape Sequence detection. The UART contains four sub-modules. This section briefly describes the basic functionality of the four blocks.

22.16 Transmitter

The transmitter accepts a parallel character from the MCU and transmits it serially. The start, stop and parity (if enabled) bits are added to the character. $\overline{\text{RTS}}$ is used to flow-control the serial data if desired. If $\overline{\text{RTS}}$ is negated (high), the transmitter finishes sending the character in progress (if any) then stops and waits for $\overline{\text{RTS}}$ to again become asserted (low). A BREAK character (continuous 0's) can be generated. For debugging purposes, parity errors can be generated. The transmitter operates from the 1x clock provided by the BRM. Normal NRZ is transmitted when the Infrared Interface is disabled.

22.17 Transmitter FIFO Empty Interrupt Suppression

The Transmitter FIFO Empty Interrupt Suppression logic is to suppress the interrupt between writes to the Transmitter FIFO. When a character is written to the Transmitter FIFO it is transferred to the Transmitter Shift Register, after one clock period of transmitter clock. The suppression allows the software to write another character to the Transmitter FIFO before the interrupt is asserted. If one more character is not written to the Transmitter FIFO before the Transmitter Shift Register is empty, the interrupt will be asserted. When a single character is written to the Transmitter FIFO, the interrupt is released. The interrupt is asserted upon these conditions:

- Reset
- The Transmitter FIFO and the Transmitter Shift Register becomes empty before another character is written to the Transmitter FIFO.
- The Transmitter FIFO contains two or more characters and then becomes empty. See Figure 22-4.

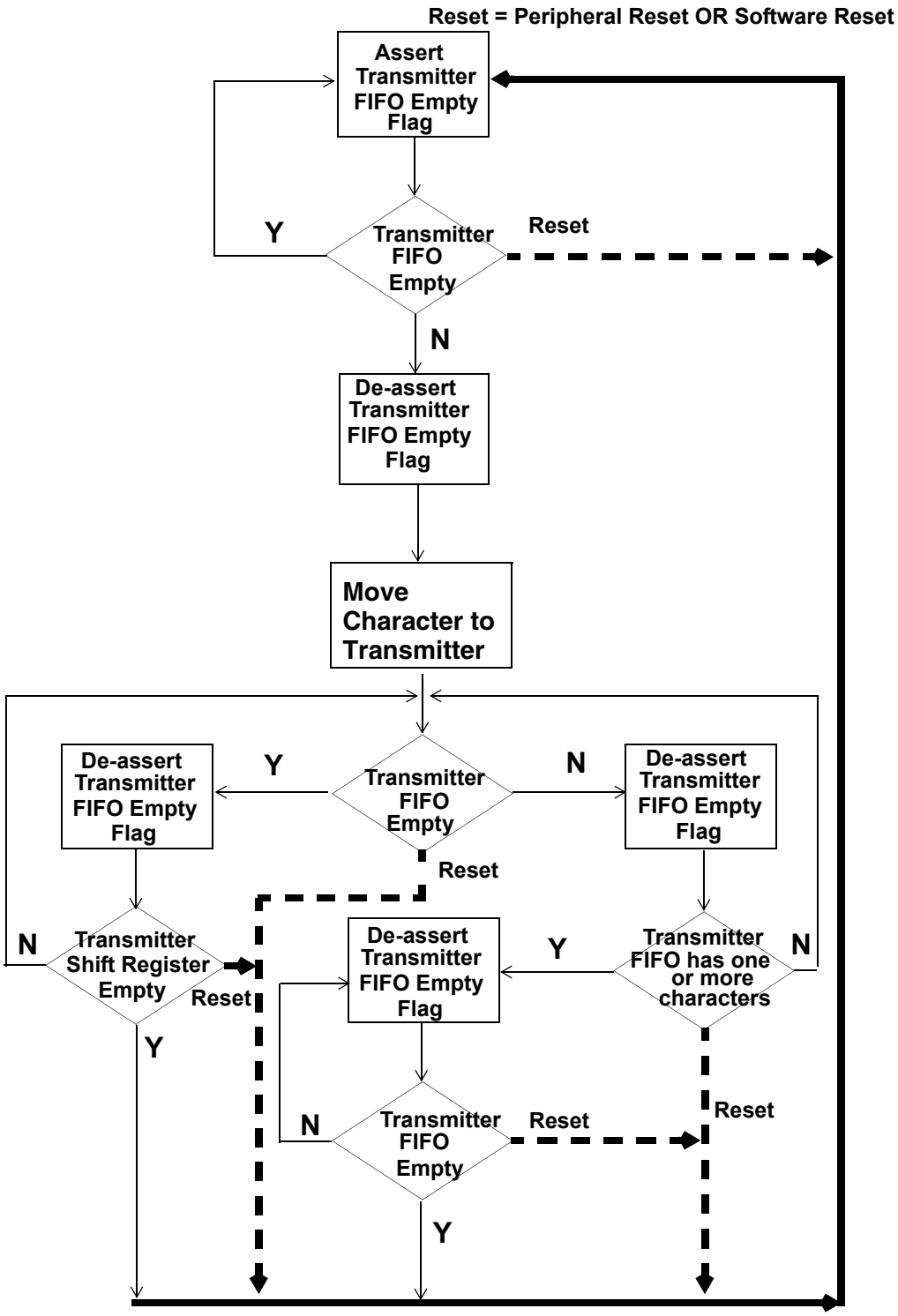


Figure 22-4. Transmitter FIFO Empty Interrupt Suppression Flow Chart

22.18 Receiver

The receiver accepts a serial data stream and converts it into a parallel character.When enabled, it searches for a start bit, qualifies it, then samples the succeeding data bits at the bit-center. Jitter tolerance and noise immunity are provided by sampling at a 16x rate and using voting techniques to clean up the samples. Once the start bit has been found, the data bits, parity bit (if enabled) and stop bits are shifted in. If parity is enabled, it is checked and its status is reported in the RX Register. Similarly, frame errors and breaks are checked and reported. When a new character is ready to be read by the host, Receive Data Ready (RDR) is asserted and an interrupt is posted (if DREN = 1). If the Receiver Trigger level is set to zero, the Receiver Ready Interrupt flag (RRDY) will be asserted as well and an interrupt is posted (if RRDYEN = 1). If the Receiver register is read as a 16-bit word and there is only one character in the FIFO, the interrupts are automatically cleared and the data along with four status bits are read by the MCU (see bit descriptions in the status register sections). The RRDY interrupt flag is cleared when the RX FIFO empties blow the programmed trigger level.

Normal NRZ is expected when the Infrared Interface is disabled.

22.18.1 Idle Line Detect

The Receiver Logic Block includes the ability to detect an idle line. This can be useful in a system to indicate the end or the beginning of a message. In order for an idle condition to occur there must be at least one word in the Receiver FIFO and the RXD pin has to be idle for more than a configured number of frames.

The detection of an idle condition can be configured in the UART Control Register 1 (UCR1) to flag an interrupt, if the IDEN bit is set and the line is idle for 4 (default), 8, 16, or 32 (max) frames. When an idle condition has been detected the IDLE bit is set in the UART Status Register 2 (USR2). The IDLE interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect.

22.18.1.1 Idle Condition Detect Configuration

The IDLE CONDITION DETECT (ICD[1:0]) is located in the UART Control Register 1. If the bits are set to 00b, then RXD has to be idle for more than 4 frames before IDLE is asserted. If the bits are set to 01b, then RXD has to be idle for more than 8 frames before IDLE is asserted. If the bits are set to 10b, then RXD has to be idle for more than 16 frames before IDLE is asserted. If the bits are set to 11b, then RXD has to be idle for more than 32 frames before IDLE is asserted (see Table 22-7 below).

Table 22-7. IDLE Detection Truth Table

IDEN	ICD[1]	ICD[0]	IDLE	WSCI_URT_MINT_RX_B
0	x	x	0	1
1	0	0	asserted after 4 idle frames	asserted after 4 idle frames
1	0	1	asserted after 8 idle frames	asserted after 8 idle frames
1	1	0	asserted after 16 idle frames	asserted after 16 idle frames
1	1	1	asserted after 32 idle frames	asserted after 32 idle frames

Note: Table 22-7 Note: This table assumes that no other interrupt is set at the same time this interrupt is set for the WSCI_URT_MINT_RX_B signal. This table shows how this interrupt affects the WSCI_URT_MINT_RX_B signal.

During a normal message there is no idle time between frames. If all information bits in a “Frame” were logic ones, the start bit would insure that at least one logic zero bit time has occurred for each frame and IDLE would not be asserted.

22.18.2 Receiver Wake Up

The Receiver Logic Block includes a Wake Up (WAKE) bit that is set if the receiver detects the start bit. WAKE is not set until the start bit has been qualified. If the Wake Up Interrupt bit (WKEN) is enabled, the receiver will flag an interrupt (WSCI_URT_MINT_UARTC_B = 0) and set the appropriate status bit in the status register. The WAKE interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect.

If the asynchronous WAKE interrupt is enable (AWAKEN = 1), the core is in STOP mode, and a falling edge is detected on the receive pin, this will assert AWAKE and WAKE the core from STOP mode. The AWAKE interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect.

If the asynchronous IR WAKE interrupt is enable (AIRINTEN = 1), the UART is configured for IR mode, the core is in STOP mode, and a falling edge is detected on the receive pin, this will assert AIRINT and WAKE the core from STOP mode. The AIRINT interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect.

Recommended procedure for programming the asynchronous interrupts is to first clear them by writing a one to the appropriate bit in the USR1. Poll or enable the interrupt for the receive not in progress status bit (RXDS). When asserted, this bit indicates to software that the receiver state machine is in the IDLE state, the next state is IDLE and the RXD pin is IDLE (HIGH). After following this procedure, enable the asynchronous interrupt and enter STOP mode.

NOTE:

The first character received after waking up from STOP or DOZE mode should be discarded. The Asynchronous Wake interrupt must not be used with the DOZE bit set. Use the WAKE interrupt instead.

22.18.3 Receiving a Break Condition

Receiving a Break Condition is detecting all zeros including a “zero” during the “stop bit” bit time. When a Break Condition is detected by the receiver, the Break Condition asserts BRCD in the UART Status Register (USR) and writes only the first Break Character to the RX FIFO. The BRCD interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect. BRCD will remain re-assert as long as a break condition exist. The break condition is ended when the receiver detects a 0 to 1 transition.

22.18.4 Vote Logic

The Vote Logic Block provides jitter tolerance and noise immunity by sampling with respect to BRM_CLK and using voting techniques to clean up the samples. The voting is implemented by sampling the incoming signal constantly on the rising edge of BRM_CLK. The receiver is provided with the majority vote value, which is two out of the three samples.

Table 22-8. Majority Vote Results

Samples	Vote
000	0

Table 22-8. Majority Vote Results (Continued)

Samples	Vote
101	1
001	0
111	1

The vote logic captures a sample on every rising edge of BRM_CLK, but the receiver takes its value in the middle of the sample frame using 16x oversampling (except for 13MHz reference frequency configured for 920kbps; 12x oversampling is used in this case). The idle character may be longer or shorter than 16 counts, but the receiver looks for a 1 to 0 transition. Then it starts to count the start bit but does not capture it in the FIFO. The start bit is validated upon receiving zeros for seven consecutive bit times following the 1 to 0 transition. Once the counter reaches F hex, it starts counting the next bit and captures it in the middle of the sampling frame, see Figure 22-6. All data bits are captured in the same manner. Once the stop bit is detected, the receiver shift register data is parallel shifted to the receiver FIFO.

There is a special case when the BRM_CLK period is longer than the period of the IR “zero” pulse. This special case is when the baud rate is configured for less than 31.25kbps. In this case, software would set the IRSC bit. When this bit is set, the patriot reference clock is used to clock the Voting Logic. The pulse is validated counting the length of the pulse. This logic will only work with 13MHz, 16.8MHz, and 19.44MHz reference clock frequencies. Enabling this bit with any other reference frequency is undefined.

22.19 Binary Rate Multiplier (BRM)

This submodule will be used to achieve all baud rates that required integer and non-integer division. The input and output frequency ratio is programmed in the UBIR and UBMR registers. The output frequency is divided by the input frequency to produce this ratio. For integer division set the UBIR = $0000 (hex) and write the divisor to the UBMR register. All values written to these registers should be one less than the actual value. This eliminates divides by zero, which is undefined, and increases the max range of the registers.

The BRM registers get updated when software writes to: (1) Both UBIR and UBMR registers. UBIR register must be written first, then the UBMR register. (2) Only to UBIR register, then the BRM clock output will update with the new UBIR register value and use the old UBMR register value. A write to the UBMR will be ignored if the UBIR was not written first.

Example Integer Division divide by 2:

Reference Frequency: 19.44MHz
UBIR = $0000 (hex); UBMR = $0001 (hex)
Note: Notice the values written to the registers were one less than the actual values.

Example Non Integer Division:

Reference Frequency: 19.44MHz
Output Frequency: 920kbps X 16 (sampling) = 14.72MHz
Ratio ==> 14.72 / 19.44 = 0.7572
UBIR = 7571 (decimal) = $1D93 (hex); UBMR = 9999(decimal) = $270F (hex)
Note: The ratio was derived directly from the division with no factoring (easiest). To derive the exact ratio, some factoring will need to be performed. Factoring the above ratio produces:
Factored Ratio: 184 / 243
UBIR = 183 (decimal) = $00B7 (hex); UBMR = 242 (decimal) = $00F2 (hex)

Example Non Integer Division:

Reference Frequency: 16.8MHz
Output Frequency: 920kbps X 16 (sampling) = 14.72MHz
Ratio ==> 14.72 / 16.8 = 0.8762
UBIR = 8761 (decimal) = $2239 (hex); UBMR = 9999 (decimal) = $270F (hex)
Note: The ratio was derived directly from the division with no factoring (easiest). To derive the exact ratio, some factoring will need to be performed. Factoring the above ratio produces:
Factored Ratio: 92 / 105
UBIR = 91 (decimal) = $005B (hex); UBMR = 104 (decimal) = $0068(hex)

Example Non Integer Division:

Reference Frequency: 13MHz
Output Frequency: 920kbps X 12 (sampling, special case) = 11.04MHz
Ratio ==> 11.04 / 13 = 0.8492
UBIR = 8491 (decimal) = $212B (hex); UBMR = 9999 (decimal) = $270F (hex)
Note: The ratio was derived directly from the division with no factoring (easiest). To derive the exact ratio, some factoring will need to be perform. Factoring the above ratio produces:
Factored Ratio: 276 / 325
UBIR = 275 (decimal) = $0113 (hex); UBMR = 324 (decimal) = $0144 (hex)

Baud Rate = (PAT_REF) / (16 x Divisor) or

Divisor = (PAT_REF) / (16 x Baud Rate)

Transmitter Clock = Baud Rate

Receiver Clock = PAT_REF / Divisor

22.19.1 Baud Rate Automatic Detection Logic

When the Baud Rate Automatic Detection Logic is enabled, the UART can lock onto the incoming baud rate. To enable this feature, set ADBR = 1 in the UART Control Register 1 (UCR1) and write a “one” ADET in the UART Status Register (USR) to clear it. When ADET = 0 and ADBR = 1, the UART automatically sets the INC = 0 and MOD = 0 (ratio = 1). Then it waits for the start bit (transition from 1 to 0) and tries to lock onto the incoming baud rate. Once the start bit has been detected, the length of the start bit is calculated by counting until the 0 to 1 transition (see Figure 22-5 and Section 22.19.1.1). Then the new Baud Rate is determined using this equation:

$$BaudRate = \left(\frac{Count}{16}\right)$$

Table 22-9. Baud Rate Automatic Detection

ADBR	ADET	Baud Rate Detection	WSCI_URT_MINT_UARTC_B
0	X	Manual configuration	1
1	0	Auto Detection	1
1	1	Auto Detection Complete	0

Note: Table 22-9 Note: This table assumes that no other interrupt is set at the same time this interrupt is set for the WSCI_URT_MINT_UARTC_B signal. This table shows how this interrupt affects the WSCI_URT_MINT_UARTC_B signal

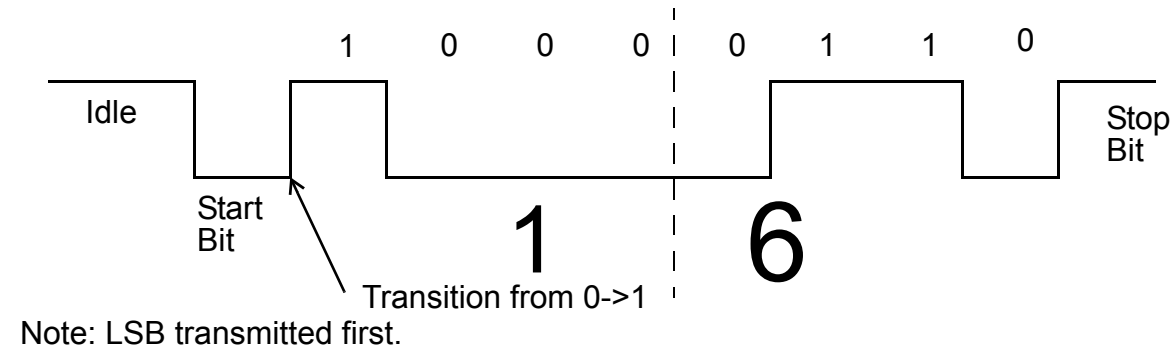


Figure 22-5. Baud Rate Detection Protocol Diagram

The BRM INC and MOD Preset registers are used when detecting the special baud rates. These registers are written to by software before the automatic baud sequence is started. After the auto baud count value has been divided by 16 and a remainder higher than three has been detected, the appropriate INC and MOD preset registers are selected, if BPEN is asserted. If BPEN is not asserted, integer division will be used for all detections and the remainder is ignored.

If the UART BRM registers are being written to simultaneously by the Baud Rate Automatic Detection Logic and Peripheral Data Bus, the Peripheral Data Bus has priority.

NOTE:

The baud rate automatic detection will fail to detect certain special baud rates unless the BRM preset registers are initialized. The special baud rates are 115.2 kbps (19.44 MHz reference frequency only),.230 kbps, 460 kbps and 920 kbps.

22.19.1.1 Baud Rate Automatic Detection Protocol

The Receiver must receive ASCII character “A” or “a” to verify that the incoming baud rate was detected properly. If ASCII character “A” or “a” was not received or an error occurred, the Baud Rate Divisor is reset to 1 and the auto detection sequence is repeated. If ASCII character “A” or “a” was received and no error occurred, the Automatic Detect Baud Rate bit is set (ADET = 1) and if enabled, an interrupt (WSCI_URT_MINT_UARTC = 0) is generated.

As long as ADET = 0 and ADBR = 1, the UART will continue to try to lock onto the incoming baud rate. Once the ASCII character “A” or “a” has been detected and the ADET bit is set, the receiver ignores the ADBR bit and continues normal operation with the calculated Baud Rate Divisor.

The UART interrupt will be active (WSCI_URT_MINT_UARTC = 0) as long as ADET = 1 and ADBR = 1. This can be disabled by clearing the Automatic Baud Rate Detection Interrupt bit (ADEN = 0). Before starting an Automatic Baud Rate Detection sequence, set ADET = 0 and ADBR = 1.

The Receive FIFO should contain the ASCII character “A” or “a” following the Automatic Baud Rate Detection Interrupt, which can be discarded by reading it. Since this UART has input clocks of 13 MHz, 16.8 MHz, and 19.44 MHz, the highest achievable baud rates are the PAT_REF divided by sixteen (PAT_REF / 16).

No Autobaud detection for 920kbps using 13MHz reference frequency.

Table 22-10. Highest Baud Rates

Reference Frequency (MHz)	Highest Baud Rate (bps)
13 MHz	812.5 Kbps
16.8 MHz	1.05 Mbps
19.44 MHz	1.215 Mbps

The 16 bit Baud Rate Count Register (UBRC) is reset to 8 and stays at hex FFFF in the case of an overflow. This register is used to count the start bit of the incoming baud rate. When the start bit has been detected and counted, the Baud Rate Count Register retains its value until the next Automatic Baud Rate Detection sequence has been initiated.

The read only Baud Rate Count Register counts only when auto detection is enabled.

22.19.2 Escape Sequence Detection

An escape “sequence” is typically made up of three escape “characters” entered in rapid succession (i.e. “+++”). Since these are valid characters by themselves, the time between characters is used to determine if it is a valid escape sequence. Too much time between two of the “+” characters would be interpreted as two “+” characters, and not part of an escape sequence.

The software would use this feature by choosing the escape character and writing its value to the escape character register. This value would be used in hardware for comparisons to incoming characters in the UART receive FIFO. When an escape character is detected, the escape character timer would start to count. The software would specify a time-out value for the maximum allowable time between escape characters. The Escape Timer is programmable in intervals of 2 msec up to a maximum of 8.192 seconds.

Table 22-11. Escape Timer Scaling

UTIM Register (Hex)	Time Between Specified Escape Characters
000	2 msec
001	4 msec
002	6 msec
003	8 msec
004	10 msec
.	.
.	.
.	.
0F8	498 msec
0F9	500 msec
.	.

Table 22-11. Escape Timer Scaling (Continued)

UTIM Register (Hex)	Time Between Specified Escape Characters
9C3	5 sec
.	.
FFD	8.188 sec
FFE	8.190 sec
FFF	8.192 sec

Note: Table 22-11: To calculate the time interval: [(UTIM_Value + 1) x 0.002 = Time_Interval]

Note: Example: (09C3 +1) x 0.002 = 5 sec.

This feature can only be used for 13MHz, 16.8MHz, and 19.44MHz reference frequencies. Enabling this feature with any other reference frequency is not supported and undefined. This feature can also be configured to assert an interrupt when an Escape Sequence is detected. When ESCI is enabled and an Escape Sequence is detected, the Escape Sequence flag (ESCF) is asserted. The ESCF interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect.

22.20 Infrared Interface

The Infrared Interface converts data to be transmitted or received as specified in the IRDA Serial Infrared Physical Layer Specification.

INVT = 0:
For each “zero” to be transmitted, a narrow negative pulse which is 3/16 of a bit time is generated. For each “one” to be transmitted no pulse is generated (output is high). This data is muxed out onto the TXD pin when IR mode is enable (IREN = 1). External circuitry has to be provided to drive an Infrared LED.

INVT = 1:
For each “zero” to be transmitted, a narrow positive pulse which is 3/16 of a bit time is generated. For each “one” to be transmitted no pulse is generated (output is low). This data is muxed out onto the TXD pin when IR mode is enable (IREN = 1). External circuitry has to be provided to drive an Infrared LED.

INVR = 0
When receiving, a narrow negative pulse is expected for each “zero” transmitted while no pulse is expected for each “one” transmitted (input is high). This data from the RXD pin is demuxed to the appropriate IR decoder logic when IR mode is enable (IREN = 1). Circuitry external to the IC transforms the Infrared signal to an electrical signal.

INVR = 1
When receiving, a narrow positive pulse is expected for each “zero” transmitted while no pulse is expected for each “one” transmitted (input is low). This data from the RXD pin is demuxed to the appropriate IR decoder logic when IR mode is enable (IREN = 1). Circuitry external to the IC transforms the Infrared signal to an electrical signal.

The SIR has an edge triggered interrupt (IRINT). This interrupt validates a zero bit being received. This interrupt is enabled by writing a “one” to ENIRI bit. The IRINT interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect.

22.21 UART Register Summary

This section describes the registers in the UART module. The Register Summary lists all registers of the module by ascending address. The absolute address of each register is given, and the value of each bit for reads and writes is given using the conventions in the Legend. The Legend table should be placed just before the Register Summary

Figure 0-3.

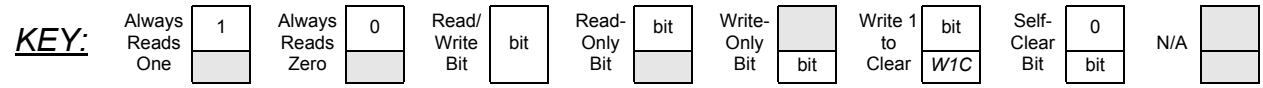


Table 22-12. UART Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
URXD UART1: ($2484_D000 + 4*n) UART2: ($2485_7000+4*n)	R	CHARRDY	ERR	OVRRUN	FRMERR	BRK	PRERR	0	0	RX_DATA[7:0]							
	W																
UTXD UART1: ($2484_D040 + 4*n) UART2 : $2485_7040+4*n)	R	0	0	0	0	0	0	0	0								
	W									TX_DATA[7:0]							
UCR1 UART1: ($2484_D080) UART2: ($2485_7080)	R	ADEN	ADBR	TRDYEN	IDEN	ICD[1:0]		RRDYEN	0	IREN	TXMPTYEN	RTSDEN	SNDBRK	0	0	DOZE	UARTEN
	W																
UCR2 UART1: ($2484_D082) UART2: ($2485_7082)	R	ESCI	IRTS	CTSC	CTS	ESCEN	RTEC[1:0]		PREN	PROE	STPB	WS	RTSEN	0	TXEN	RXEN	SRST_B
	W																
UCR3 UART1: ($2484_D084) UART2: ($2485_7084)	R	DTEC[1:0]		DTREN	0	0	DSR	DCD	RI	0	RXDSEN	AIRINTEN	AWAKEN	REF19	REF16	INVT	BPEN
	W																
UCR4 UART1: ($2484_D086) UART2: ($2485_7086)	R	CTSTL[5:0]						INVR	ENIRI	WKEN	REF13	IRSC	SAMP12	TCEN	BKEN	OREN	DREN
	W																
UFCR UART1: ($2484_D088) UART2: ($2485_7088)	R	TXTL[5:0]						0	0	0	0	RXTL[5:0]					
	W																
USR1 UART1:($2484_D08A) UART2:($2485_708A)	R	0	RTSS	TRDY	RTSD	ESCF	0	RRDY	0	0	RXDS	AIRINT	AWAKE	0	0	0	0
	W				W1C	W1C						W1C	W1C				

Table 22-12. UART Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USR2 UART1:($2484_D08C) UART2: ($2485_708C)	R	ADET	TXFE	DTRF	IDLE	0	0	0	IRINT	WAKE	0	0	RTSF	TXDC	BRCD	ORE	RDR
	W	W1C		W1C	W1C				W1C	W1C			W1C		W1C	W1C	
UESC UART1:($2484_D08E) UART2: ($2485_708E)	R	0	0	0	0	0	0	0	0	ESC_CHAR [7:0]							
	W																
UTIM UART1:($2484_D090) UART2: ($2485_7090)	R	0	0	0	0	TIM [11:0]											
	W																
UBIR UART1:($2484_D092) UART2: ($2485_7092)	R	INC [15:0]															
	W																
UBMR UART1:($2484_D094) UART2: ($2485_7094)	R	MOD [15:0]															
	W																
UBRC UART1:($2484_D096) UART2: ($2485_7096)	R	BCNT[15:0]															
	W																
BIPR1 UART1:($2484_D098) UART2: ($2485_7098)	R	INCP1 [15:0]															
	W																
BMPR1 UART1:($2484_D09A) UART2: ($2485_709A)	R	MODP1 [15:0]															
	W																
BIPR2 UART1:($2484_D09C) UART2: ($2485_709C)	R	INCP2 [15:0]															
	W																
BMPR2 UART1:($2484_D09E) UART2: ($2485_709E)	R	MODP2 [15:0]															
	W																
BIPR3 UART1:($2484_D0A0) UART2: ($2485_70A0)	R	INCP3 [15:0]															
	W																

Table 22-12. UART Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BMPR3 UART1:($2484_D0A2) UART2: ($2485_70A2)	R	MODP3 [15:0]															
	W																
BIPR4 UART1:($2484_D0A4) UART2: ($2485_70A4)	R	INCP4 [15:0]															
	W																
BMPR4 UART1:($2484_D0A6) UART2: ($2485_70A6)	R	MODP4 [15:0]															
	W																
UTS UART1:($2484_D0A8) UART2: ($2485_70A8)	R	0	RAMT	FRCPERR	LOOP	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Note: N = (0 - 15)

22.21.1 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various UART registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

This register is a 16-bit read-only register. This register contains the received character and its status. After Reset, if the Receiver block is enabled (RXEN = 1), until the first character is received this register contains random data and CHAR RDY bit is zero(0). The RX register is mapped to 16 word addresses in order to support Load Multiple Registers (LDM) instruction.

URXD UART Receiver Register

Addr (UART1): $2484_D000+4*n
(UART2): $2485_7000+4*n

BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
CHAR RDY	ERR	OVR RUN	FRME RR	BRK	PRER R			RX_DATA[7:0]							
TYPE: r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET: 0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?

Note: n = (0-15)

Table 22-13. URXD Description

Name	Description	Settings
CHARRDY Bit 15	**Character Ready** —This read-only bit indicates an invalid read when the FIFO becomes empty and software tries to read the same old data. This bit should not be used for polling for data written to the RX FIFO.	0 = character in RX_DATA field and associated flags are invalid 1 = character in RX_DATA field and associated flags valid and ready for reading.
ERR Bit 14	**Error detect**— This read-only bit indicates, while HIGH, that the character present in the RX_DATA field has an error status. The error can be an OVRRUN, FRMERR, BRK or PRERR. This bit is updated and valid for each received character.	0 = no error status detected 1 = error status detected
OVRRUN Bit 13	**Receiver Overrun**— This read-only bit indicates, while HIGH, that the receiver ignored data to prevent overwriting the data in the FIFO. Under normal circumstances, this bit should never be set. It indicates that the user's software is not keeping up with the incoming data rate. This bit is updated and valid for each received character. This bit is set for the last character written to the FIFO indicating that all characters following this character will be ignored if a read is not performed by software.	0 = no FIFO overrun 1 = a FIFO overrun was detected
FRMERR Bit 12	**Frame Error—**This read-only bit indicates, while HIGH, that the current character had a framing error (missing stop bit.) The data is possibly corrupted. This bit is updated for each character read from the FIFO.	0 = character has no framing error 1 = character has a framing error

Table 22-13. URXD Description (Continued)

Name	Description	Settings
BRK Bit 11	**Break Detect**—This read-only bit indicates, while HIGH, that the current character was detected as a BREAK. The data bits are all zero and the stop bit was also zero. The frame error bit will always be set when this bit is set. If odd parity is selected, Parity Error will also be set when this bit is set. This bit is valid for each character read from the FIFO.	0 = character is not a break character 1 = character is a break character
PRERR Bit 10	**Parity Error**—This read-only bit indicates, while HIGH, that the current character was detected with a parity error. The data is possibly corrupted. This bit is updated for each character read from the FIFO. While parity is disabled, this bit always reads zero.	0 = no parity error detected for data in RX_DATA field 1 = parity error detected for data in RX_DATA field
Bits 8-9	**Reserved** — Read as zero and written with zero for future compatibility.	N/A
RX_DATA[7:0] Bits 7-0	**Received Data**—These read-only bits are the received character. While in 7-bit mode, the MSB is forced to zero. While in 8-bit mode, all bits are active.	N/A

This register is a 16-bit write only register used by the MCU to write the data to be transmitted. The low-byte is write-only. If this register is read, bits TX[15:8] will always be read as zero(0), and TX[7:0] is not driving the bus and will be read as all zeros. The TX register is mapped to 16 word addresses in order to support Store Multiple Registers (STM) instruction.

UTXD UART Transmitter Register

Addr (UART1): $2484_D**040+4*n**
(UART2): $2485_7**040+4*n**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									TX_DATA[7:0]							
TYPE:	r	r	r	r	r	r	r	r	w	w	w	w	w	w	w	w
RESET:	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?

Note: n = (0-15)

Table 22-14. UTXD Description

Name	Description	Settings
Bits 15-8	**Reserved bits**— Read as zero should be written with zero for future compatibility.	N/A
TX_DATA[7:0] Bit 7-0	**Transmit Data**—These write-only bits are the parallel transmit data inputs. While in 7-bit mode D7 is ignored. While in 8-bit mode, all bits are used. Data is transmitted LSB first. A new character is transmitted when these bits are written.	N/A

The Control Register 1 is a read/write register. This register enables the UART and the Transmit and Receive Blocks. It controls the Tx and Rx FIFO levels and enables the TRDY and RRDY interrupts.

UCR1 UART Control Register 1 **Addr (UART1): $2484_D080 (UART2) : $2485_7080**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	ADEN	ADBR	TRDY EN	IDEN	ICD[1:0]		RRDY EN		IREN	TXMP TYEN	RTSD EN	SNDB RK			DOZE	UART EN
TYPE:	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw	r	r	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-15. UCR1 Description

Name	Description	Settings
ADEN Bit 15	**Automatic Baud Rate Detection Interrupt Enable**—When set, it enables the status bit ADET to cause an interrupt (WSCI_URT_MINT_UARTC_B = 0).	0 = Automatic Baud Rate Detection Interrupt disabled 1 = Automatic Baud Rate Detection Interrupt enabled
ADBR Bit 14	**Automatic Detection of Baud Rate**— When this bit is set (ADBR = 1) and the Automatic Baud Rate Detection Complete bit is cleared (ADET = 0), the receiver will detect the incoming baud rate automatically. The ADET flag is set (ADET = 1) when the receiver verifies the incoming baud rate has been detected properly by detecting an ASCII character “A” or “a” (hex $61 or $41).	0 = Automatic Detection of Baud Rate disabled 1 = Automatic Detection of Baud Rate enabled
TRDYEN Bit 13	**Transmitter Ready Interrupt Enable**— This active HIGH bit enables an interrupt when the transmitter has one or more slots available in the TX FIFO. The fill level in the TX FIFO at which an interrupt is generated is controlled by the TxFL bits. While this bit is negated, the transmitter interrupt is disabled	0 = Transmitter Ready interrupt disabled 1 = Transmitter Ready interrupt enabled
IDEN Bit 12	**Idle Condition Detected Interrupt Enable**—When set, it enables the status bit IDLE to cause an interrupt (WSCI_URT_MINT_RX_B = 0).	0 = Idle Condition Detected Interrupt disabled 1 = Idle Condition Detected Interrupt enabled
ICD[1:0] Bit 11-10	**Idle Condition Detection**—These two bits control the amount of frames RXD is allowed to be idle before reporting an Idle Condition.	00 = Idle for more than 4 frames 01 = Idle for more than 8 frames 10 = Idle for more than 16 frames 11 = Idle for more than 32 frames

Table 22-15. UCR1 Description (Continued)

Name	Description	Settings
RRDYEN Bit 9	**Receiver Ready Interrupt Enable**— This active HIGH bit enables an interrupt when the receiver has data in the RX FIFO. The fill level in the RX FIFO at which an interrupt is generated, is controlled by the RxFL bits. While negated, this interrupt is disabled.	0 = RX interrupt disabled 1 = RX interrupt enabled
Bit 8	**Reserved bit**—Read as zero and written with zero for future compatibility.	N/A
IREN Bit 7	**Infrared Interface Enable**— This active HIGH bit enables the Infrared interface. Refer to the Infrared Interface description in Section 22.20.	0 = Infrared interface disabled 1 = Infrared interface enabled
TXMPTYEN Bit 6	**Transmitter Empty Interrupt Enable**— This active HIGH bit enables a Transmitter FIFO Empty interrupt. While negated, this interrupt is disabled. The interrupt generated by the Transmitter FIFO Empty condition is logically OR-ed with the Transmitter Ready Interrupt and this OR condition is reflected by one bit only in the Interrupt Controller Registers.	0 = Transmitter FIFO Empty interrupt disabled 1 = Transmitter FIFO Empty interrupt enabled
RTSDEN Bit 5	**RTS Delta Interrupt Enable—**This active HIGH bit enables a RTS delta interrupt. While negated, this interrupt is disabled. The current status of the $\overline{\text{RTS}}$ pin is read in the UART Status Register.	0 = RTS interrupt disabled 1 = RTS interrupt enabled
SNDBRK Bit 4	**Send Break**—This bit forces the transmitter to send a BREAK character. The transmitter will finish sending the character in progress (if any) then send break until this bit is reset. The user is responsible to ensure that this bit is high for a sufficient period of time to generate a valid BREAK; the transmitter samples SNDBRK after every ***bit*** is transmitted. Following completion of the BREAK transmission, the UART will transmit two mark bits. The user can continue to fill the FIFO and any characters remaining will be transmitted when the BREAK is terminated.	0 = do not send break 1 = send break (continuous zeros)
Bits 2-3	**Reserved bits**—Read as zeros and written with zeros for future compatibility.	N/A
DOZE Bit 1	**DOZE** — When the CPU executes a **doze** instruction and the system is placed in the Doze State, the DOZE bit affects operation of the UART. While in the Doze State, if this bit is asserted, the UART is disabled. Refer to the description in Section 22.22, “UART Operation In Low-Power System States.” .	0 = UART enabled, while in Doze State 1 = UART disabled, while in Doze State

Table 22-15. UCR1 Description (Continued)

Name	Description	Settings
UARTEN Bit 0	**UART Enable**—This active HIGH bit enables the UART. While this bit is negated, the UART is disabled. While this bit is asserted the UART is enabled. If this bit is negated in the middle of a transmission, the transmitter stops and pulls the TXD line to logic one.	0 = UART disabled 1 = UART enabled

The Control Register 2 is a read/write register except for UCR2[3] bit which will always be read as zero(0). This register controls the overall operation of the UART. The user sets the BITSEL and its source, specifies the number of bits per character, enables or disables parity generation and checking, controls the $\overline{RTS}$ and $\overline{CTS}$ pins.

UCR2 UART Control Register 2 **Addr (UART1): $2484_D082 (UART2) : $2485_7082**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	ESCI	IRTS	CTSC	CTS	ESCEN	RTEC[1:0]		PREN	PROE	STPB	WS	RTSEN		TXEN	RXEN	SRST_B
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 22-16. UCR2 Description

Name	Description	Settings
ESCI Bit 15	**Escape Sequence Interrupt Enable**—When this bit is set, it will allow the Escape Sequence detection to cause an interrupt.	0 = Escape Sequence Interrupt disabled 1 = Escape Sequence Interrupt enabled
IRTS Bit 14	**Ignore $\overline{RTS}$ Pin**—This active HIGH bit forces the $\overline{RTS}$ input signal presented to the transmitter to always be asserted, effectively ignoring the external pin. While in this mode, the $\overline{RTS}$ pin can be used as a general purpose input.	0 = transmit only while $\overline{RTS}$ pin is asserted 1 = ignore $\overline{RTS}$ pin
CTSC Bit 13	**$\overline{CTS}$ Pin Control**—This active HIGH bit controls the operation of the $\overline{CTS}$ output pin. While this bit is asserted, the $\overline{CTS}$ output pin is controlled by the receiver. When the RX FIFO has been filled to the level of the programmed trigger level and the start bit of the overflowing character (TRIGGER LEVEL + 1) has been validated, the $\overline{CTS}$ output pin is negated to indicate to the far-end transmitter to stop transmitting. If the trigger level is programmed for less than 64, the receiver will continue to receive data until the FIFO is full. While the CTSC bit is negated, the $\overline{CTS}$ output pin is controlled by the CTS bit. On reset, since this bit is cleared to zero, the $\overline{CTS}$ pin is controlled by the CTS bit, which again is cleared to zero on reset. This means that on reset the $\overline{CTS}$ signal is negated.	0 = $\overline{CTS}$ pin controlled by the CTS bit 1 = $\overline{CTS}$ pin controlled by the receiver
CTS Bit 12	**CTS bit**—This bit controls the $\overline{CTS}$ pin while the CTSC bit is negated. While CTSC is asserted this bit has no function.	0 = $\overline{CTS}$ pin is high (inactive) 1 = $\overline{CTS}$ pin is low (active)
ESCEN Bit 11	**Escape Enable**—This bit will enable the Escape Sequence detection logic.	0 = Escape Sequence detection disabled 1 = Escape Sequence detection enabled

Table 22-16. UCR2 Description (Continued)

Name	Description	Settings
RTEC[1:0] Bit 10-9	**Request to Send Edge Control**—These two bits select the edge at which the RTS interrupt will be triggered. This has no effect on the RTS delta interrupt. These bit only have an effect when RTSEN = 1 (see Table 22-5).	00 = trigger interrupt on rising edge 01 = trigger interrupt on falling edge 1X = trigger interrupt on any edge
PREN Bit 8	**Parity Enable**—This active HIGH bit controls the parity generator in the transmitter and parity checker in the receiver. While asserted, they are enabled. While negated, they are disabled.	0 = parity disabled 1 = parity enabled
PROE Bit 7	**Parity Odd/Even**—This bit controls the sense of the parity generator and checker. While HIGH, odd parity is generated and expected. While LOW, even parity is generated and expected. This bit has no function if PREN is low.	0 = even parity 1 = odd parity
STPB Bit 6	**Stop Bits**—This bit controls the number of stop bits transmitted after a character. While HIGH, two stop bits are sent. While LOW, one stop bit is sent. This bit has no effect on the receiver which expects one or more stop bits.	0 = 1 stop bit transmitted. 1 = 2 stop bits transmitted.
WS Bit 5	**Word Size**— This bit controls the character length. While HIGH, the transmitter and receiver are in eight bit mode. While low, they are in seven-bit mode. The transmitter then ignores B7 and the receiver sets B7 to zero. This bit can be changed in-between transmission (reception) of characters, but not while a transmission (reception) is in progress, in which case the length of the current character being transmitted (received) is unpredictable.	0 = 7 bit transmit and receive character length (not including START, STOP or PARITY bits). 1 = 8 bit transmit and receive character length (not including START, STOP or PARITY bits).
RTSEN Bit 4	**Request to Send Interrupt Enable—**This bit controls the RTS edge sensitive interrupt. When this bit is asserted, an interrupt will be asserted when the programmed edge is detected on the $\overline{\text{RTS}}$ pin and the RTSF bit is asserted (see Table 22-5). When this bit is cleared, the interrupt will be masked.	0 = Request to Send Interrupt Disabled 1 = Request to Send Interrupt Enabled
Bit 3	**Reserved bit**—Read as zero and written with zero for future compatibility.	N/A
TXEN Bit 2	**Transmitter Enable**—This active HIGH bit enables the transmitter. While this bit is negated the transmitter is disabled and idle. While UARTEN and TXEN bits are set the transmitter is enabled. If this bit is negated in the middle of a transmission, the UART disables the transmitter immediately, and starts marking ones. The transmitter FIFO cannot be written to when this bit is cleared.	0 = transmitter disabled 1 = transmitter enabled

Table 22-16. UCR2 Description (Continued)

Name	Description	Settings
RXEN Bit 1	**Receiver Enable**— This active HIGH bit enables the receiver.When the receiver is enabled, if the RXD line is already low, the receiver does not recognize break characters, since it requires a valid one-to-zero transition before it can accept any character. If disabled during a reception, the receiver will complete the current reception then disable.	0 = receiver disabled 1 = receiver enabled
SRST_B Bit 0	**Software Reset**—The software reset will reset the transmitter and receiver state machines and FIFOs. All Status Registers will be reset. Once software writes a zero to this bit, software reset will remain active for four clock cycles of PAT_REF. Hardware will de-assert this bit after four clock cycles of PAT_REF. Software can only write a zero to this bit. Writing a one to this bit is ignored.	0 = Reset UART module 1 = No Reset

The Control Register 3 is a read/write register. This register controls the overall operation of the UART.

UCR3 UART Control Register 3 **Addr (UART1): $2484_D084 (UART2) : $2485_7084**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	DTEC[1:0]		DTREN			DSR	DCD	RI		RXDSEN	AIRINTEN	AWAKEN	REF19	REF16	INVT	BPEN
TYPE:	rw	rw	rw	r	r	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-17. UCR3 Description

Name	Description	Settings
DTEC[1:0] Bit 15-14	**DTR Interrupt Edge Control**—These bits control the edge on which an interrupt will be generated. An interrupt will only be generated if the DTREN bit is set.	00 = interrupt generated on rising edge 01 = interrupt generated on falling edge 1X = interrupt generated on either edge
DTREN Bit 13	**Data Terminal Ready Interrupt Enable**—This bit controls the DTR edge sensitive interrupt. When this bit is asserted, an interrupt will be asserted when the programmed edge is detected on the $\overline{DTR}$ pin and the DTRF bit is asserted (see Table 34.7-1). When this bit is cleared, the interrupt will be masked.	0 = Data Terminal Ready Interrupt Disabled 1 = Data Terminal Ready Interrupt Enabled
Bit 12-11	**Reserved**—Read as zeros and written with zeros for future compatibility.	N/A
DSR Bit 10	**Data Set Ready**—This bit is used by software to control the $\overline{DSR}$ pin for the modem interface.	0 = $\overline{DSR}$ pin is logic zero 1 = $\overline{DSR}$ pin is logic one
DCD Bit 9	**Data Carrier Detect**—This bit is used by software to control the $\overline{DCD}$ pin for the modem interface.	0 = $\overline{DCD}$ pin is logic zero 1 = $\overline{DCD}$ pin is logic one
RI Bit 8	**Ring Indicator**—This bit is used by software to control the $\overline{RI}$ pin for the modem interface.	0 = $\overline{RI}$ pin is logic zero 1 = $\overline{RI}$ pin is logic one
Bit 7	**Reserved—**Read as zeros and written with zeros for future compatibility.	N/A
RXDSEN Bit 6	**Receive Status Interrupt Enable** -- This bit controls the Receive Status interrupt. When this bit is asserted, the receiver state machine is in a IDLE state, next state is IDLE, and the receive pin is HIGH, an interrupt will be generated.	0 = RXDS interrupt disabled 1 = RXDS interrupt enabled

Table 22-17. UCR3 Description (Continued)

Name	Description	Settings
AIRINTEN Bit 5	**Asynchronous IR WAKE Interrupt Enable** -- This bit controls the asynchronous IR WAKE interrupt. When this bit is asserted and a pulse is detected on the RXD pin, an interrupt will be generated.	0 = AIRINT interrupt disabled 1 = AIRINT interrupt enabled
AWAKE Bit 4	**Asynchronous WAKE Interrupt Enable** -- This bit controls the asynchronous WAKE interrupt. When this bit is asserted and a falling edge is detected on the RXD pin, an interrupt will be generated.	0 = AWAKE interrupt disabled 1 = AWAKE interrupt enabled
REF 19 Bit 3	**Reference Frequency 19.44MHz**—This bit indicates to the hardware that a reference clock of 19.44MHz is being used.When this bit is set, all of the preset registers are used during Auto Baud Detection (if BPEN is asserted).	0 = 19.44MHz reference clock not being used. 1 = 19.44MHz reference clock being used.
REF16 Bit 2	**Reference Frequency 16.8MHz**—This bit indicates to the hardware that a reference clock of 16.8MHz is being used. When this bit is set, the INCP4 and MODP4 registers are disabled.	0 = 16.8MHz reference clock not being used. 1 = 16.8MHz reference clock being used
INVT Bit 1	**Inverted Infrared Transmission**— When cleared, the Infrared Logic Block transmits an active low or negative infrared 3/16 pulse for all zeros. Ones are transmitted for “ones”. When this bit is set (INVT = 1), the Infrared Logic Block expects an active high or positive infrared 3/16 pulse for all zeros. Zeros are transmitted for “ones”.	0 = Active Low transmission 1 = Active High transmission
BPEN Bit 0	**Preset Registers Enable**—This bit activates the BRM Preset Registers for use during Automatic Baud Rate Detection mode. When this bit is de-asserted, the remainder from the Automatic Baud Count Register divided by 16 is ignored. When this bit is asserted, the remainder and the dividend chooses the appropriate preset register upon the detection of special baud rates. If the criteria is not met for selecting one of the preset registers, integer division is performed by writing the dividend to the UBMR register. **Note:** Automatic detection of 920kbps at 13MHz is not implemented.	0 = BRM Preset Registers not used--normal operation 1 = BRM Preset Registers are used--auto detecting special baud rates

The Control Register 4 is a read/write register. This register controls the operation of other UART interrupts.

UCR4 UART Control Register 4 **Addr (UART1): $2484_D086 (UART2) : $2485_7086**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	CTSTL[5:0]						INVR	ENIRI	WKEN	REF13	IRSC	SAMP12	TCEN	BKEN	OREN	DREN
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Table 22-18. UCR4 Description

Name	Description	Settings
CTSTL[5:0] Bits 15-10	**CTS Trigger Level**—These bits control the threshold at which the CTS pin will be de-asserted by the Receiver FIFO. Although the trigger level has been reached and the CTS pin has been de-asserted, the receive FIFO will continue to receive data until the FIFO becomes full. The bits are encoded as follows.	**Bits** / **Description** 000000 — Zero characters received (min) 000001 — One character have been written to the Receive Buffer (default) . . . 111110 — 62 characters have been written to the Receive Buffer 111111 — 63 characters have been written to the Receive Buffer
INVR Bit 9	**Inverted Infrared Reception**—When cleared, the Infrared Logic Block expects an active low or negative infrared 3/16 pulse. Ones are expected for “ones”. When this bit is set (INVR = 1), the Infrared Logic Block expects an active high or positive infrared 3/16 pulse. Zeros are expected for “ones”.	0 = Active Low detection 1 = Active High detection
ENIRI Bit 8	**Serial Infrared Interrupt Enable**—This bit enables the Serial Infrared Interrupt.	0 = Serial Infrared Interrupt disabled 1 = Serial Infrared Interrupt enabled

Table 22-18. UCR4 Description (Continued)

Name	Description	Settings
WKEN Bit 7	**Wake Up Interrupt Enable**—When set, it enables the status bit WAKE in the USR to cause an interrupt. The WAKE bit is set by the receiver at the detection of a start bit by the receiver.	0 = Wake Up Interrupt disabled 1 = Wake Up Interrupt enabled
REF13 Bit 6	**Reference Frequency 13 MHz**—This bit indicates to the hardware that a reference clock of 13MHz is being used.This bit disables the 920kbps preset register for the Automatic Baud Rate Detection logic for the 13MHz reference frequency.	0 = 920kbps preset register is enabled--normal operation 1 = 13MHz reference clock--920kbps preset register is disabled
IRSC Bit 5	**IR Special Case**—When set, it switches the Vote Logic clock from the Sampling clock to the Patriot Reference clock. The IR pulses are counted a predetermined amount of time depending on the reference frequency.	0 = Vote Logic uses Sampling clock (16 x baud rate) - normal operation 1 = Vote Logic uses Patriot Reference clock
SAMP12 Bit 4	**Sample 12x—**This bit is used to choose 12x oversampling for the special case of 920kbps baud rate using the 13MHz reference frequency. Note: This bit is not reset by a software reset.	0 = 12x oversampling disabled (normal 16x oversampling) 1 = 12x oversampling enabled
TCEN Bit 3	**Transmitter Complete Interrupt Enable**—When set, it enables the status bit TXDC to cause an interrupt (WSCI_URT_MINT_TX_B = 0).	0 = Transmitter Complete Interrupt disabled 1 = Transmitter Complete Interrupt enabled
BKEN Bit 2	**Break Condition Detected Interrupt Enable**—When set, it enables the status bit BRCD to cause an interrupt (WSCI_URT_MINT_UARTC_B = 0).	0 = Break Condition Detected Interrupt disabled 1 = Break Condition Detected Interrupt enable
OREN Bit 1	**Receiver Overrun Interrupt Enable**— When set, it enables the status bit ORE to cause an interrupt (WSCI_URT_MINT_UARTC_B = 0).	0 = Overrun Error Interrupt disabled 1 = Overrun Error Interrupt enabled
DREN Bit 0	**Receive Data Ready Interrupt Enable**—When set, it enables the status bit RDR to cause an interrupt (WSCI_URT_MINT_RX_B = 0).	0 = Receive Data Ready Interrupt disabled 1 = Receive Data Ready Interrupt enabled

The FIFO Control Register is a read/write register. This register controls the operation of the UART FIFO trigger levels and interrupts.

UFCR UART FIFO Control Register

Addr (UART1): $2484_D088
(UART2): $2485_7088

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	TXTL[5:0]										RXTL[5:0]					
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1

Table 22-19. UFCR Description

Name	Description	Settings
TXTL[5:0] Bit 15-10	**Transmitter Trigger Level**—These bits control the threshold at which a maskable interrupt will be generated by the Transmitter FIFO. A maskable interrupt will be generated whenever the data level in the Transmitter FIFO falls below the selected threshold. The bits are encoded as follows.	**Bits** / **Description** 000000 — Reserved 000001 — Reserved 000010 — There are less than 2 characters in the Transmitter FIFO . . . 111110 — There are less than 62 characters in the Transmitter FIFO 111111 — There are less than 63 characters in the Transmitter FIFO
Bit 9-6	**Reserved bits**— Read as zero and written with zero for future compatibility.	N/A

Table 22-19. UFCR Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>RXTL[5:0]
Bit 5-0</td><td>Receiver Trigger Level—These bits control the threshold at which a maskable interrupt will be generated by the Receiver FIFO. A maskable interrupt will be generated whenever the data level in the Receiver FIFO reaches the selected threshold. The bits are encoded as follows</td><td>
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>000000</td><td>Zero characters received (min)</td></tr>
<tr><td>000001</td><td>One character have been written to the Receive Buffer (default)</td></tr>
<tr><td>.
.
.</td><td>.
.
.</td></tr>
<tr><td>111110</td><td>62 characters have been written to the Receive Buffer</td></tr>
<tr><td>111111</td><td>63 characters have been written to the Receive Buffer</td></tr>
</table>
</td></tr>
</table>

The UART Status Register 1 is a 16-bit read/write register that represents reports of errors and status flags.

USR1 UART Status Register 1 **Addr (UART1): $2484_D08A (UART2): $2485_708A**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
		RTSS	TRDY	RTSD	ESCF		RRDY			RXDS	AIRINT	AWAKE				
TYPE:	r	r	r	w1c	w1c	r	r	r	r	r	w1c	w1c	r	r	r	r
RESET:	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-20. USR1 Description

Name	Description	Settings
Bit 15	**Reserved**—Read as zero and written with zero for future compatibility.	N/A
RTSS Bit 14	**$\overline{\text{RTS}}$ Pin Status**—This bit indicates the current status of the $\overline{\text{RTS}}$ pin. A “snapshot” of the pin is taken immediately before this bit is presented to the data bus. Can not be cleared. All writes to this bit is ignored. At Reset, this bit is set to zero(0).	0 = $\overline{\text{RTS}}$ pin is high (inactive) 1 = $\overline{\text{RTS}}$ pin is low (active)
TRDY Bit 13	**Transmitter Ready Interrupt Flag**— This bit indicates that the TX FIFO has emptied below its target threshold and needs data. This bit is automatically cleared when the data level in the TX FIFO goes beyond the set threshold level by TxFL bits. At Reset, this bit is set to one(1).	0 = transmitter does not need data 1 = transmitter needs data (interrupt posted)
RTSD Bit 12	**RTS Delta—**This bit indicates, while HIGH, that the $\overline{\text{RTS}}$ pin changed state. It generates a maskable interrupt. When in STOP mode, only RTS assertion will set this bit and wake the M·CORE. The current state of the $\overline{\text{RTS}}$ pin is available on the RTSS bit. The RTSD interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect. At Reset, this bit is set to zero(0).	0 = $\overline{\text{RTS}}$ pin did not change state since last cleared. 1 = $\overline{\text{RTS}}$ pin changed state (write a “one” to clear).
ESCF Bit 11	**Escape Sequence Interrupt Flag**—This bit is asserted, when ESCI bit is set and an escape sequence has been detected in the receive FIFO. The ESCF interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect.	0 = no escape sequence detected 1 = escape sequence detected (write a “one” to clear)
Bit 10	**Reserved bit**—Read as zero and written with zero for future compatibility.	N/A

Table 22-20. USR1 Description (Continued)

Name	Description	Settings
RRDY Bit 9	**Receiver Ready Interrupt Flag**—This active HIGH bit indicates that the receive FIFO data level is above that specified by RxFL bits threshold. When asserted, it generates a maskable interrupt. Refer to the RxFL bits description for setting the interrupt threshold. In conjunction with the CHARRDY bit in URXD, host software can continue to read the RX FIFO in an interrupt service routine until the RX FIFO is empty. This bit is automatically cleared when data level in the RX FIFO goes below the set threshold level. At Reset, this bit is set to zero(0).	0 = no character ready 1 = character(s) ready (interrupt posted)
Bit 8-7	**Reserved bits**—Read as zeros and written with zeros for future compatibility.	N/A
RXDS Bit 6	**Receiver IDLE Interrupt Flag** -- This active HIGH bit indicates that the receiver state machine is in a IDLE state, next state is IDLE, and the receive pin is HIGH. This flag is automatically cleared when a character is being received. This flag indicates the IDLE state only when the UART enable (UCR1 UARTEN) and Receiver Enable (UCR2 RXEN) bits are asserted.	0 = Receive in progress 1 = Receiver IDLE
AIRINT Bit 5	**Asynchronous IR WAKE Interrupt Flag** -- This active HIGH bit indicates that the IR WAKE pulse was detected. The AIRINT interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect.	0 = no pulse detected RXD pin 1 = pulse detected RXD pin
AWAKE Bit 4	**Asynchronous WAKE Interrupt Flag** -- This active HIGH bit indicates that a falling edge was detected on the RXD pin. The AWAKE interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect.	0 = no falling edge detected RXD pin 1 = falling edge detected on RXD pin
Bit 3-0	**Reserved bits**—Read as zeros and written with zeros for future compatibility.	N/A

The UART Status Register 2 is a 16-bit register that represents reports of errors and status flags.

USR2 UART Status Register 2 **(UART1):** $2484_D**08C** **(UART2):** $2485_7**08C**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	ADET	TXFE	DTRF	IDLE				IRINT	WAKE			RTSF	TXDC	BRCD	ORE	RDR
TYPE:	w1c	r	w1c	w1c	r	r	r	w1c	w1c	r	r	w1c	r	w1c	w1c	r
RESET:	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Table 22-21. USR2 Description

Name	Description	Settings
ADET Bit 15	**Automatic Baud Rate Detect Complete**—This bit indicates that an “A” or “a” has been received. This means that the receiver has detected and verified the incoming baud rate. The ADET interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect.	0 = ASCII “A” or “a” has not been received 1 = ASCII “A” or “a” has been received (write a “one” to clear)
TXFE Bit 14	**Transmitter Buffer Empty**—This read only bit indicates that the Transmitter Buffer is empty. This flag is cleared automatically cleared when data is written to the TX FIFO.	0 = Transmitter Buffer not empty 1 = Transmitter Buffer Empty
DTRF Bit 13	**DTR edge triggered interrupt flag**—This bit is asserted when the programmed edge is detected on the $\overline{\text{DTR}}$ pin. The DTEC bits select the edge which causes an interrupt (see Table 21-5). The DTRF interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect.	0 = Programmed edge not detected on $\overline{\text{DTR}}$ 1 = Programmed edge detected on $\overline{\text{DTR}}$ (write a “one” to clear)
IDLE Bit 12	**Idle Condition**—This bit reports the status of an Idle Condition for more than a programmed amount frames, see Section 22.18.1.1. The **IDLE** interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect.	0 = No Idle Condition Detected 1 = Idle Condition Detected (write a “one” to clear)
Bit 11-9	**Reserved bits**—Read as zeros and written with zeros for future compatibility.	N/A
IRINT Bit 8	**Serial Infrared Interrupt Flag**—When an edge is detected on the RX pin during SIR Mode, this flag will be asserted. When ENIRI is asserted and IRINT is asserted, this will assert the common interrupt, WSCI_URT_MINT_UARTC_B. The **IRINT** interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect.	0 = no edge detected 1 = valid edge detected (write a “one” to clear)

Table 22-21. USR2 Description (Continued)

Name	Description	Settings
WAKE Bit 7	**Wake Up**—This bit indicates the start bit has been detected. The **WAKE** interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect.	0 = Start bit not detected 1 = Start bit detected (write a “one” to clear)
Bit 6-5	**Reserved bits**— Read as zeros and written with zeros for future compatibility	N/A
RTSF Bit 4	**RTS edge triggered interrupt flag—**This bit is asserted when the programmed edge is detected on the $\overline{\text{RTS}}$ pin. The RTEC bits select the edge which causes an interrupt (see Table 22-5. The RTSF interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect.	0 = Programmed edge not detected on $\overline{\text{RTS}}$ 1 = Programmed edge detected on $\overline{\text{RTS}}$ (write a “one” to clear)
TXDC Bit 3	**Transmitter Complete**—This read only bit indicates that the Transmitter Buffer and Shift Register is empty; therefore the transmission is complete. This flag is cleared automatically cleared when data is written to the TX FIFO.	0 = Transmit Incomplete 1 = Transmit Complete
BRCD Bit 2	**Break Condition Detected**—This bit indicates that a Break Condition was detected by the Receiver. The **BRCD** interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect.	0 = No Break Condition Detected 1 = Break Condition Detected (write a “one” to clear)
ORE Bit 1	**Over Run Error**— This bit indicates that the Receiver Buffer was full when data was being received. The **ORE** interrupt is cleared by writing a one(1) to this bit. Writing zero(0) to this bit has no effect.	0 = No Over Run Error 1 = Over Run Error (write a “one” to clear)
RDR Bit 0	**Receive Data Ready**—This bit indicates at least one character has been received and written to the Receive FIFO. If the Receiver register is read and there is only one character in the FIFO, this flag is automatically cleared.	0 = No Receive Data 1 = Receive Data Ready

UECS UART Escape Character Register

Addr (UART1): $2484_D**08E**
(UART2) : $2485_7**08E**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									ESC_CHAR[7:0]							
TYPE:	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1

Table 22-22. UECS Description

Name	Description	Settings
Bits 15-8	Reserved	N/A
ESC_CHAR [7:0] Bits 7:0	**UART Escape Character Register—**Software writes the escape character in this register and it is compared to all the received characters to detect an escape sequence.	N/A

UTIM — UART Escape Timer Register

Addr (UART1): $2484_D090 (UART2) : $2485_7090

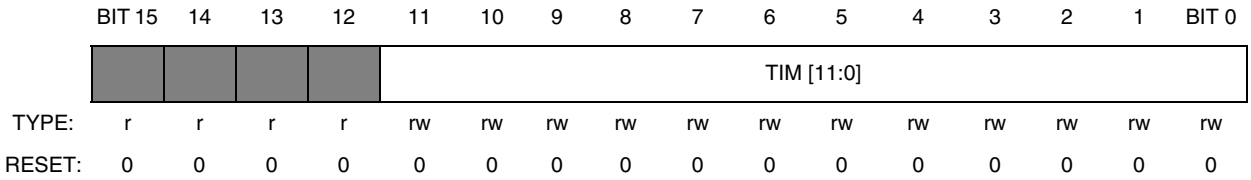


Table 22-23. UTIM Description

Name	Description	Settings
Bits 15-12	Reserved	N/A
TIM[11:0] Bits 11-0	**UART Escape Timer Register—**Software writes to this register to configure the maximum interval between escape characters. This register is scalable in intervals of 2msec up to a maximum of 8.192 sec.	N/A

UBIR UART BRM INC Register

Addr (UART1): $2484_D092 (UART2) : $2485_7092

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	INC [15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-24. UBIR Description

Name	Description	Settings
INC[15:0]	**BRM INC Register—**This register is a read/write register that holds the numerator value minus one of the BRM ratio (see Section 22.19). This register can be written to at any time. Hardware will update the BRM MOD value at the appropriate time to avoid glitches on the BRM_CLK output (sampling clock). The BRM will not be updated until both register have been written to by software. If only one register is written to by software, the BRM will ignore this data until the other register is written to by software. Updating this register using byte accesses is not recommended and undefined.	N/A

UBMR UART BRM MOD Register

Addr (UART1): $2484_D094
(UART2): $2485_7094

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	MOD [15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-25. UBMR Description

Name	Description	Settings
MOD[15:0]	**BRM MOD Register—**This register is a read/write register that holds the value of the denominator minus one of the BRM ratio (see Section 22.19). This register can be written to at any time. Hardware will update the BRM MOD value at the appropriate time to avoid glitches on the BRM_CLK output (sampling clock). The BRM will not be updated until both register have been written to by software. If only one register is written to by software, the BRM will ignore this data until the other register is written to by software. Updating this register using byte accesses is not recommended and undefined.	N/A

UBRC UART Baud Rate Count Register

Addr (UART1): $2484_D**096**
(UART2): $2485_7**096**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	BCNT [15:0]															
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Table 22-26. UBRC Description

Name	Description	Settings
BCNT[15:0]	**Baud Rate Count Register—**This read only register is used to count the start bit of the incoming baud rate. When the start bit has been detected and counted, the Baud Rate Count Register retains its value until the next Automatic Baud Rate Detection sequence has been initiated. The 16 bit Baud Rate Count register is reset to 8 and stays at hex FFFF in the case of an overflow. This register is used to count or measure the length of the incoming baud rate start bit when in Automatic Baud Rate Detection mode. This register is clocked by the BRM_CLK.	N/A

These registers are used in conjunction with the automatic baud detection logic. These registers are assigned preset values for the special baud rates 920kbps, 460kbps, 230kbps, and 115.2kbps. When the BPEN bit is asserted these registers are used upon the detection of a remainder. These registers must be written to prior to enabling the Auto Baud Detection logic. This feature only supports 13MHz, 16.8MHz, and 19.44MHz reference frequencies. Enabling this feature with other reference frequencies is not supported and undefined.

Addr (UART1):
BIPR1: $2484_D098
BIPR2:$ $2484_D09C
BIPR3: $2484_D0A0
BIPR4: $2484_D0A4

BIPRi — UART BRM INC Preset Register i

(UART2) :
BIPR1 : $2485_7098
BIPR2 : $2485_709C
BIPR3 : $2485_70A0
BIPR4 : $2485_70A4

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	INCPi [15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 22-27. BIPRi Description

Name	Description	Settings
INCPi[15:0]	**BRM INC Preset Register—**These registers are used to load the BRM with the appropriate numerator for the special baud rates while in automatic detect mode. • BIPR1 register is for the 920kbps • BIPR2 register is for the 460kbps • BIPR3 register is for the 230kbps • BIPR4 register is for the 115.2kbps (only used with the 19.44MHz reference frequency) **Note:** The BIPR1 register is not implemented for the 13MHz reference frequency.	N/A

Addr
(UART1):
BMPR1:$2484_D**09A**
BMPR2:$2484_D**09E**
BMPR3:$2484_D**0A2**
BMPR4:$2484_D**0A6**

BMPRi UART BRM MOD Preset Register i

(UART2):
BMPR1:$2485_7**09A**
BMPR2:$2485_7**09E**
BMPR3:$2485_7**0A2**
BMPR4:$2485_7**0A6**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	MODi [15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 22-28. BMPRi Description

Name	Description	Settings
MODPi[15:0]	**Baud Rate Count Register—**These registers are used to load the BRM with the appropriate numerator for the special baud rates while in automatic detect mode. • BMPR1 register is for the 920kbps • BMPR2 register is for the 460kbps • BMPR3 register is for the 230kbps (only used with the 19.44MHz reference frequency) • BIPR4 register is for the 115.2kbps (only used with the 19.44MHz reference frequency) **Note:** The BMPR1 register is not implemented for the 13 MHz reference frequency.	N/A

The Test Register is a read/write register except for bits TS[15], TS[11] and TS[9:0] which will always be read as zero(0). This register contains miscellaneous bits to control test features of the UART block.

The RAMTEST feature is not supported in Neptune LTE UART because it have FIFO instead of Memory in CDR3 UART. FIFO is made of flops which are testable. Therefore there is no need of RAMTEST.

UTS UART Test Register 3 **Addr (UART1):** $2484_D**0A8** **(UART2)** $2485_7**0A8**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
		RAMT	FRCP ERR	LOOP												
TYPE:	r	rw	rw	rw	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-29. UTS Description

Name	Description	Settings
Bit 15	**Reserved**—Read as zero and written with zero for future compatibility	N/A

Table 22-29. UTS Description (Continued)

Name	Description	Settings
RAMT Bit 14	**RAM Test**—This bit is used to enter RAM test mode. During this mode the normal registers will have no effect except BIPR1, BIPR2, BMPR1, BMPR2, UTXD and URXD. During RAM Test mode these registers are used as follows: • BIPR1 register is the READ ADDRESS register (bits 0-5 used; bits 6-15 ignored) • BMPR1 register is the WRITE ADDRESS register (bits 0-5 used; bits 6-15 ignored) • BIPR2 register is the READ DATA register (bits 0-7 used; bits 8-15 ignored) • BMPR2 register is the WRITE DATA register (bits 0-13 used; bits 14-15 ignored) Steps for writing to and reading from RX FIFO: 1. Write the write address to the WRITE ADDRESS register (BMPR1). 2. Write data to the WRITE DATA register (BMPR2). 3. Write data will be loaded to the FIFO after step 2 with the MCU clock. 4 For reading data, first write the read address in the READ ADDRESS register (BIPR1). 5. Then read data from the URX register. Repeat steps 1-5 for writing and reading the TX FIFO but in step 2 write data to the UTXD register to test the TX FIFO and in step 4 read data from the READ DATA register (BIPR2).	0 = RAM Test Mode disabled--Normal operation 1 = RAM Test Mode enabled
FRCPERR Bit 13	**Force Parity Error**—This bit, while HIGH, forces the transmitter to generate a parity error if parity is enabled. This bit is provided for system debugging.	0 = generate normal parity 1 = generate inverted parity (error)
LOOP Bit 12	**Loop TX and RX for Test**—This bit controls loopback for test purposes. While this bit is high, the receiver input is internally connected to the transmitter and ignores the RXD pin. The transmitter is unaffected by this bit.	0 = normal receiver operation 1 = internal connect transmitter output to receiver input
Bit 11	**Reserved bit**—Read as zero and written with zero for future compatibility	N/A
Bits 10-0	**Reserved bits**—Read as zeros and written with zeros for future compatibility	N/A

22.22 UART Operation In Low-Power System States

The UART's serial interface will operate as long as the 16x bit clock generator is provided with the UART-CLK. The bus interface will be operational if the MCU-PER-CLK is running. The three bits RXEN, TXEN and UARTEN, set by the user, give the capability to control low-power modes through software. Table 22-30 shows the UART functionality while in hardware controlled low-power modes.

Table 22-30. UART Low Power State Operation

	Normal State	Wait State	Doze State		Stop State
			DOZE = 0	DOZE = 1	
UART-Clock	ON	ON	ON	ON	OFF
UART Serial I/F	ON	ON	ON	OFF	OFF

While in Doze State, the UART behavior depends on the DOZE control bit. While the DOZE bit is negated, the UART serial interface is operational. While the system is in the Doze State, and the DOZE bit is asserted, the UART is disabled. If the Doze State is entered with the DOZE bit asserted while the UARTs serial interface was receiving or transmitting data, it will complete the receive/transmit of the current character and signal to the far-end transmitter/receiver to stop sending/receiving. The control/status/data registers won't change when getting into/out of low power modes.

The following UART interrupts can wake the MCU from STOP mode:

- RTS Delta (RTSD)
- DTR (DTRF)
- Asynchronous Wake (AWAKE)
- IrDA Asynchronous Wake (AIRINT)

All UART interrupts can wake the MCU from the DOZE or WAIT modes.

NOTE:

The first character received after waking up from STOP or DOZE mode should be discarded. The Asynchronous Wake interrupt must not be used with the DOZE bit set. Use the WAKE interrupt instead.

22.23 UART Operation In System Debug State

When in debug mode, reads of the RXREG will fail to cause the RX FIFO read pointer to increment. That is, the value at the read side of the RX FIFO will not change as a result of reading RXREG when in debug mode. Repeated reads of the RXREG, therefore, will not cause the RXREG to change once it contains a valid character

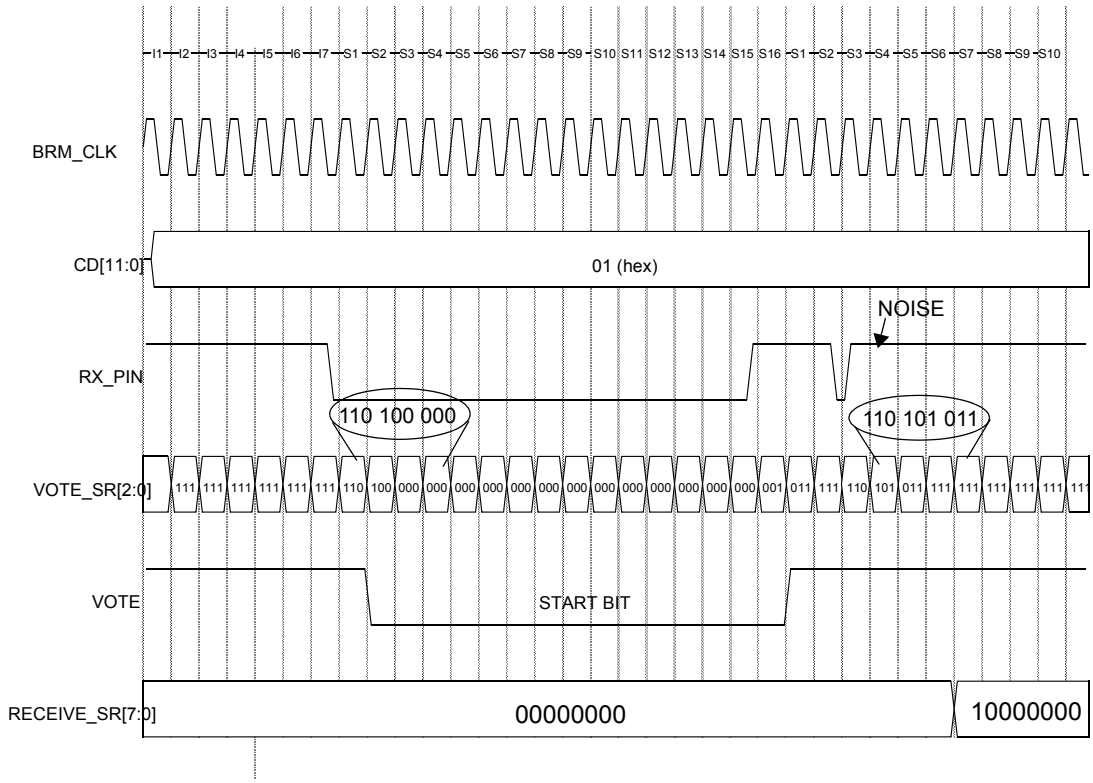


Figure 22-6. Majority Vote Results

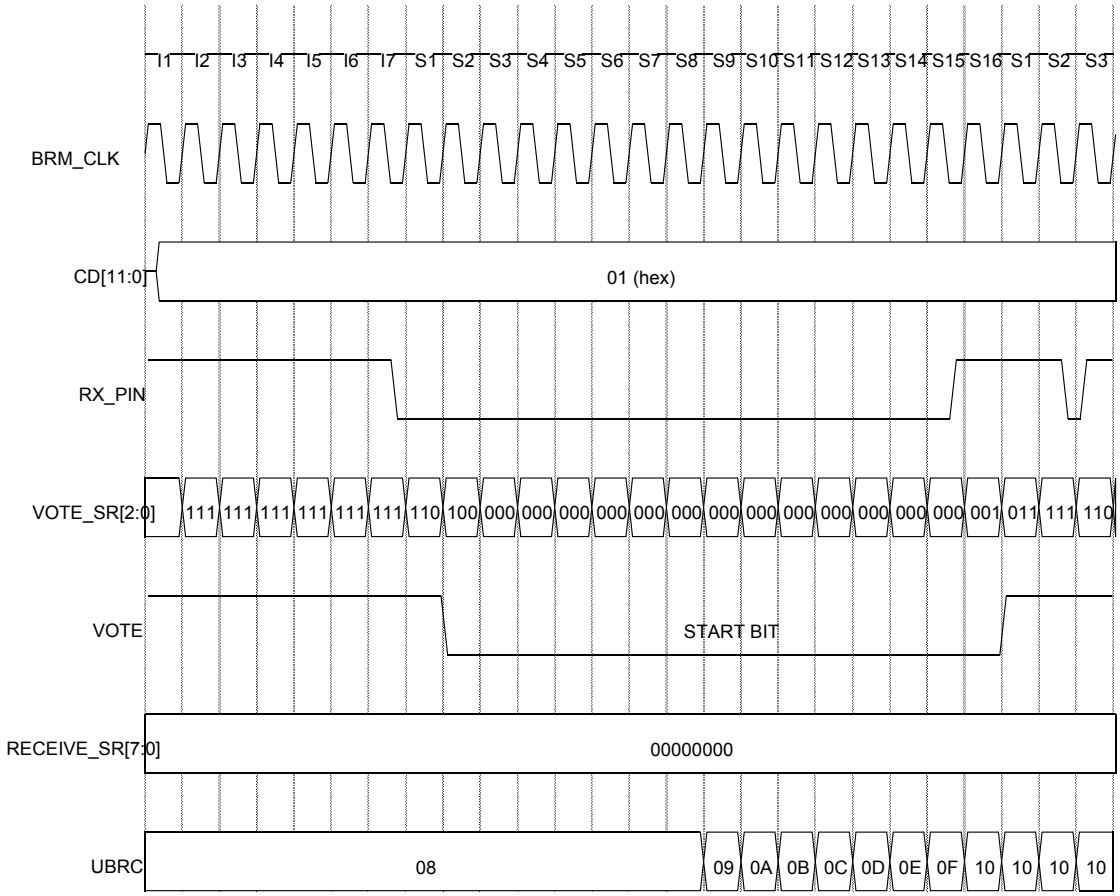


Figure 22-7. Baud Rate Detection of Divisor = 1

22.24 Verification

22.24.1 Registers

1. Values of registers at peripheral reset
2. Values of registers at software reset (SRST_B)
3. Reads from reserved bits are read as zeros
4. Writes to reserved bits are ignored
5. Writes are successful to all read/write bits
6. Writes to read only bits are ignored
7. Writes to read only register UART Baud Rate Count Register are ignored.

22.24.2 Transmitter Functionality

1. Transmission of "1" at peripheral reset WCLK_INT_RESET_B and software reset SRST_B

2. Transmission of “1” while buffer is empty
3. Transmission of a break character
4. Shift register empty signal asserts TXDC.
   a) TXDC asserts WSCI_URT_MINT_TX_B when TCEN is set.
   b) Writing a “1” to TXDC (read only bit) will not clear TXDC and WSCI_URT_MINT_TX_B.
   c) Clearing TCEN will mask out TXDC from asserting WSCI_URT_MINT_TX_B.
5. Transmitter Buffer empty signal asserts TXFE.
   a) TXFE asserts WSCI_URT_MINT_TX_B when TXFEN is set.
   b) Writing a “1” to TXFE (read only bit) will not clear TXFE and WSCI_URT_MINT_TX_B.
   c) Clearing TXFEN will mask out TXFE from asserting WSCI_URT_MINT_TX_B.
6. Buffer emptied below trigger level signal asserts TRDY.
   a) All trigger levels (1-63) asserts TRDY.
   b) TRDY asserts WSCI_URT_MINT_TX_B when TRDYEN is set
   c) Writing a “1” to TRDY (read only bit) will not clear TRDY and WSCI_URT_MINT_TX_B.
   d) Refilling FIFO above trigger level will clear TRDY
   e) Clearing TRDYEN will mask out TRDY from asserting WSCI_URT_MINT_TX_B.
7. Verification of the functionality of the status bits (TRDY, RTSD, RTS_B) by applying all possible combinations of the control bits (UARTEN, TXEN, TRDYEN, RTSDEN). Simultaneously, the external signal RTS_B is varied, and the interrupt signals URT_MINT_TX_B (TX FIFO below trigger level), URT_MINT_RTS_B (RTS edge detected) are monitored.
8. Verification of the basic functionality of the UART transmitter in the simple mode. The TRDY bit is constantly monitored to verify that it gets set/cleared as discussed in the specifications. The transmitter FIFO is overflowed, and it is verified that the characters written to the FIFO after the FIFO overflowed do not get transmitted out. At the same time, the initial characters (before overflow) are transmitted as specified via the control registers.
9. The SNDBRK function is verified thoroughly. Verification that none of the characters written to the FIFO, while the transmitter is sending breaks, are lost. The overflow functionality of the transmitter, as far as writing to the FIFO is concerned, is the same during break transmission, as it is during normal transmission. Verification that when entering the doze mode while sending break the transmitter will continue sending break.
10. Verification of the functionality of the RTS_B pin to ensure that the transmission is not aborted when RTS_B is negated in the middle of a transmission. This section also verifies that the FIFO write pointer is not incremented when the RTS_B signal is negated. Similarly it verifies the functionality of the TXEN and the UARTEN bits.
11. Verification of sending a parity error when FRCPERR bit in UTS register is set.
12. Verification that the setting of the TXFL level causes the TRDY flag to be set/reset accordingly.

22.24.2.1 Transmission Modes

13. Transmission of start bit, 8 data bits, no parity, one stop bit at default baud rate

14. Transmission of start bit, 8 data bits, no parity, two stop bits at default baud rate
15. Transmissions of start bit, 8 data bits, odd parity, one stop bit at default baud rate
16. Transmission of start bit, 8 data bits, odd parity, two stop bits at default baud rate
17. Transmission of start bit, 8 data bits, even parity, one stop bit at default baud rate
18. Transmission of start bit, 8 data bits, even parity, two stop bits at default baud rate
19. Repeat 8-13 with 7 data bits.

22.24.2.2 Baud Rate Variation

20. Transmission of same data in all modes at standard baud rates at all three reference frequencies (13MHz, 16.8MHz, 19.44MHz).

Table 22-31. Standard Baud Rates

Baud Rate	Baud Rate	Baud Rate	Baud Rate
300 bps	4800 bps	28.8 kbps	460.8 kbps
600 bps	9600 bps	57.6 kbps	812.5 kbps
1200 bps	14.4 kbps	115.2 kbps	
2400 bps	19.2 kbps	230.4 kbps	

21. Transmission of data at 920kbps, 460kbps, and 230kbps at all reference frequencies (13MHz, 16.8MHz, 19.44MHz).

22.24.2.3 Automatic Baud Rate Detection Mode

22. Transmissions are halted when ADBR is set and RTS is de-asserted
23. The RTS pin asserts RTSF when it is configured to detect on a specific edge (rising, falling, or either).
    a) RTSF asserts WSCI_URT_MINT_UARTC_B when RTEN is set.
    b) Writing a “1” to RTSF will clear RTSF and WSCI_URT_MINT_RTS_B.
    c) Clearing RTEN will mask out RTSF from asserting WSCI_URT_MINT_RTS_B

22.24.3 Receiver Functionality

1. No receptions are possible when RXEN is cleared.
2. Receptions are possible when RXEN is set
3. Clearing RXEN during a reception will disable the receiver following the detection of the stop bit(s) and that this data is stored in the buffer.
4. Setting RXEN during a reception will cause the incoming data to be discarded. The first byte received will begin with the next valid start bit after the active reception is complete.
5. Receiver can detect a false start
6. Detection of framing error
7. Check CHARRDY, ERR, OVRRUN, BRK, FRMERR, and PRERR for a received character in all modes and baud rates.

8. Reception of the start bit asserts WAKE
   a) WAKE asserts WCLK_INT_RESET_B when WKEN is set.
   b) Writing a “1” to WAKE clears WAKE and WCLK_INT_RESET_B.
   c) Clearing IDEN will mask out WAKE from asserting WSCI_URT_MINT_UARTC_B.
9. Reception of idle frames asserts IDLE.
   a) All idle condition configurations 4,8, 16, and 32 frames assert IDLE
   b) IDLE asserts WSCI_URT_MINT_RX_B when IDEN is set.
   c) Writing a “1” to IDLE clears IDLE and WSCI_URT_MINT_RX_B.
   d) Clearing IDEN will mask out IDLE from asserting WSCI_URT_MINT_RX_B.
10. Reception of a break condition asserts BRCD.
   a) BRCD asserts WSCI_URT_MINT_UARTC_B when BKEN is set.
   b) Writing a “1” to BRCD clears BRCD and WSCI_URT_MINT_UARTC_B.
   c) Clearing BKEN will mask out BRCD from asserting WSCI_URT_MINT_UARTC_B
11. Buffer overrun error signal asserts ORE.
   a) ORE asserts WSCI_URT_MINT_UARTC_B when OREN is set.
   b) Writing a “1” to ORE clears ORE and WSCI_URT_MINT_RX_B.
   c) Clearing OREN will mask out ORE from asserting WSCI_URT_MINT_RX_B.
12. Buffer not empty signal asserts RDR
   a) RDR asserts WSCI_URT_MINT_RX_B when DREN is set.
   b) Writing a “1” to RDR (read only bit) will not clear RDR and WSCI_URT_MINT_RX_B.
   c) Clearing DREN will mask out RDR from asserting WSCI_URT_MINT_RX_B.
13. Buffer filled to trigger level signal asserts RRDY.
   a) All trigger levels (1-63) asserts RRDY.
   b) RRDY asserts WSCI_URT_MINT_RX_B when RRDYEN is set
   c) Writing a “1” to RRDY (read only bit) will not clear RRDY and WSCI_URT_MINT_RX_B.
   d) FIFO exceeding trigger level will clear RRDY
   e) Clearing RRDYEN will mask out RRDY from asserting WSCI_URT_MINT_RX_B.
14. Verification of the functionality of the status bits (RRDY, CHARRDY) by applying all possible combinations of the control bits (UARTEN, RRDYEN, RXEN). Simultaneously, the interrupt signal URT_MINT_RX_B (RX FIFO has filled pass trigger level) is monitored.
15. Verification of the behavior of the receiver on a framing error reception. The characters following a framing error are also monitored to verify that they are not received in a corrupted manner. Add cases where the receiver has been setup to receive two stop bits, but a character with only one stop bit is received. This should not be flagged as a framing error.
16. Verification of the behavior of the receiver on reception of single break characters, long break characters, multiple break characters with intermediate marks, and break characters interleaved among normal characters. The behavior of the receiver when break characters are mixed with framing-errored characters is also verified.

17. Verification of the receiver behavior on reception of single character with parity error, multiple characters with parity error, and characters with parity error mixed with normal characters.
18. Verification of the setting of the RXFL level causes the RRDY flag to be set/reset accordingly.
19. Verification of the behavior of the UART when the receiver and the transmitter are running off skewed clocks.
20. Verification of the functionality of the CTSTL bit so that the $\overline{\text{CTS}}$ is de-asserted by the receiver.

22.24.3.1 Reception Modes

21. Reception of start bit, 8 data bits, no parity, one stop bit at default baud rate
22. Reception of start bit, 8 data bits, odd parity, one stop bit at default baud rate
23. Reception of start bit, 8 data bits, even parity, one stop bit at default baud rate
24. Repeat 14, 15, 16 with 7 data bits.

22.24.3.2 Baud Rate Variation

25. Reception of same data in all modes at standard baud rates, see Table 22-30.
26. Reception of same data, same mode at all baud rates

22.24.3.3 Automatic Baud Rate Detection Mode

27. The Receiver Buffer full signal de-asserts CTS when ADBR is set.
28. Reception of an ASCII character “a” or “A” asserts ADET when ADEN is set.
    a) ADET assets WSCI_URT_MINT_UARTC_B when ADEN is set
    b) Writing a “1’ to ADET (read only bit) will not clear ADET and WSCI_URT_MINT_UARTC_B.
    c) Clearing ADEN will mask out the reception of an ASCII character “a” or “A” from asserting ADET and WSCI_URT_MINT_UARTC_B
29. Verify the preset registers functionality

22.24.4 Serial Infrared Functionality

1. Serial Infrared Block is bypassed when IREN is cleared.
2. Serial Infrared Block is selected when IREN is set.
3. Encoder is encoding the transmitted signal
    a) Transmitters Functionality in this mode with INVT cleared.
    b) Transmitters Functionality in this mode with INVT set.
4. Decoder is decoding the received signal
    a) Receivers Functionality in this mode with INVR cleared.
    a. Receivers Functionality in this mode with INVR set.
5. Detection of either edge on RX pin will assert IRINT.
    a) IRINT asserts WSCI_URT_MINT_UARTC_B when ENIRI is set.

b) Writing a “1” to IRINT clears IRINT and WSCI_URT_MINT_UARTC_B.

c) Clearing ENIRI will mask out IRINT from asserting WSCI_URT_MINT_UARTC_B.

6. Verify functionality of IRSC bit.

22.24.5 Vote Logic Functionality

1. At reset Vote result is “1”.
   a) Peripheral and Software reset, resets the three votes to “111”
2. All combinations of Vote inputs produce right vote result.

22.24.6 Test Register

1. Verify all functions of Test register bits.

22.24.7 Low Power Mode Test

1. All low power modes should be tested. For
   a) UART’s functionality in various modes.
   b) It should be seen that whether UART is generating appropriate interrupt in the corresponding mode.

Chapter 23
General Purpose Input/Output (GPIO)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	03/19/02	Vishal Gupta	-Added description for additional ports used in Netune-LTS. (B[5,6] , C[0,15} , E[15:12]). -Removed input enable signals from INT .
0.2	04/18/02	Vishal Gupta	- Used inverted clk gclkw_b inplace of gclkw -replaced scan_mode by ipt_scan_mode, tcm_scan_enable with ipt_clk_se - Removed scan_divergence test mode. - Removed input enable signals from sap, dtimer. -Changed reset values of registers . -Added DFT related changes in ports.
0.3	05/03/02	Vishal Gupta	- Correction in spec, The priority given to DSP interrupts in interrupt status register, following DDTS No. DSPhl14152.
0.4	06/19/02	Vishal Gupta	- Spec updation following DDTS DSPhl14289 - Added description for priority selection of multiple inputs: SC2A, SCKA, SRDA, INT4, INT5, RTS1, RXD1, DTR1,MISOB. - Added BIST related signals, owire_oe and gpio_porta2_ode signals in GPIO port list. -changed names of codec enable signals and test pin. -Updated test mode sheet for scan mode and dsp_address trace mode. -Updated internal interrupt visibility table. -removed dmem and mmux scan_mode signals from port list following DSPhl13759 - Added res port in GPIO, changed reset value of gpio_dsp_irq_b to 0. -Added description for port configuration in BIST mode.
0.5	07/17/02	Vishal Gupta	- Spec correction following DDTS DSPhl14951. - Removed port res.
0.6	07/22/02	Shannon Osgood	Updated per DDTS# TLSbo24386.
0.7	05/07/03		Removed references to LTS and ULS, updated per DDTS# DSPhl16791/DSPhl16789.

23.1 GPIO Introduction

The GPIO module is a stand alone module that serves as the all-in-one communication link between the outside world, the MCU module and the DSP module. The GPIO communicates with the MCU through the IP Bus and to the DSP through the DSP Peripheral Bus. The GPIO module houses five 16 pin bi-directional Ports.

23.2 GPIO Features

- Five 16 pin bi-directional Ports (A-E)
- Alternate output function to select 1 of up to 8 output paths for each GPIO pin
- Alternate input function to select 1 of up to four input paths for each GPIO pin
- Shared Port Data Access for DSP and MCU
- MCU, DSP interrupt capability with selectable rise/fall edge triggered or high/low level sensitive interrupts for MCU interrupt
- Interrupt status register with “write-one-to-clear” for MCU interrupt
- Internal interrupt visibility of any of 64 interrupts
- Bus Master Mode functionality
- MuxDFF SCAN mode functionality
- DSP Address Trace Mode and DSP Master Mode I/O
- Bist Mode functionality

23.3 GPIO Block Diagram

The overall block diagram, Figure 23-1 on page 23-3 below, shows the five ports on the GPIO as well as the bus interfaces, peripheral interface and the Interrupt Status Register.

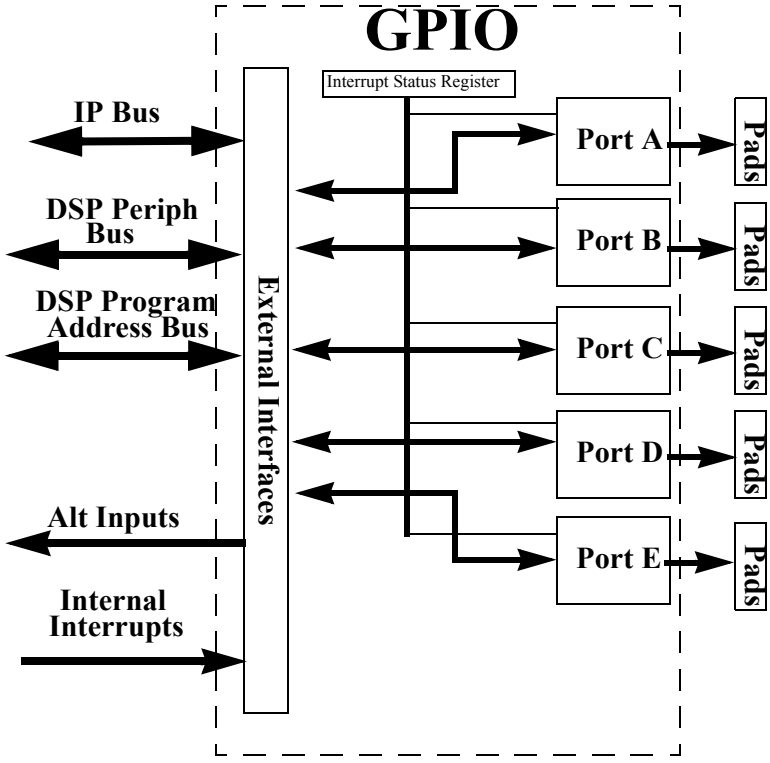


Figure 23-1. GPIO Overall Block Diagram

23.4 GPIO Functional Description

The GPIO has five 16 bit I/O Ports (A-E). Each GPIO Port pin has multiple functionality that allows sharing between designated chip functions, several test functions, and general purpose I/O. These multiple functions are selected by software through port configuration registers and/or by hardware through several control signals.

23.4.1 External Interrupt Control

Ports A-E can be configured to generate an interrupt from any pin of these Ports. Each Port pin can be individually masked through the Port's interrupt mask register. The interrupts are detected from an OR of all the unmasked port pin inputs. A 4 bit interrupt control register in each port provides two control bits to configure the interrupt for rise/fall edge or high/low level sensitive for the MCU interrupt only, and one bit each to mask the MCU or DSP interrupts separately. The GPIO interrupt status register shows the status of the DSP or the MCU interrupts. If only the DSP interrupts are enabled, it shows the status of the DSP interrupts. If only the MCU interrupts are enabled, it shows the status of the mcu interrupts. If both the DSP and MCU interrupts are enabled, the interrupt status register shows the status of the DSP interrupts.

The GPIO interrupt status register has a write-one-to-clear provision for MCU interrupts only. write-one-clear has no effect when the dsp interrupts are enabled.If only the MCU interrupts are enabled and the interrupts are configured for edge detection, write-one-to-clear clears the interrupt status register. However, write-one-to-clear does not clear the interrupt status register in case of level triggered MCU interrupts as long as the interrupts are active.

One interrupt request per port is output for the MCU. A single interrupt request to the DSP is output which is an OR of the five Port interrupts.

Restriction on configuration of interrupt mask and interrupt control register:

1) For falling edge interrupts, we would expect the signal to be high when the interrupt is unmasked.

2) For rising edge interrupts, we would expect the signal to be low when the interrupt is enabled.

3) The interrupt control register should only be changed when the interrupts are masked.

23.4.2 Alternate Output Functionality

Each Port pin can output one of up to eight muxed alternate signals. Two of these signals are from the Port's DSP and MCU data registers and the remaining are from various modules on the chip. The alternate outputs are selected in software through the Port's configuration registers or in some cases by hardware through control signals.

23.4.3 Alternate Input Functionality

Each Port pin can input data and route it to one of up to four alternate signals. The alternate inputs are selected in software through the Port's alternate input registers or in some cases by hardware through control signals.

23.4.4 Internal Interrupt Visibility

Internal interrupts from the chip can be visible through the GPIO for debug purposes. Sixty-four (64) internal interrupts are muxed in Port B and made visible on five Port B pins (0-4) and five Port E pins (4-8). These are configured by the interrupt select register.

23.4.5 DSP Address Trace Mode and DSP Master Mode

DSP Address Trace Mode allows visibility of the DSP Program Address Bus through Port E. Below is a description of how to get into Address Trace Mode:

1. Set the ATE bit in DSP OMR register.
2. Set the nsertst signal in the TCM Miscellaneous Control Register, through a MCU write. The other way to set the nsertst signal is through the DSP JTAG register. This signal will configure Port E correctly.

DSP Master Mode provides access to the DSP Port A external memory interface. To get into DSP Master Mode, assert the ngtm signal instead of the nsertst signal. This signal can also be asserted the same way the nsertst signal is for the Address Trace Mode. DSP master mode configures Port A[15-0], Port B12 and Port C[10], Port C[8-1] for bdio[23:0], bdio_rd_b and bdio_wr_b.

23.4.6 Shared Test Connections

The GPIO shares I/O for seven different chip test connections. These test modes are selected by a combination of software through the port configuration registers and by hardware through several control signals. Look at *TEST Control Module* spec for more detail.

23.5 GPIO Register Summary

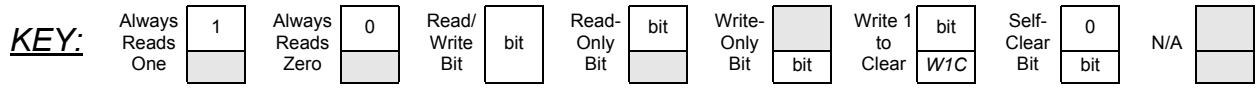


Table 23-1. GPIO Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAConfReg0 ($2484_1000)	R	0	0	P9_S2	P9_S1	P9_S0	P8_S2	P8_S1	P8_S0	P7_S2	P7_S1	P7_S0	P6_S2	P6_S1	P6_S0	P5_S2	P5_S1
	W																
	R	P5_S0	P4_S2	P4_S1	P4_S0	P3_S2	P3_S1	P3_S0	P2_S2	P2_S1	P2_S0	P1_S2	P1_S1	P1_S0	P0_S2	P0_S1	P0_S0
	W																
PAConfReg1 ($2484_1004)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	P15_S2	P15_S1
	W																
	R	P15_S0	P14_S2	P14_S1	P14_S0	P13_S2	P13_S1	P13_S0	P12_S2	P12_S1	P12_S0	P11_S2	P11_S1	P11_S0	P10_S2	P10_S1	P10_S0
	W																
PADDirReg ($2484_1008)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	DirP15	DirP14	DirP13	DirP12	DirP11	DirP10	DirP9	DirP8	DirP7	DirP6	DirP5	DirP4	DirP3	DirP2	DirP1	DirP0
	W																
PAIMaskReg ($2484_100C)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	Msk15	Msk14	Msk13	Msk12	Msk11	Msk10	Msk9	Msk8	Msk7	Msk6	Msk5	Msk4	Msk3	Msk2	Msk1	Msk0
	W																
PAIntCtrlReg ($2484_1010)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MCU Mask	DSP Mask	Edge/ Level	Rise/fall High/low
	W																
PAAltInReg ($2484_1014)	R	Alt15S1	Alt15S0	Alt14S1	Alt14S0	Alt13S1	Alt13S0	Alt12S1	Alt12S0	Alt11S1	Alt11S0	Alt10S1	Alt10S0	Alt9S1	Alt9S0	Alt8S1	Alt8S0
	W																
	R	Alt7S1	Alt7S0	Alt6S1	Alt6S0	Alt5S1	Alt5S0	Alt4S1	Alt4S0	Alt3S1	Alt3S0	Alt2S1	Alt2S0	Alt1S1	Alt1S0	Alt0S1	Alt0S0
	W																
PBConfReg0 ($2484_1018)	R	0	0	P9_S2	P9_S1	P9_S0	P8_S2	P8_S1	P8_S0	P7_S2	P7_S1	P7_S0	P6_S2	P6_S1	P6_S0	P5_S2	P5_S1
	W																
	R	P5_S0	P4_S2	P4_S1	P4_S0	P3_S2	P3_S1	P3_S0	P2_S2	P2_S1	P2_S0	P1_S2	P1_S1	P1_S0	P0_S2	P0_S1	P0_S0
	W																
PBConfReg1 ($2484_101C)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	P15_S2	P15_S1
	W																
	R	P15_S0	P14_S2	P14_S1	P14_S0	P13_S2	P13_S1	P13_S0	P12_S2	P12_S1	P12_S0	P11_S2	P11_S1	P11_S0	P10_S2	P10_S1	P10_S0
	W																

Table 23-1. GPIO Register Summary (Continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PBDDirReg ($2484_1020)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R W	DirP15	DirP14	DirP13	DirP12	DirP11	DirP10	DirP9	DirP8	DirP7	DirP6	DirP5	DirP4	DirP3	DirP2	DirP1	DirP0
PBIMaskReg ($2484_1024)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R W	Msk15	Msk14	Msk13	Msk12	Msk11	Msk10	Msk9	Msk8	Msk7	Msk6	Msk5	Msk4	Msk3	Msk2	Msk1	Msk0
PBIntCtrlReg ($2484_1028)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MCU Mask	DSP Mask	Edge/ Level	Rise/fall High/low
	W																
PBAltInReg ($2484_102C)	R W	Alt15S1	Alt15S0	Alt14S1	Alt14S0	Alt13S1	Alt13S0	Alt12S1	Alt12S0	Alt11S1	Alt11S0	Alt10S1	Alt10S0	Alt9S1	Alt9S0	Alt8S1	Alt8S0
	R W	Alt7S1	Alt7S0	Alt6S1	Alt6S0	Alt5S1	Alt5S0	Alt4S1	Alt4S0	Alt3S1	Alt3S0	Alt2S1	Alt2S0	Alt1S1	Alt1S0	Alt0S1	Alt0S0
PCConfReg0 ($2484_1030)	R	0	0	P9_S2	P9_S1	P9_S0	P8_S2	P8_S1	P8_S0	P7_S2	P7_S1	P7_S0	P6_S2	P6_S1	P6_S0	P5_S2	P5_S1
	W																
	R W	P5_S0	P4_S2	P4_S1	P4_S0	P3_S2	P3_S1	P3_S0	P2_S2	P2_S1	P2_S0	P1_S2	P1_S1	P1_S0	P0_S2	P0_S1	P0_S0
PCConfReg1 ($2484_1034)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	P15_S2	P15_S1
	W																
	R W	P15_S0	P14_S2	P14_S1	P14_S0	P13_S2	P13_S1	P13_S0	P12_S2	P12_S1	P12_S0	P11_S2	P11_S1	P11_S0	P10_S2	P10_S1	P10_S0
PCDDirReg ($2484_1038)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R W	DirP15	DirP14	DirP13	DirP12	DirP11	DirP10	DirP9	DirP8	DirP7	DirP6	DirP5	DirP4	DirP3	DirP2	DirP1	DirP0
PCIMaskReg ($2484_103C)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R W	Msk15	Msk14	Msk13	Msk12	Msk11	Msk10	Msk9	Msk8	Msk7	Msk6	Msk5	Msk4	Msk3	Msk2	Msk1	Msk0
PCIntCtrlReg ($2484_1040)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MCU Mask	DSP Mask	Edge/ Level	Rise/fall High/low
	W																
PCAltInReg ($2484_1044)	R W	Alt15S1	Alt15S0	Alt14S1	Alt14S0	Alt13S1	Alt13S0	Alt12S1	Alt12S0	Alt11S1	Alt11S0	Alt10S1	Alt10S0	Alt9S1	Alt9S0	Alt8S1	Alt8S0
	R W	Alt7S1	Alt7S0	Alt6S1	Alt6S0	Alt5S1	Alt5S0	Alt4S1	Alt4S0	Alt3S1	Alt3S0	Alt2S1	Alt2S0	Alt1S1	Alt1S0	Alt0S1	Alt0S0
PDConfReg0 ($2484_1048)	R	0	0	P9_S2	P9_S1	P9_S0	P8_S2	P8_S1	P8_S0	P7_S2	P7_S1	P7_S0	P6_S2	P6_S1	P6_S0	P5_S2	P5_S1
	W																
	R W	P5_S0	P4_S2	P4_S1	P4_S0	P3_S2	P3_S1	P3_S0	P2_S2	P2_S1	P2_S0	P1_S2	P1_S1	P1_S0	P0_S2	P0_S1	P0_S0

Table 23-1. GPIO Register Summary (Continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDConfReg1 ($2484_104C)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	P15_S2	P15_S1
	W																
	R	P15_S0	P14_S2	P14_S1	P14_S0	P13_S2	P13_S1	P13_S0	P12_S2	P12_S1	P12_S0	P11_S2	P11_S1	P11_S0	P10_S2	P10_S1	P10_S0
	W																
PDDDirReg ($2484_1050	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	DirP15	DirP14	DirP13	DirP12	DirP11	DirP10	DirP9	DirP8	DirP7	DirP6	DirP5	DirP4	DirP3	DirP2	DirP1	DirP0
	W																
PDIMaskReg ($2484_1054)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	Msk15	Msk14	Msk13	Msk12	Msk11	Msk10	Msk9	Msk8	Msk7	Msk6	Msk5	Msk4	Msk3	Msk2	Msk1	Msk0
	W																
PDIntCtrlReg ($2484_1058)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MCU Mask	DSP Mask	Edge/ Level	Rise/fall High/low
	W																
PDAltInReg ($2484_105C)	R	Alt15S1	Alt15S0	Alt14S1	Alt14S0	Alt13S1	Alt13S0	Alt12S1	Alt12S0	Alt11S1	Alt11S0	Alt10S1	Alt10S0	Alt9S1	Alt9S0	Alt8S1	Alt8S0
	W																
	R	Alt7S1	Alt7S0	Alt6S1	Alt6S0	Alt5S1	Alt5S0	Alt4S1	Alt4S0	Alt3S1	Alt3S0	Alt2S1	Alt2S0	Alt1S1	Alt1S0	Alt0S1	Alt0S0
	W																
PEConfReg0 ($2484_1060)	R	0	0	P9_S2	P9_S1	P9_S0	P8_S2	P8_S1	P8_S0	P7_S2	P7_S1	P7_S0	P6_S2	P6_S1	P6_S0	P5_S2	P5_S1
	W																
	R	P5_S0	P4_S2	P4_S1	P4_S0	P3_S2	P3_S1	P3_S0	P2_S2	P2_S1	P2_S0	P1_S2	P1_S1	P1_S0	P0_S2	P0_S1	P0_S0
	W																
PEConfReg1 ($2484_1064)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	P15_S2	P15_S1
	W																
	R	P15_S0	P14_S2	P14_S1	P14_S0	P13_S2	P13_S1	P13_S0	P12_S2	P12_S1	P12_S0	P11_S2	P11_S1	P11_S0	P10_S2	P10_S1	P10_S0
	W																
PEDDirReg ($2484_1068)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	DirP15	DirP14	DirP13	DirP12	DirP11	DirP10	DirP9	DirP8	DirP7	DirP6	DirP5	DirP4	DirP3	DirP2	DirP1	DirP0
	W																
PEIMaskReg ($2484_106C)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	Msk15	Msk14	Msk13	Msk12	Msk11	Msk10	Msk9	Msk8	Msk7	Msk6	Msk5	Msk4	Msk3	Msk2	Msk1	Msk0
	W																
PEIntCtrlReg ($2484_1070)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MCU Mask	DSP Mask	Edge/ Level	Rise/fall High/low
	W																
PEAltInReg ($2484_1074)	R	Alt15S1	Alt15S0	Alt14S1	Alt14S0	Alt13S1	Alt13S0	Alt12S1	Alt12S0	Alt11S1	Alt11S0	Alt10S1	Alt10S0	Alt9S1	Alt9S0	Alt8S1	Alt8S0
	W																
	R	Alt7S1	Alt7S0	Alt6S1	Alt6S0	Alt5S1	Alt5S0	Alt4S1	Alt4S0	Alt3S1	Alt3S0	Alt2S1	Alt2S0	Alt1S1	Alt1S0	Alt0S1	Alt0S0
	W																

Table 23-1. GPIO Register Summary (Continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IntSelReg ($2484_1078)	R	0	0	IV4S5	IV4S4	IV4S3	IV4S2	IV4S1	IV4S0	IV3S5	IV3S4	IV3S3	IV3S2	IV3S1	IV3S0	IV2S5	IV2S4
	W																
	R	IV2S3	IV2S2	IV2S1	IV2S0	IV1S5	IV1S4	IV1S3	IV1S2	IV1S1	IV1S0	IV0S5	IV0S4	IV0S3	IV0S2	IV0S1	IV0S0
	W																
IntStatReg $2484_107C	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	ISTATE	ISTATD	ISTATC	ISTATB	ISTATA
	W												*W1C*	*W1C*	*W1C*	*W1C*	*W1C*
MCUPADataReg ($2484_1080)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	MCUPAData[15:0]															
	W																
MCUPBDataReg ($2484_1084)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	MCUPBData[15:0]															
	W																
MCUPCDataReg ($2484_1088)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	MCUPCData[15:0]															
	W																
MCUPDDataReg ($2484_108C)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	MCUPDData[15:0]															
	W																
MCUPEDataReg ($2484_1090)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	MCUPEData[15:0]															
	W																
MCUBGReg ($2484_1094)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	Vrefen	0	0	0	0	0	0	0	BGrdy
	W																
DSPPADataReg (Y:$FFD0)	R	DSPPAData[15:0]															
	W																
DSPPBDataReg (Y:$FFD1)	R	DSPPBData[15:0]															
	W																
DSPPCDataReg (Y:$FFD2)	R	DSPPCData[15:0]															
	W																
DSPPDDataReg (Y:$FFD3)	R	DSPPDData[15:0]															
	W																
DSPPEDataReg (Y:$FFD4)	R	DSPPEData[15:0]															
	W																
DSPBGRef (Y:$FFD7)	R	0	0	0	0	0	0	0	Vrefen	0	0	0	0	0	0	0	BGrdy
	W																

23.6 Port A Functional Description

Port A is a 16 bit I/O Port that can be configured for a multitude of functions. Diagrams and register descriptions will describe the general functionality of Port A as well as the configuration of the pins for the various tests modes that share Port A pins. Refer to the GPIO Configuration section in Chapter 2, “Neptune LTE IC Pin Description (PINS).” for specific chip pin assignments to Port A for dedicated functions. Also refer to Section 23.14, “Port Diagrams,” on page 23-25 and Section 23.15, “Detailed Register Descriptions (Port A-E),” on page 23-76, for more detailed descriptions.

23.7 Port B Functional Description

Port B is a 16 bit I/O Port that can be configured for a multitude of functions. Diagrams and register descriptions will describe the general functionality of Port B as well as the configuration of the pins for the various tests modes that share Port B pins. Refer to the GPIO Configuration section in Chapter 2, “Neptune LTE IC Pin Description (PINS).” for specific chip pin assignments to Port B for dedicated functions. Also refer to Section 23.14, “Port Diagrams,” on page 23-25 and Section 23.15, “Detailed Register Descriptions (Port A-E),” on page 23-76, for more detailed descriptions.

23.8 Port C Functional Description

Port C is a 16 bit I/O Port that can be configured for a multitude of functions. Diagrams and register descriptions will describe the general functionality of Port C as well as the configuration of the pins for the various tests modes that share Port C pins. Refer to the GPIO Configuration section in Chapter 2, “Neptune LTE IC Pin Description (PINS).” for specific chip pin assignments to Port C for dedicated functions. Also refer to Section 23.14, “Port Diagrams,” on page 23-25 and Section 23.15, “Detailed Register Descriptions (Port A-E),” on page 23-76 for more detailed descriptions.

23.9 Port D Functional Description

Port D is a 16 bit I/O Port that can be configured for a multitude of functions. Diagrams and register descriptions will describe the general functionality of Port D as well as the configuration of the pins for the various tests modes that share Port D pins. Refer to the GPIO Configuration section in Chapter 2, “Neptune LTE IC Pin Description (PINS).” for specific chip pin assignments to Port D for dedicated functions. Also refer to Section 23.14, “Port Diagrams,” on page 23-25 and Section 23.15, “Detailed Register Descriptions (Port A-E),” on page 23-76 for more detailed descriptions.

23.10 Port E Functional Description

Port E is a 16 bit I/O Port that can be configured for a multitude of functions. Diagrams and register descriptions will describe the general functionality of Port E as well as the configuration of the pins for the various tests modes that share Port E pins. Refer to the GPIO Configuration section in Chapter 2, “Neptune LTE IC Pin Description (PINS).” for specific chip pin assignments to Port E for dedicated functions. Also refer to Section 23.14, “Port Diagrams.” and Section 23.15, “Detailed Register Descriptions (Port A-E),” on page 23-76 for more detailed descriptions.

23.11 GPIO Special Functions

The GPIO has some special configurations for some of the ports. PortB 10, Port C3, Port C4 are configured as open drain for SIM_DIO, Keypad Column0 and Column1 signals respectively. In Port D11, the PWM signal from the EGPT is inverted to provide the PWM_N on the pad when the port is configured to select functional output 2.

Following ports require additional control signals to determine the direction of ports and to enable data in and out of GPIO for some of their mux inputs.

1) PortA0: When functional output 010 is selected, the direction of CKO is determined by cko_od control from ccm. If cko_od = 1, CKO output to pad is disabled. If cko od = 0, cko output to pad is available.

2) Port A14: When functional output 010 is selected, the direction of the watchdog output is determined by wdog_oe from wdog module. If wdog_oe = 1, output path to pad is enabled. If wdog_oe = 0, output path to pad is disabled.

3) Port A14: When functional output 100 is selected, the direction of the CKOH is determined by the ckoh_od from CCM. If ckoh_od = 1, ckoh output to pad is disabled. If ckoh_od = 0, ckoh output to pad is enabled.

4) PortB1: When functional output 010 is selected, the direction of sap_scka is determined by sap_scka_oe signal. If sap_scka_oe = 1, output path from sap_scka to pad is enabled. If sap_scka_oe = 0, output path to pad is disabled and the port behaves as input.

5) PortB2: When functional output 010 is selected, the direction of sap_stda is determined by sap_stda_oe signal. If sap_stda_oe = 1, output path from sap_stda to pad is enabled. If sap_stda_oe = 0, output path to pad is disabled and the port behaves as input.

6) PortB3: When functional output 010 is selected, the direction of sap_sc2a is determined by sap_sc2a_oe signal. If sap_sc2a_oe = 1, output path from sap_sc2a to pad is enabled. If sap_sc2a_oe = 0, output path to pad is disabled and the port behaves as input.

7) PortB2: When functional output 111 is selected, the direction of bbp_stdb is determined by bbp_stdb_oe from BBP. if bbp_stdb_oe = 1, output path from bbp_stdb to pad is enabled. If bbp_stdb=0, output path to pad is disabled and the port behaves as input.

8) PortB13, B14 and B15: When alternate input 0 and functional output 101 are selected, the direction of dtimer signal is determined by the dtimer_ie signal. If dtimer_ie = 0, the output path from dtimer_fiod to pad is enabled. If dtimer_ie = 1, the output path from dtimer_fiod to pad is disabled

9) PortC3, C4: When alternate input 0 and functional output 010 are selected, the direction is determined by column1(0)_oe and column1(0)_ie signals from KPP. If column1(0)_oe = 1, output path from column1(0) to pad is enabled.

The column1(0)_ode signals from KPP determine the open drain enable for each of these ports.

10) PortA2: When functional output 111 are selected, the direction is determined by owire_oe signal from OWIRE. If owire_oe = 1, output path to pad is enabled.

The open drain enable is generated from GPIO if functional output 111 is selected, which determine the open drain enable for the pad.

23.12 Primary Selection for Multiple Inputs.

23.12.1 Multiple SAP SCKA Connections

Since SAP signal SCKA has two assigned I/O locations through GPIO ports PB1 and PE10, a provision to prioritize which port selection has priority if more than one is selected is done. For primary configuration of SCKA through Port B1, the Port B Configuration Register must select Port B1 alternate output 2 and the Port B Alternate Input Register must select Port B1 alternate input 0. For primary configuration of SCKA through Port E10, the Port E Configuration Register must select Port E10 alternate output 7 and the Port E Alternate Input Register must select Port E10 alternate input 1. In addition, alternate input 0 cannot be selected on Port B1 (to release the input priority mux to the Port E10 input).

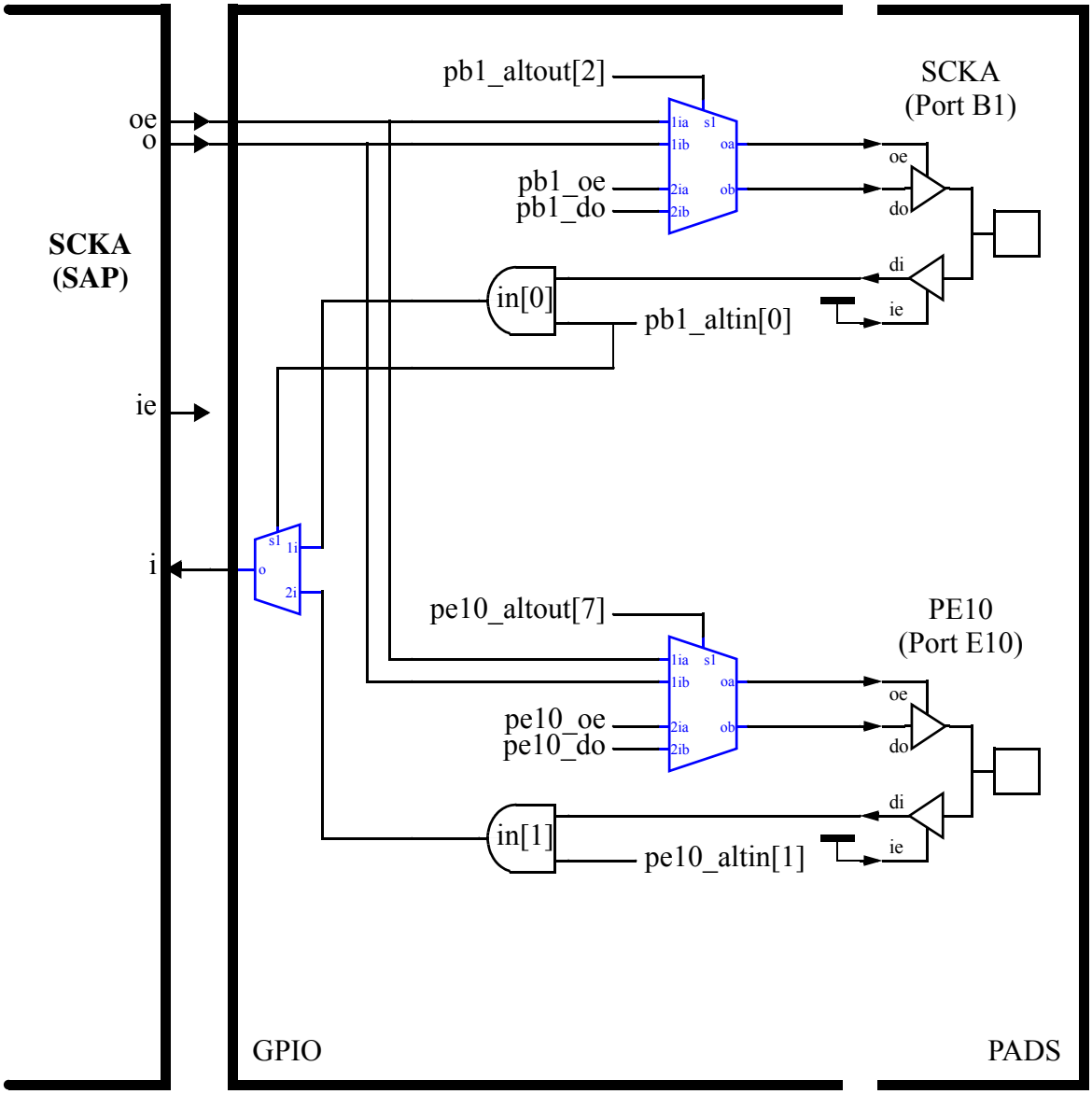


Figure 23-2.

23.12.2 Multiple SAP SC2A Connections

Since SAP signal SC2A has two assigned I/O locations through GPIO ports PB3 and PB0, a provision to prioritize which port selection has priority if more than one is done. For primary configuration of SC2A through Port B3, the Port B Configuration Register must select Port B3 alternate output 2 and the Port B Alternate Input Register must select Port B3 alternate input 0. For primary configuration of SCKA through Port B0, the Port B Configuration Register must select Port B0 alternate output 3 and the Port B Alternate Input Register must select Port B0 alternate input 1. In addition, alternate input 0 cannot be selected on Port B3 (to release the input priority mux to the Port B0 input).

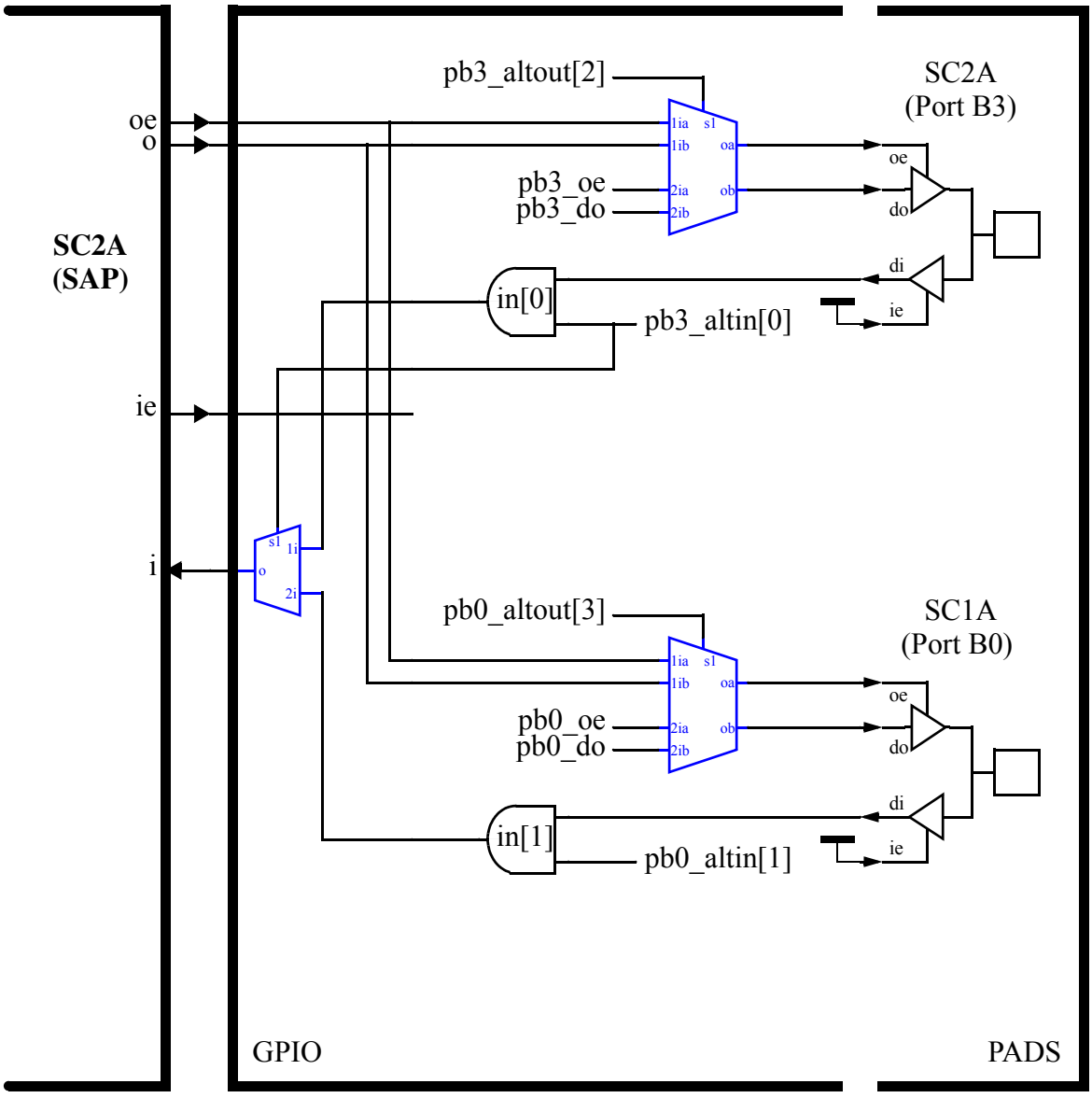


Figure 23-3.

23.12.3 Multiple SAP SRDA Inputs

Since SAP signal SRDA is an input signal on both Port B4 and Port E12, a selection priority is created to determine which input will be routed to the SAP module. If Port B4 alternate input 0 is selected, SRDA will route from Port B4 to the SAP module. To select SRDA through Port E12, set the Port E Alternate Input Register to select Port E12 alternate input 2 and set the Port B Alternate Input Register to select any Port B4 alternate input EXCEPT for selection 0 (to release the input mux priority to the Port E12 input).

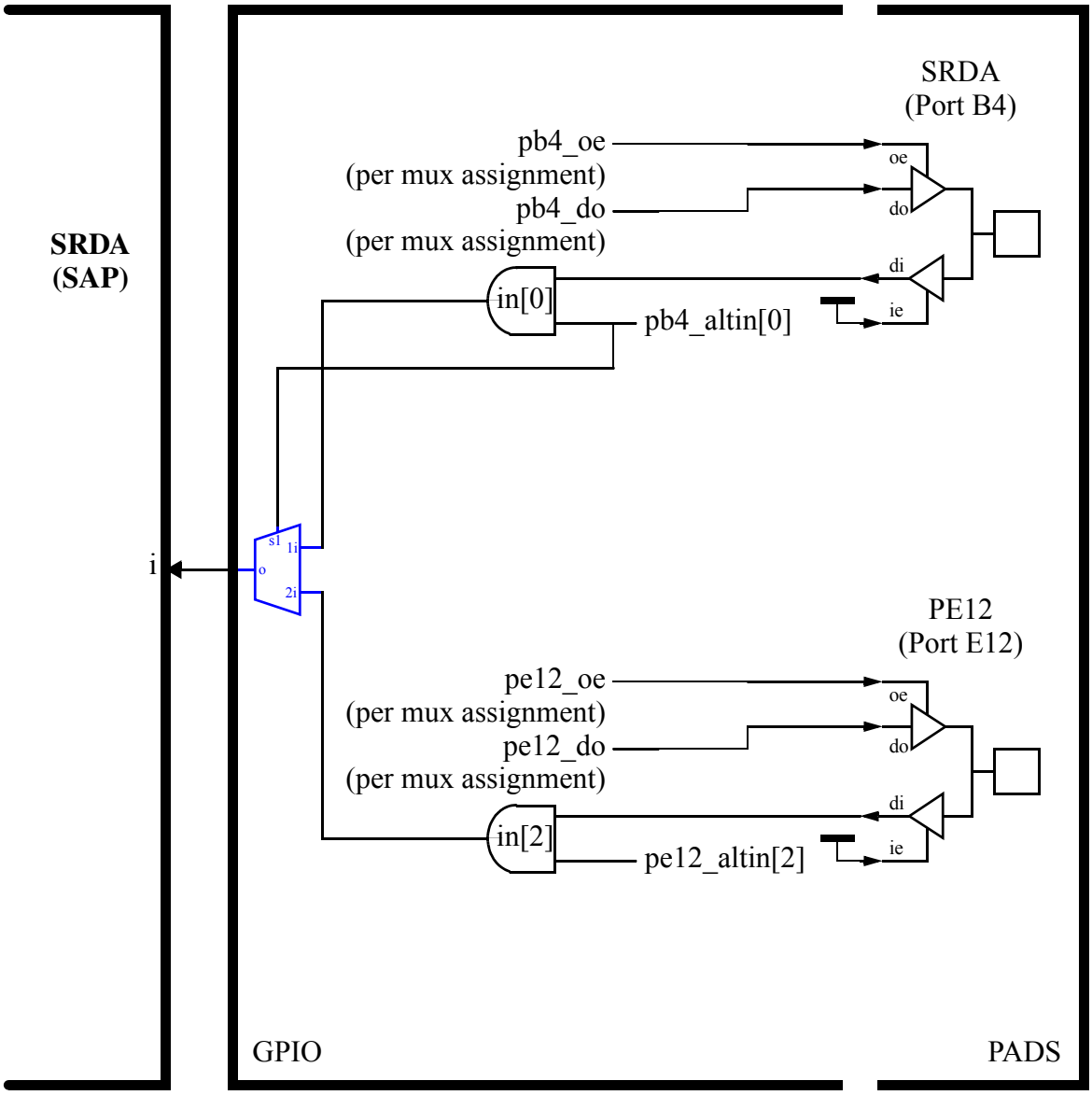


Figure 23-4.

23.12.4 Multiple INT INT4 Inputs

Since INT signal INT4 is an input signal on both Port A13 and Port E3, a selection priority is created to determine which input will be routed to the INT module. If Port A13 alternate input 0 is selected, INT4 will route from Port A13 to the INT module. To select INT4 through Port E3, set the Port E Alternate Input Register to select Port E3 alternate input 1 and set the Port A Alternate Input Register to select any Port A13 alternate input EXCEPT for selection 0 (to release the input mux priority to the Port E3 input). (Note that although EBW shows a vss input enable, other configuration options on this pad interpret this as an active-low and enable the input buffer.)

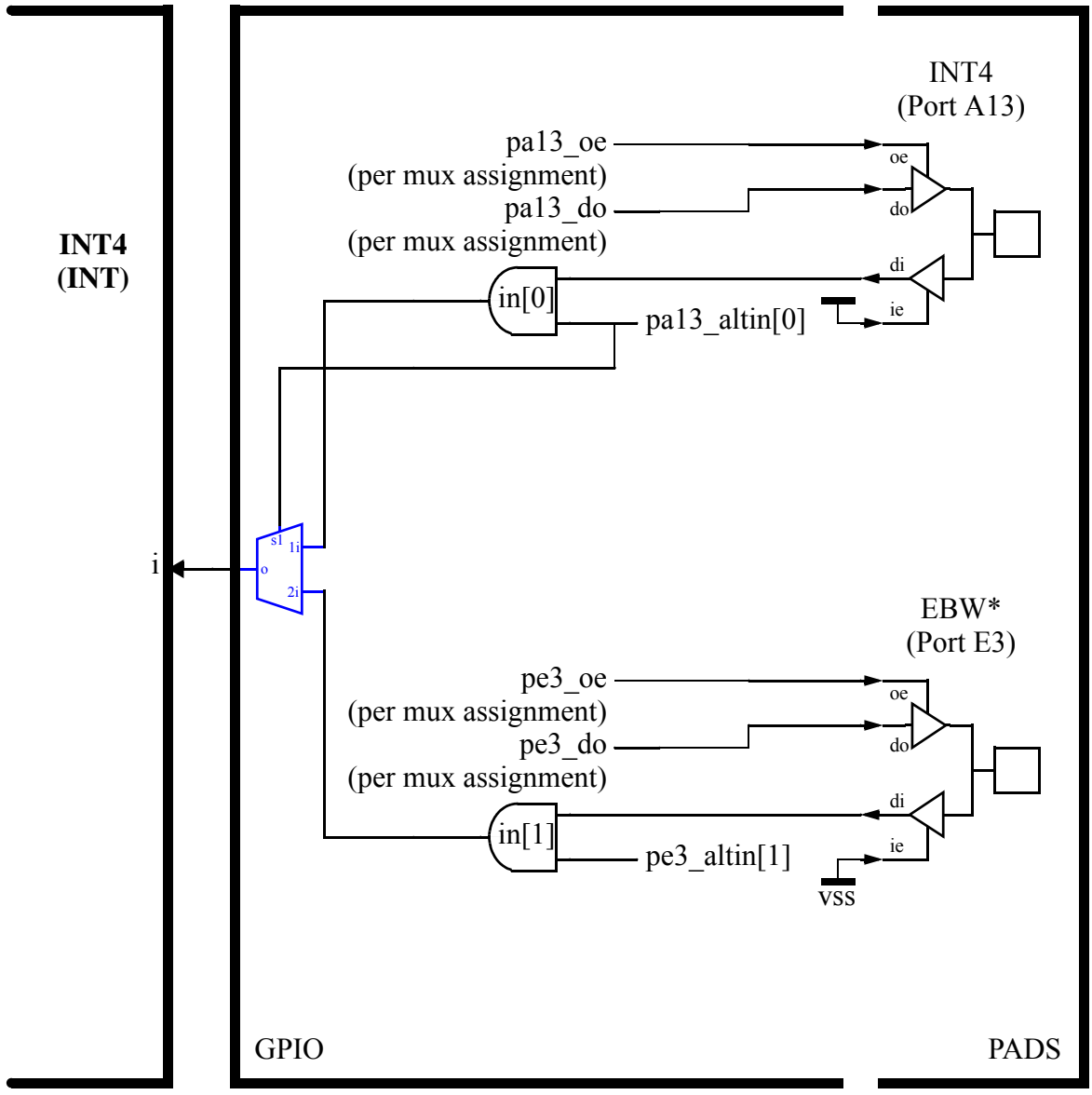


Figure 23-5.

23.12.5 Multiple INT INT5 Inputs

Since INT signal INT5 is an input signal on Port A12 and a dedicated pin INT5, a selection priority is created to determine which input will be routed to the INT module. If Port A12 alternate input 3 is selected, INT5 will route from Port A12 to the INT module. To select the dedicated INT5 pin, set the Port A Alternate Input Register to select any Port A12 alternate input EXCEPT for selection 3 (to release the input mux priority to the dedicated INT5 pin).

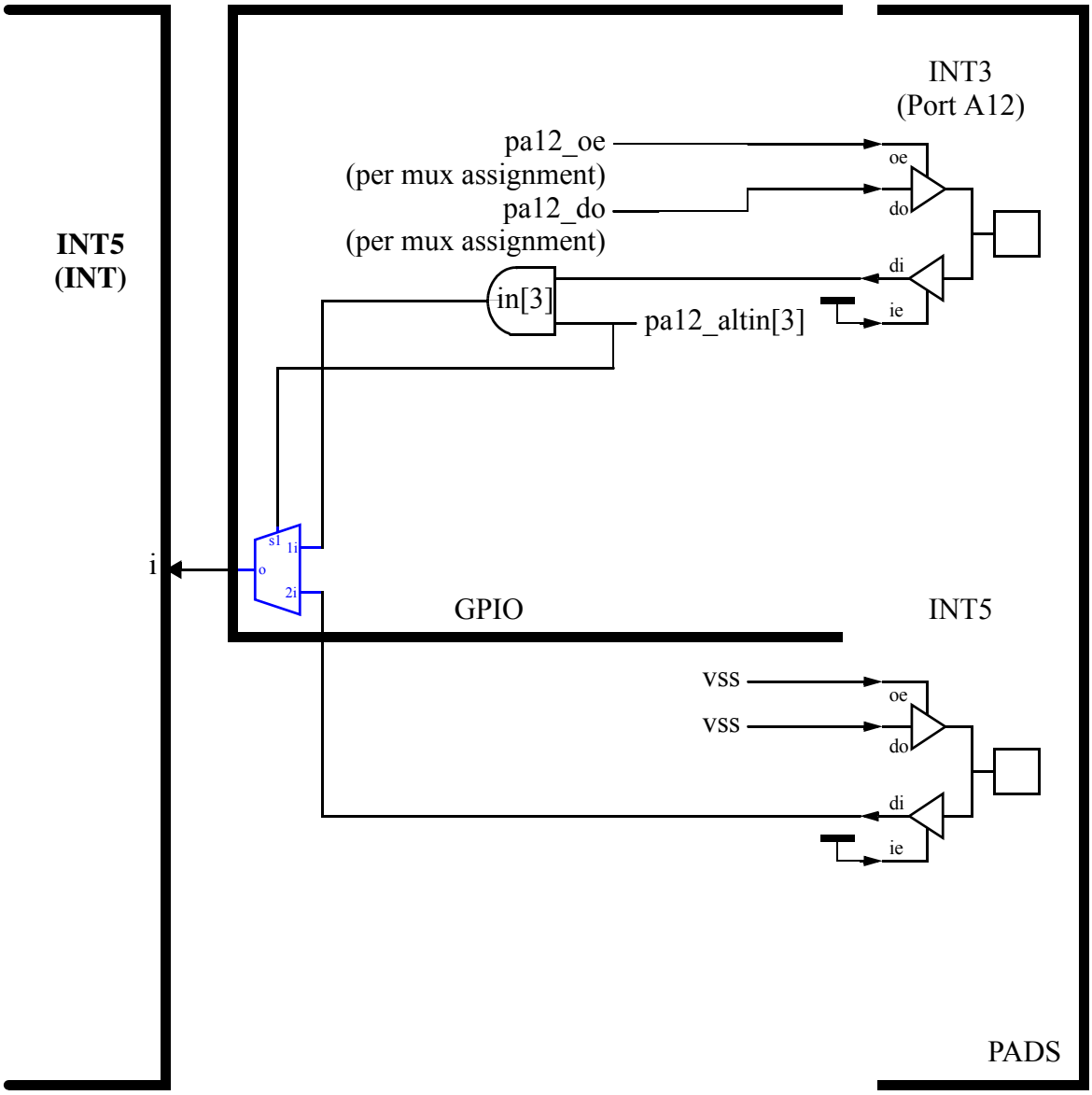


Figure 23-6.

23.12.6 Multiple UART1 DTR1 Inputs

Since UART1 signal DTR1 is an input signal on both Port D11 and Port E12, a selection priority is created to determine which input will be routed to the UART1 module. If Port D11 alternate input 2 is selected, DTR1 will route from Port D11 to the UART module. To select DTR1 through Port E12, set the Port E Alternate Input Register to select Port E12 alternate input 3 and set the Port D Alternate Input Register to select any Port D11 alternate input EXCEPT for selection 2 (to release the input mux priority to the Port E12 input).

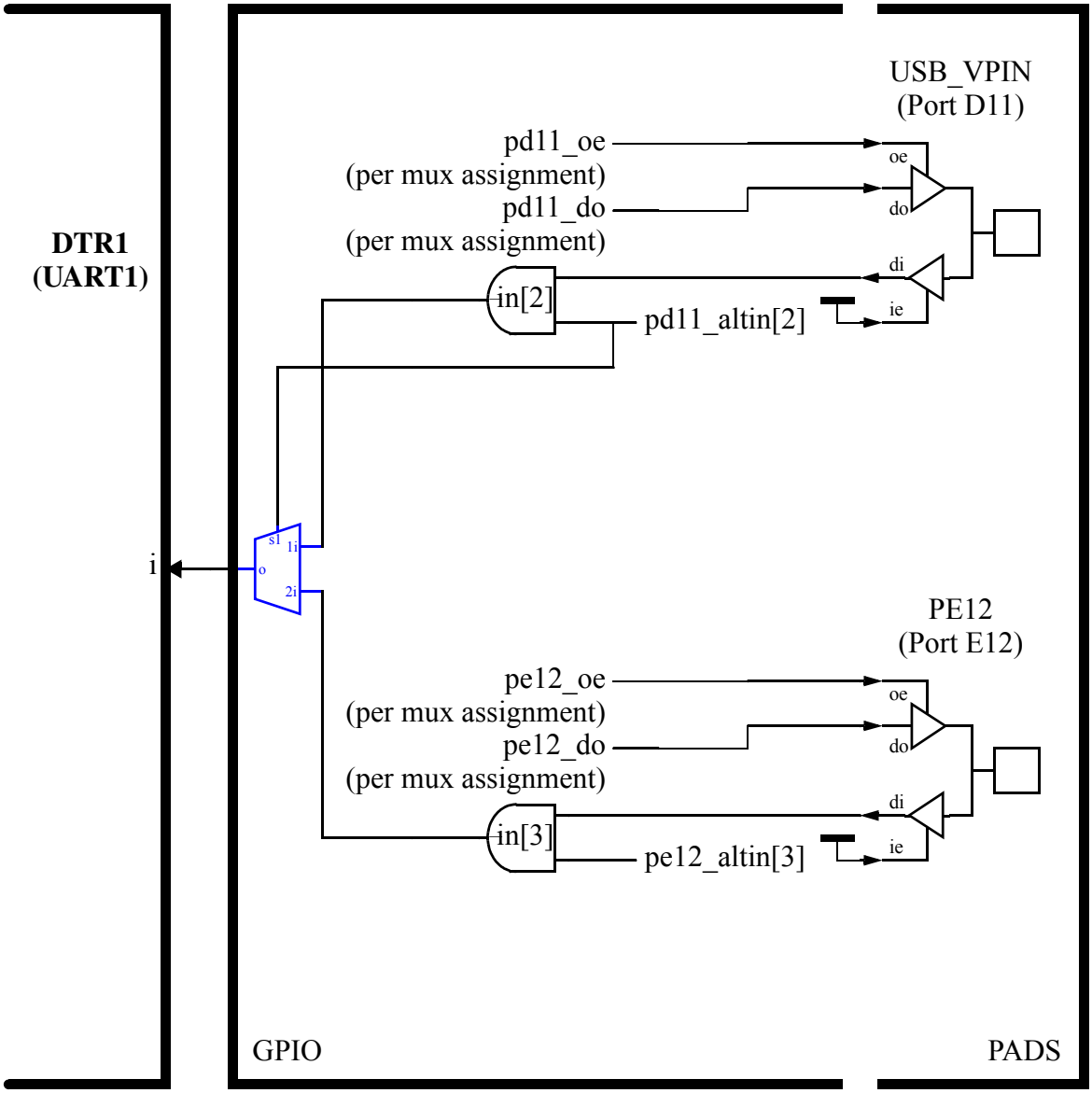


Figure 23-7.

23.12.7 Multiple UART1 RXD1 Inputs

Since UART1 signal RXD1 is an input signal on Port E11, Port D12, and dedicated input URXD1, a selection priority is created to determine which input will be routed to the UART1 module.

- If Port E11 alternate input 1 is selected, RXD1 will route from Port E11 to the UART1 module.
- To select RXD1 through Port D12, set the Port D Alternate Input Register to select Port D12 alternate input 2 and set the Port E Alternate Input Register to select any Port E11 alternate input EXCEPT for selection 1 (to release the input mux priority to the Port D12 input).
- To select RXD1 through the dedicated URXD1 pin, set the Port E Alternate Input Register to select any Port E11 alternate input EXCEPT for selection 1 and set the Port D Alternate Input Register to select any Port D12 alternate input EXCEPT for selection 2 (to release the input mux priority to the dedicated URXD1 pin).

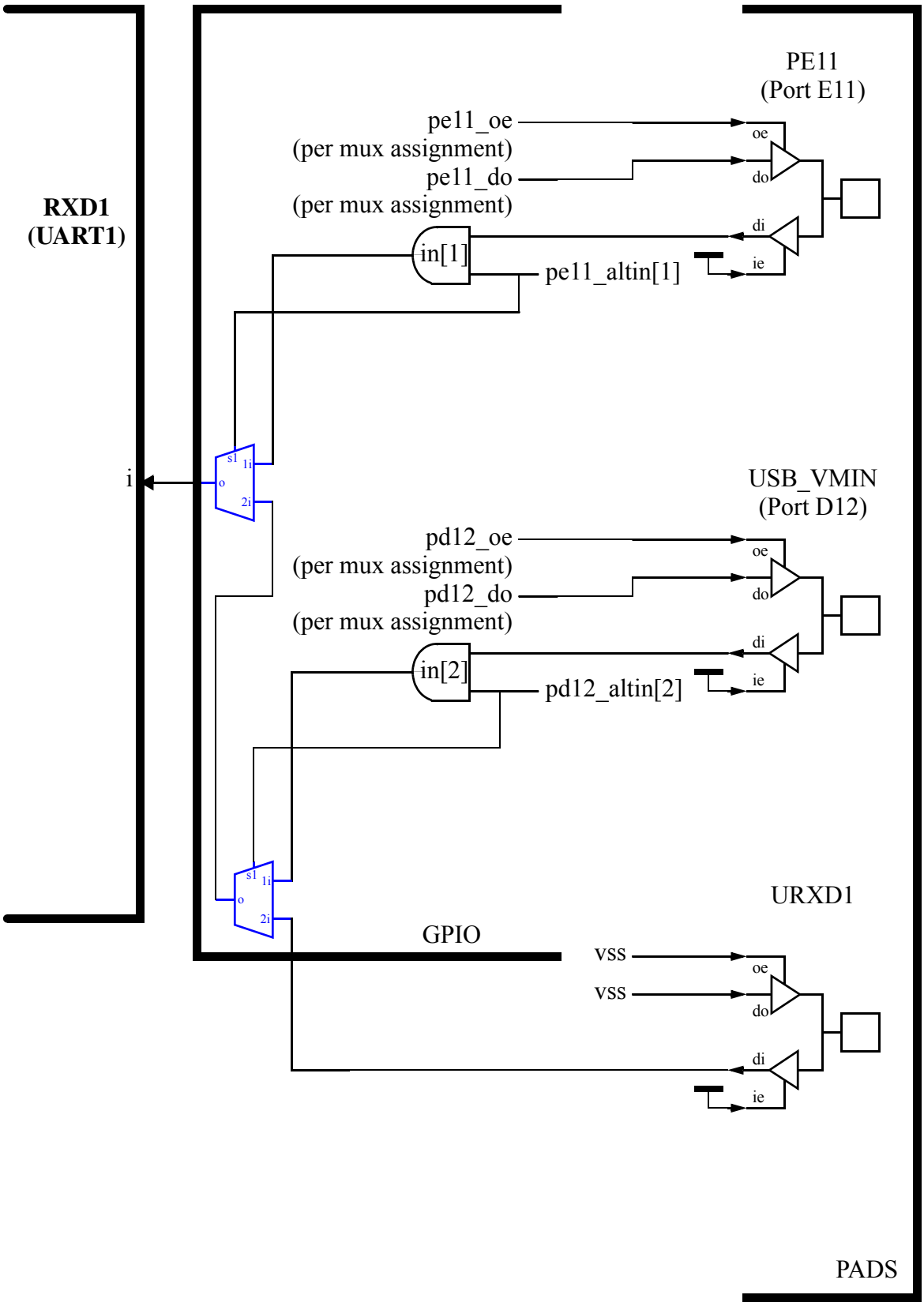

PE11
(Port E11)
pe11_oe
(per mux assignment)
pe11_do
(per mux assignment)
oe
do
di
ie
in[1]
pe11_altin[1]
RXD1
(UART1)
i
s1
1i
o
2i
USB_VMIN
(Port D12)
pd12_oe
(per mux assignment)
pd12_do
(per mux assignment)
in[2]
pd12_altin[2]
URXD1
GPIO
VSS
VSS
PADS

Figure 23-8.

23.12.8 Multiple UART1 RTS1 Inputs

Since UART1 signal RTS1 is an input signal on Port E12, Port E0, and Port D13, a selection priority is created to determine which input will be routed to the UART1 module.

- If Port E12 alternate input 1 is selected, RTS1 will route from Port E12 to the UART1 module.
- To select RTS1 through Port E0, set the Port E Alternate Input Register to select Port E0 alternate input 0 and set the Port E Alternate Input Register to select any Port E12 alternate input EXCEPT for selection 1 (to release the input mux priority to the Port E0 input).
- To select RTS1 through Port D13, set the Port D Alternate Input Register to select Port D13 alternate input 2 and set the Port E Alternate Input Register to select any Port E12 alternate input EXCEPT for selection 1 and to select any Port E0 alternate input EXCEPT for selection 0 (to release the input mux priority to Port D13).

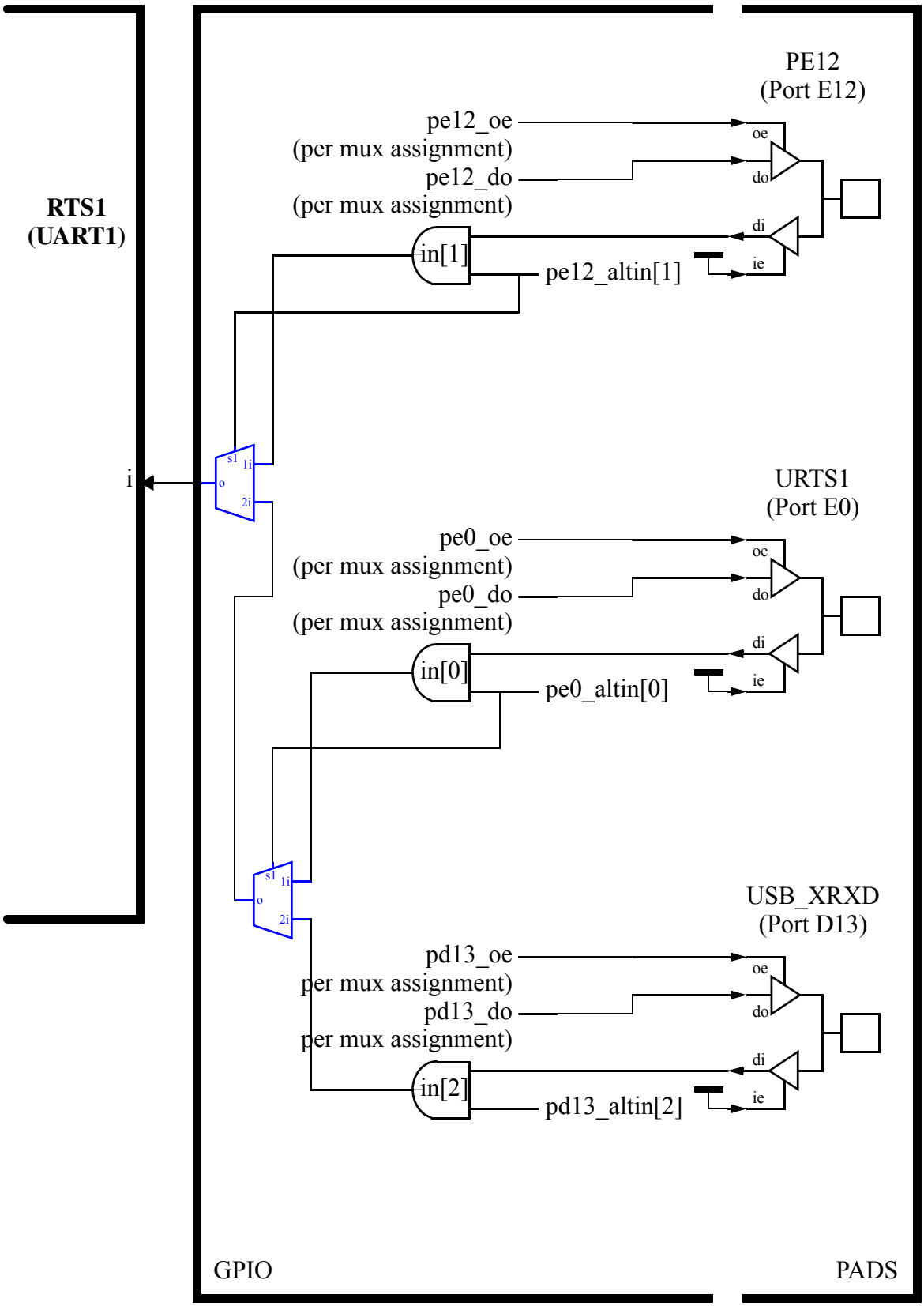

RTS1
(UART1)
i
PE12
(Port E12)
pe12_oe
(per mux assignment)
pe12_do
(per mux assignment)
oe
do
di
ie
in[1]
pe12_altin[1]
s1
1i
o
2i
URTS1
(Port E0)
pe0_oe
(per mux assignment)
pe0_do
(per mux assignment)
in[0]
pe0_altin[0]
USB_XRXD
(Port D13)
pd13_oe
per mux assignment)
pd13_do
per mux assignment)
in[2]
pd13_altin[2]
GPIO
PADS

Figure 23-9.

23.12.9 Multiple MQSPI MISOB Inputs

Since MQSPI signal MISOB is an input signal on both Port A12 and Port E14, a selection priority is created to determine which input will be routed to the MQSPI module. If Port A12 alternate input 2 is selected, MISOB will route from Port A12 to the MQSPI module. To select MISOB through Port E14, set the Port E Alternate Input Register to select Port E14 alternate input 0 and set the Port A Alternate Input Register to select any Port A12 alternate input EXCEPT for selection 2 (to release the input mux priority to the Port E14 input).

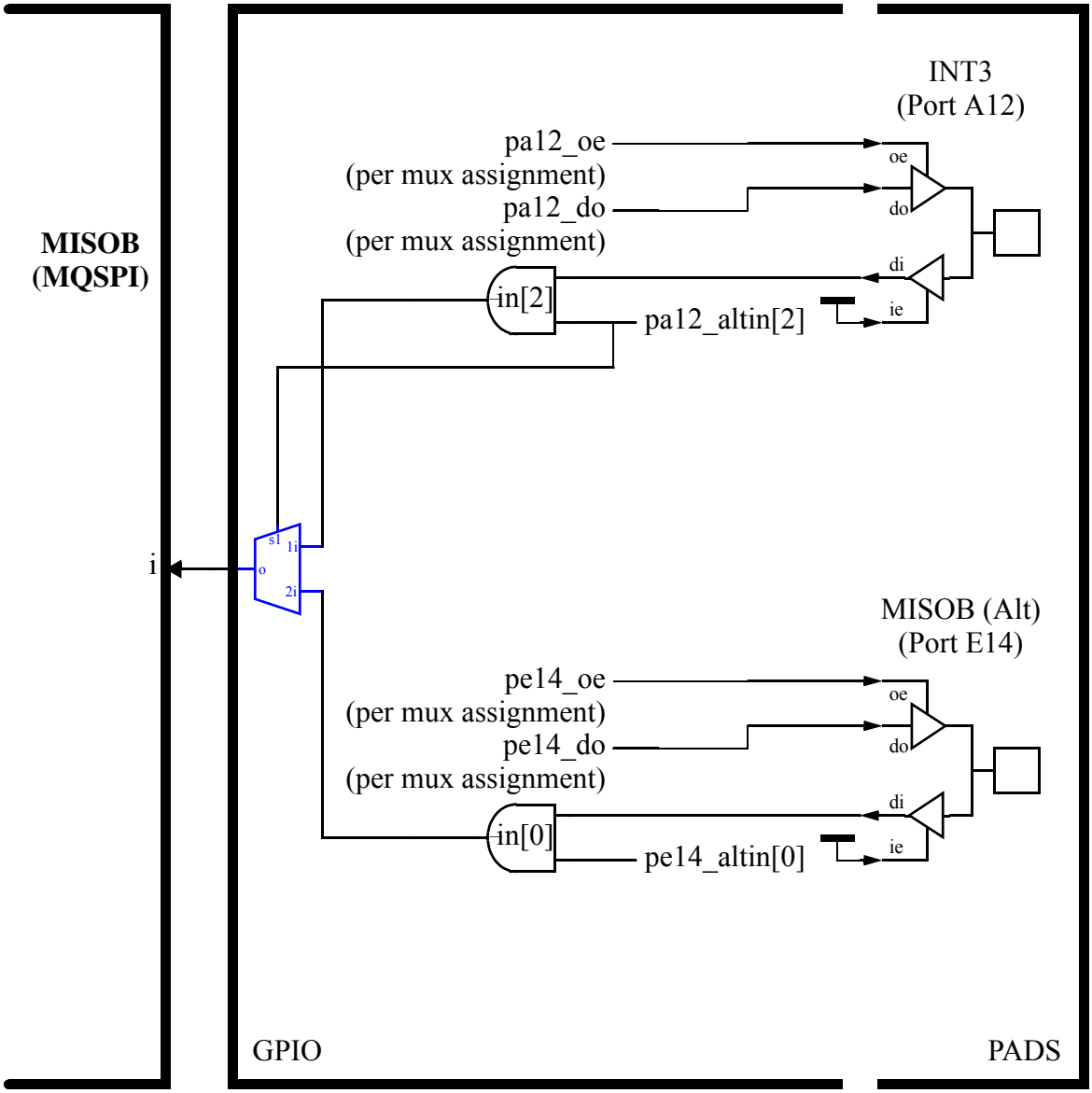


Figure 23-10.

23.13 Internal Interrupt Visibility Table

The Table of Interrupt Visibility is derivative dependent and is defined in *Chapter 6: Chip Configuration and Memory Maps.*

Note: The rxsdg's det_flag_b interrupt cannot be viewed through the gpio, but it can be viewed on the analog test pins under control of the ANATEST module.

Table 23-2. Internal Interrupt Table, Port B and Port E

	GPIO Pinout 0	GPIO Pinout 1	GPIO Pinout 2	GPIO Pinout 3	GPIO Pinout 4
Interrupts	Pin B4 & E4	Pin B3 & E5	Pin B2 & E6	Pin B1 & E7	Pin B0 and E8
63	DMA2	DMA2	DMA2	DMA2	DMA2
62	SCC SMN	SCC SMN	SCC SMN	SCC SMN	SCC SMN
61	SCC SCM	SCC SCM	SCC SCM	SCC SCM	SCC SCM
60	HACC IRQ	HACC IRQ	HACC IRQ	HACC IRQ	HACC IRQ
59	MDI OR'd	MDI OR'd	MDI OR'd	MDI OR'd	MDI OR'd
58	External 7	External 7	External 7	External 7	External 7
57	MDI general1	MDI general1	MDI general1	MDI general1	MDI general1
56	MQSPI Port 2	MQSPI Port 2	MQSPI Port 2	MQSPI Port 2	MQSPI Port 2
55	DSP DTIMER	DSP DTIMER	DSP DTIMER	DSP DTIMER	DSP DTIMER
54	Keypad	Keypad	Keypad	Keypad	Keypad
53	DSP L1TIMER	DSP L1TIMER	DSP L1TIMER	DSP L1TIMER	DSP L1TIMER
52	VIAC	VIAC	VIAC	VIAC	VIAC
51	SIM GENERAL	SIM GENERAL	SIM GENERAL	SIM GENERAL	SIM GENERAL
50	External 6	External 6	External 6	External 6	External 6
49	EPIT	EPIT	EPIT	EPIT	EPIT
48	EGPT	EGPT	EGPT	EGPT	EGPT
47	DSP USB	DSP USB	DSP USB	DSP USB	DSP USB
46	MDI General 0	MDI General 0	MDI General 0	MDI General 0	MDI General 0
45	UART2 RTS	UART2 RTS	UART2 RTS	UART2 RTS	UART2 RTS
44	VOCOD	VOCOD	VOCOD	VOCOD	VOCOD
43	DMA 1	DMA 1	DMA 1	DMA 1	DMA 1
42	UART2 Common	UART2 Common	UART2 Common	UART2 Common	UART2 Common
41	External 5	External 5	External 5	External 5	External 5
40	LEM	LEM	LEM	LEM	LEM
39	PDDM	PDDM	PDDM	PDDM	PDDM

Table 23-2. Internal Interrupt Table, Port B and Port E (Continued)

	GPIO Pinout 0	GPIO Pinout 1	GPIO Pinout 2	GPIO Pinout 3	GPIO Pinout 4
Interrupts	Pin B4 & E4	Pin B3 & E5	Pin B2 & E6	Pin B1 & E7	Pin B0 and E8
38	UART1 RTS	UART1 RTS	UART1 RTS	UART1 RTS	UART1 RTS
37	UART1 DTR	UART1 DTR	UART1 DTR	UART1 DTR	UART1 DTR
36	DMAC_IRQ	DMAC_IRQ	DMAC_IRQ	DMAC_IRQ	DMAC_IRQ
35	UART1 Common	UART1 Common	UART1 Common	UART1 Common	UART1 Common
34	USB SUSPEND, RESUME, RESET	USB SUSPEND, RESUME, RESET	USB SUSPEND, RESUME, RESET	USB SUSPEND, RESUME, RESET	USB SUSPEND, RESUME, RESET
33	GEM	GEM	GEM	GEM	GEM
32	External 4	External 4	External 4	External 4	External 4
31	GPIO Port E	GPIO Port E	GPIO Port E	GPIO Port E	GPIO Port E
30	BBP	BBP	BBP	BBP	BBP
29	MQSPI Port 1	MQSPI Port 1	MQSPI Port 1	MQSPI Port 1	MQSPI Port 1
28	External 3	External 3	External 3	External 3	External 3
27	SAP	SAP	SAP	SAP	SAP
26	GPIO Port D	GPIO Port D	GPIO Port D	GPIO Port D	GPIO Port D
25	MDI rx1/tx1 or'd	MDI rx1/tx1 or'd	MDI rx1/tx1 or'd	MDI rx1/tx1 or'd	MDI rx1/tx1 or'd
24	External 2	External 2	External 2	External 2	External 2
23	DSP MDI Command	DSP MDI Command	DSP MDI Command	DSP MDI Command	DSP MDI Command
22	GPIO Port C	GPIO Port C	GPIO Port C	GPIO Port C	GPIO Port C
21	MDI rx0/tx0 or'd	MDI rx0/tx0 or'd	MDI rx0/tx0 or'd	MDI rx0/tx0 or'd	MDI rx0/tx0 or'd
20	L1T MCU_INT1	L1T MCU_INT1	L1T MCU_INT1	L1T MCU_INT1	L1T MCU_INT1
19	L1T RFI	L1T RFI	L1T RFI	L1T RFI	L1T RFI
18	L1T RFCI	L1T RFCI	L1T RFCI	L1T RFCI	L1T RFCI
17	L1T_CI	L1T_CI	L1T_CI	L1T_CI	L1T_CI
16	L1T MCU INT0	L1T MCU INT0	L1T MCU INT0	L1T MCU INT0	L1T MCU INT0
15	DSM	DSM	DSM	DSM	DSM
14	DSP MDI	DSP MDI	DSP MDI	DSP MDI	DSP MDI

Table 23-2. Internal Interrupt Table, Port B and Port E (Continued)

	GPIO Pinout 0	GPIO Pinout 1	GPIO Pinout 2	GPIO Pinout 3	GPIO Pinout 4
Interrupts	Pin B4 & E4	Pin B3 & E5	Pin B2 & E6	Pin B1 & E7	Pin B0 and E8
13	UART2 TX	UART2 TX	UART2 TX	UART2 TX	UART2 TX
12	SIM 1 DATA	SIM 1 DATA	SIM 1 DATA	SIM 1 DATA	SIM 1 DATA
11	UART2 RX	UART2 RX	UART2 RX	UART2 RX	UART2 RX
10	UART TX	UART TX	UART TX	UART TX	UART TX
9	External 1	External 1	External 1	External 1	External 1
8	GPIO Port B	GPIO Port B	GPIO Port B	GPIO Port B	GPIO Port B
7	USB TX	USB TX	USB TX	USB TX	USB TX
6	MCU USB RX	MCU USB RX	MCU USB RX	MCU USB RX	MCU USB RX
5	RTCA_INT	RTCA_INT	RTCA_INT	RTCA_INT	RTCA_INT
4	UART1 RX	UART1 RX	UART1 RX	UART1 RX	UART1 RX
3	External 0	External 0	External 0	External 0	External 0
2	RTR	RTR	RTR	RTR	RTR
1	GPIO Port A	GPIO Port A	GPIO Port A	GPIO Port A	GPIO Port A
0	Reserved	Reserved	Reserved	Reserved	Reserved

23.14 Port Diagrams

23.14.1 Port A Block Diagrams

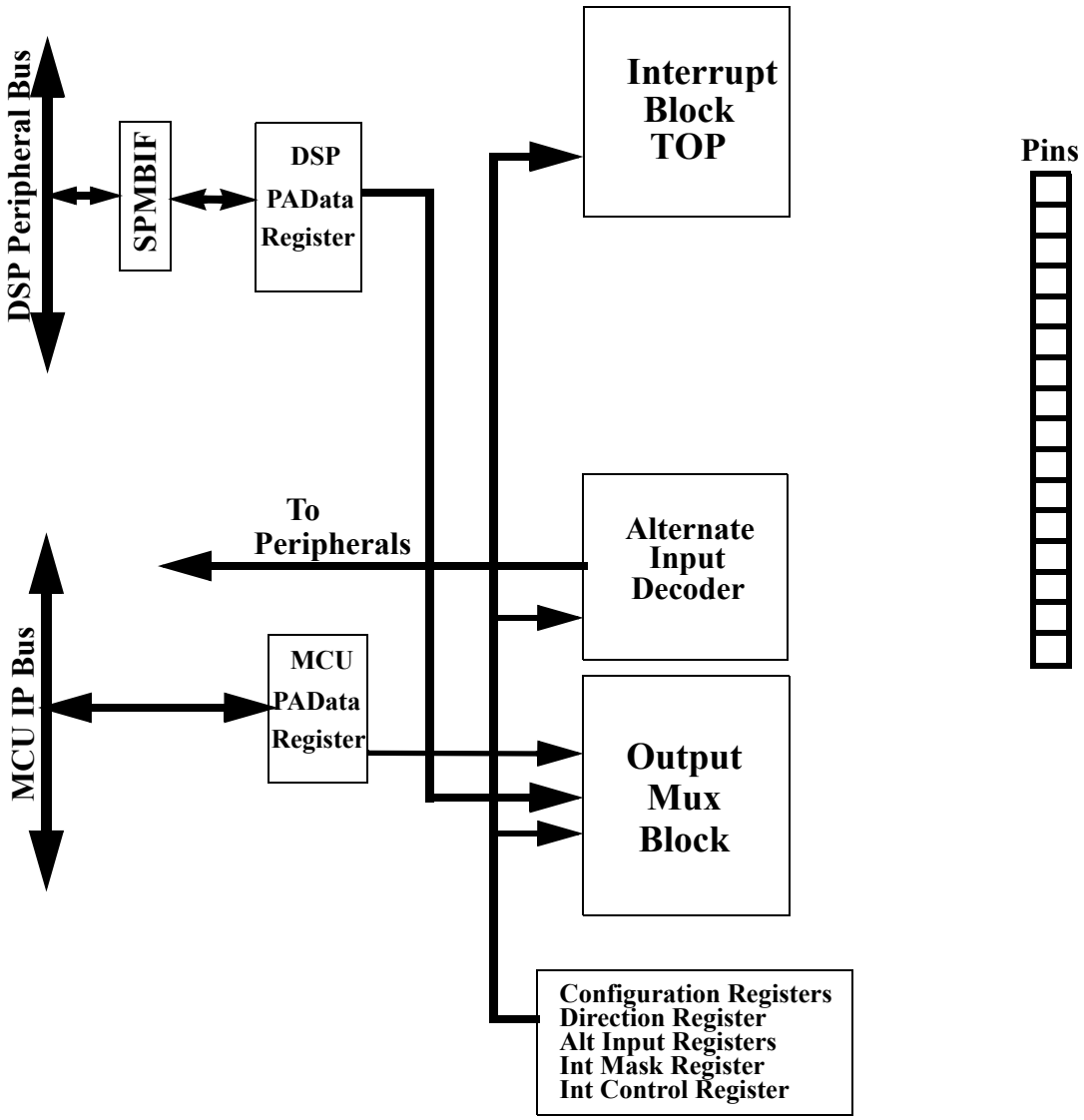


Figure 23-11. Port A Top Level Block Diagram

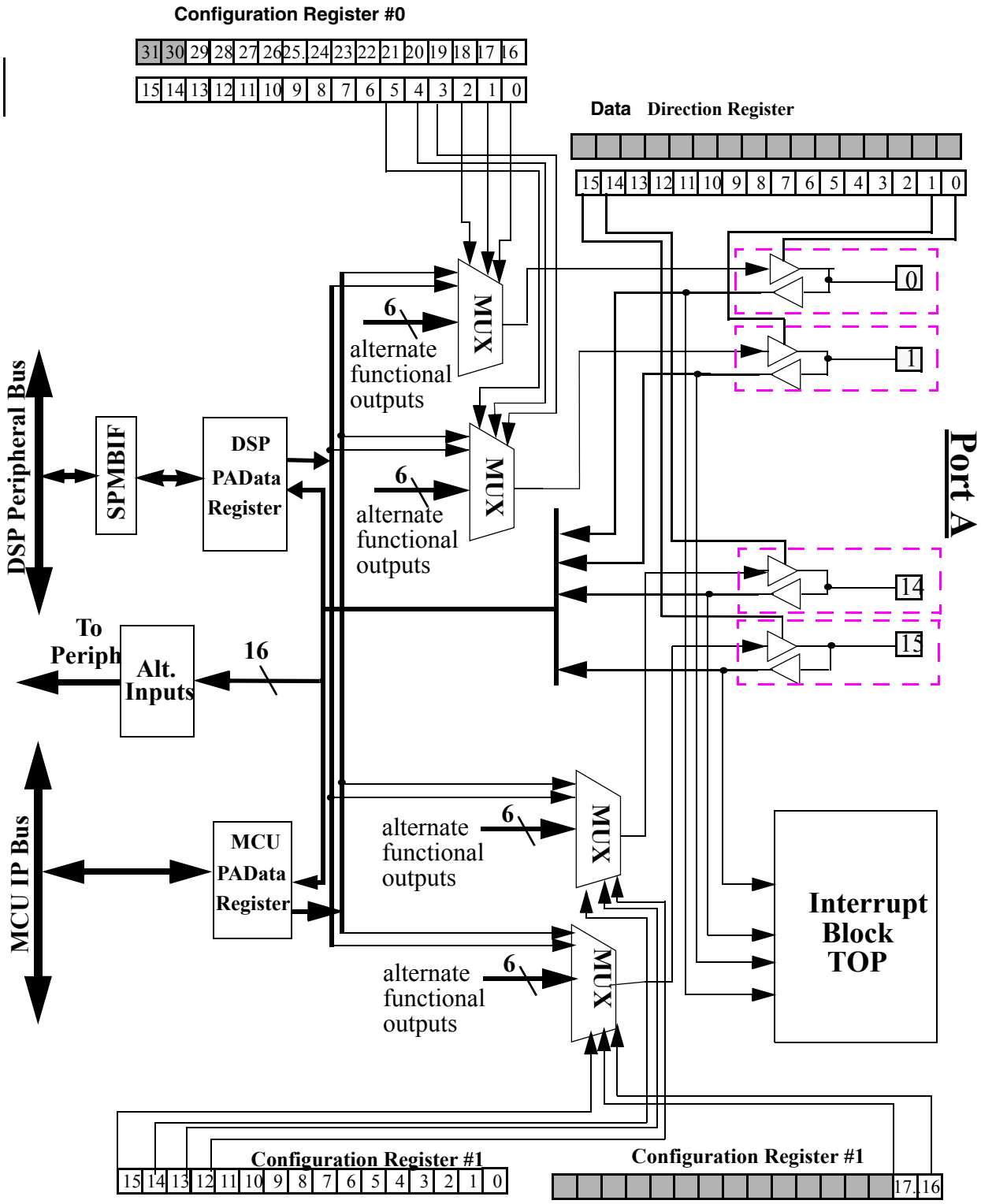


Figure 23-12. Port A Output Configuration Block Diagram

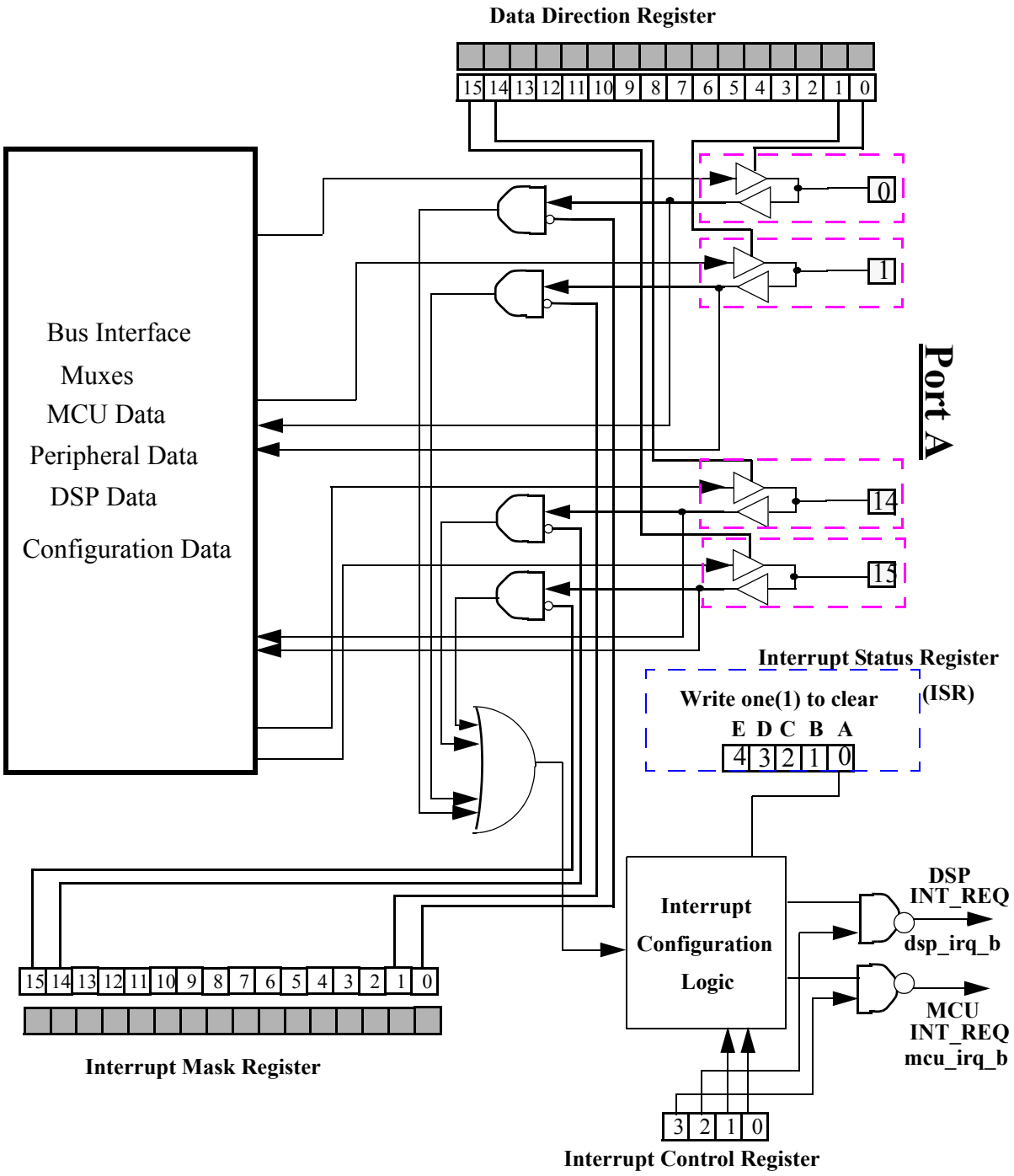


Figure 23-13. External Interrupt Control Block Diagram

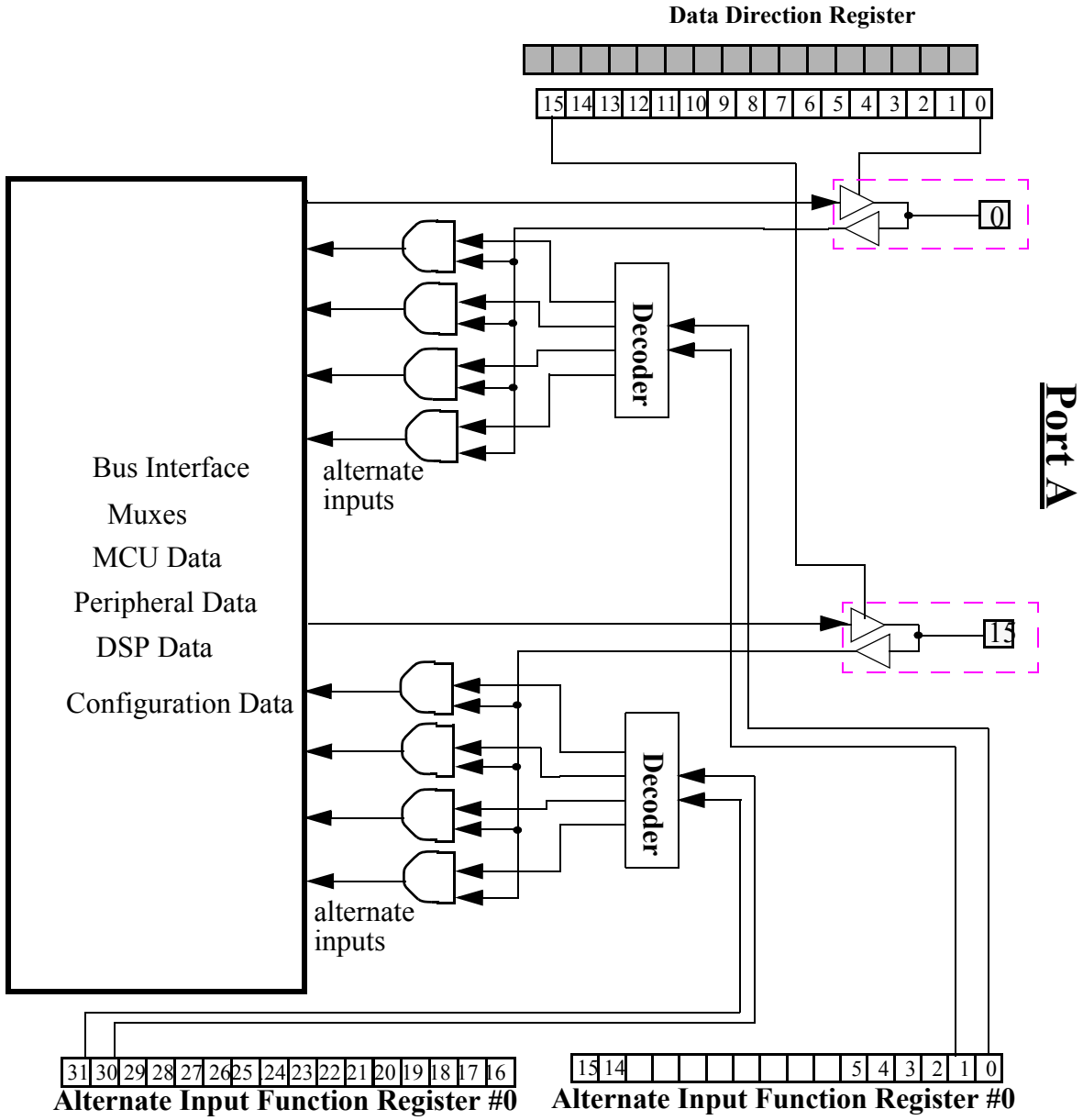


Figure 23-14. Alternate Input Functionality Block Diagram

23.14.1.1 Port A, B, C, D and E Pin Logic Diagrams

The following diagrams and paragraphs show how software and hardware configure the Port A, B, C, D and E pins to achieve the desired functionality. The exact signals connected as funcout[5:0] and altin[3:0] are configured at chip level and therefore are defined in *Chapter 6: Chip Configuration and Memory Maps.*

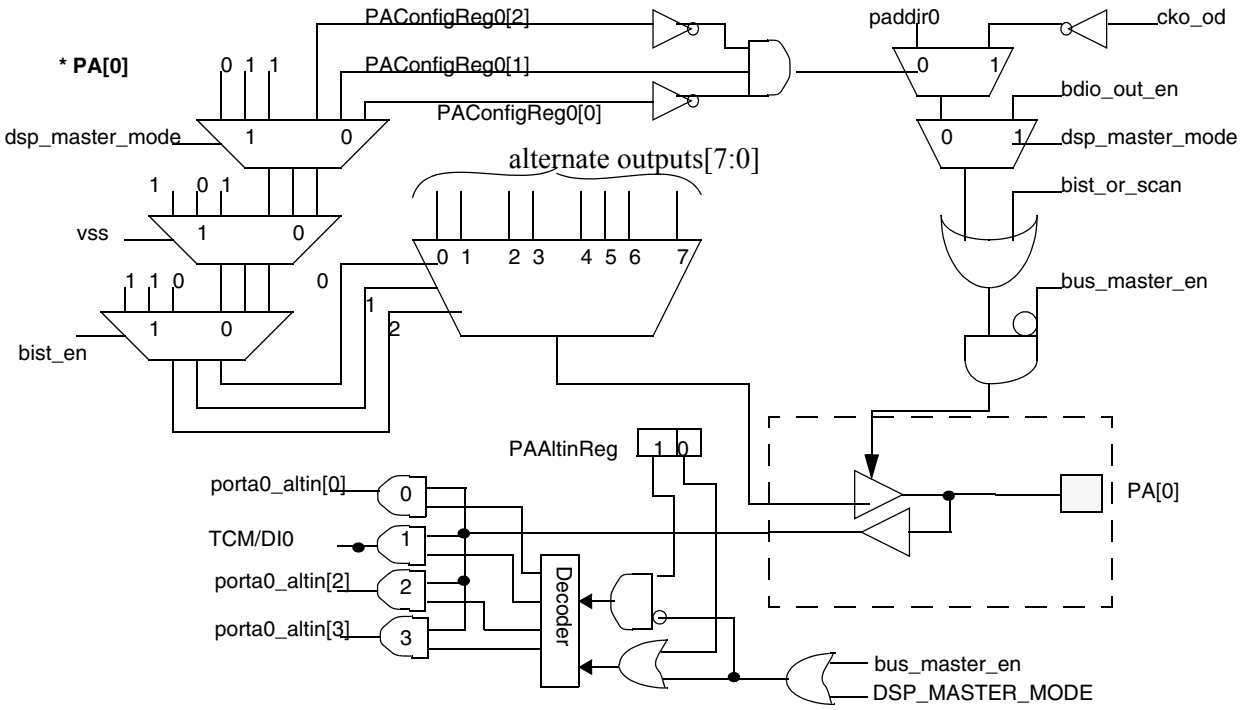


Figure 23-15. gpio_porta_funcout[3] Configured for scan_mode

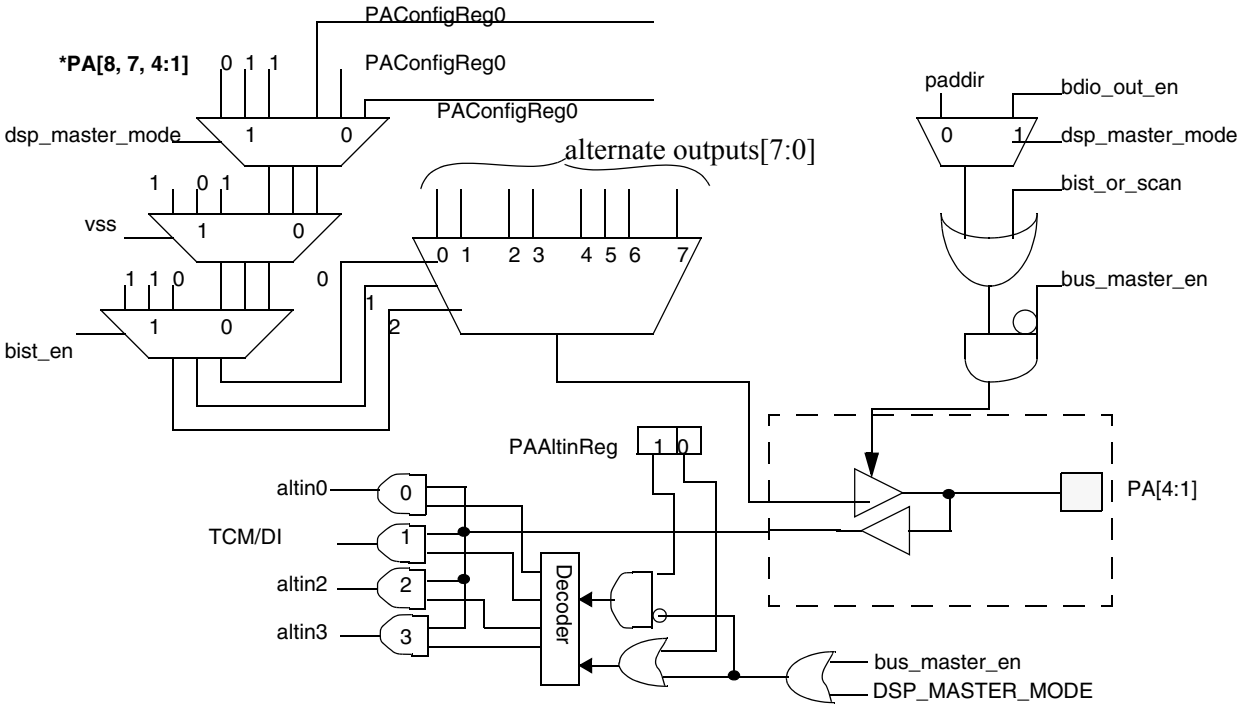


Figure 23-16. * gpio_porta_funcout[3] Configured for scan_mode

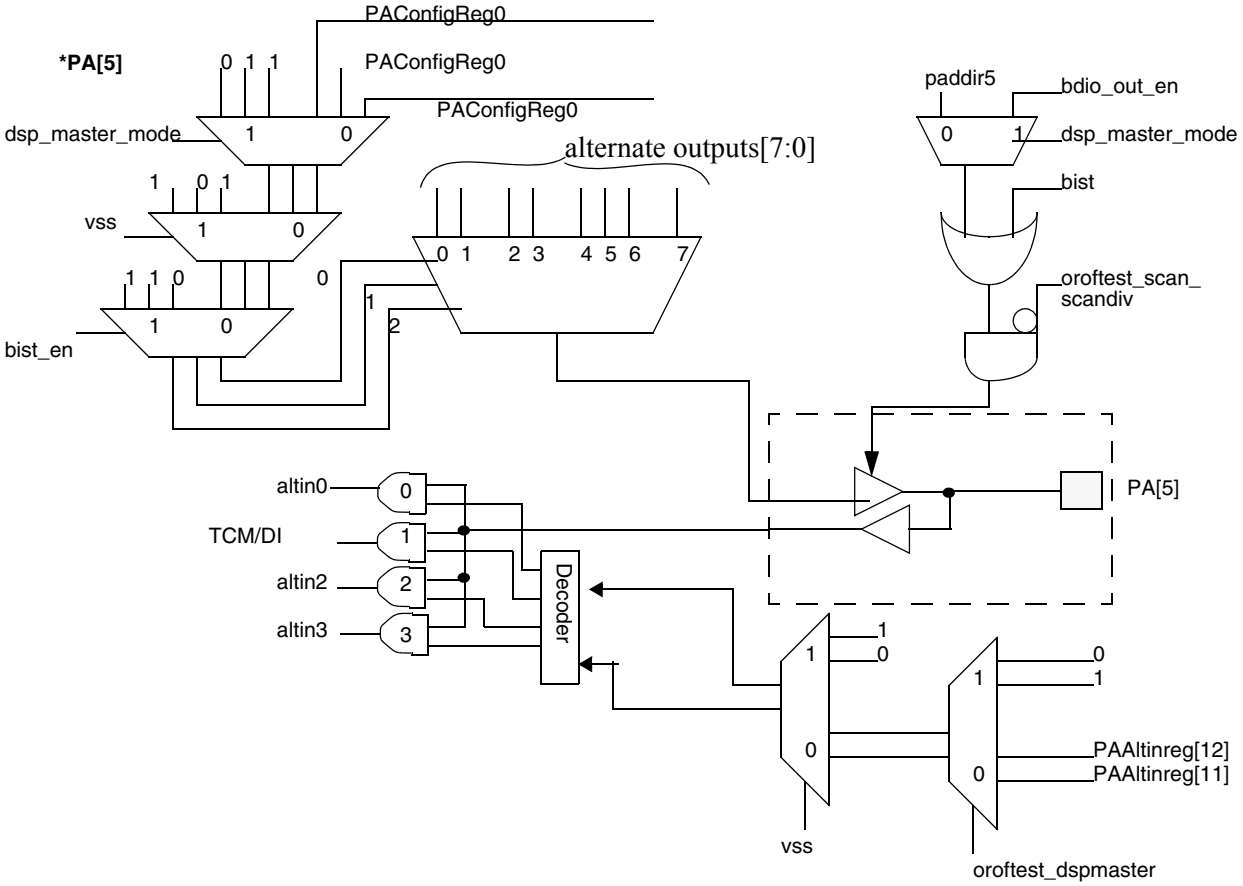


Figure 23-17. * gpio_porta_altin[2] configured for scan_mode

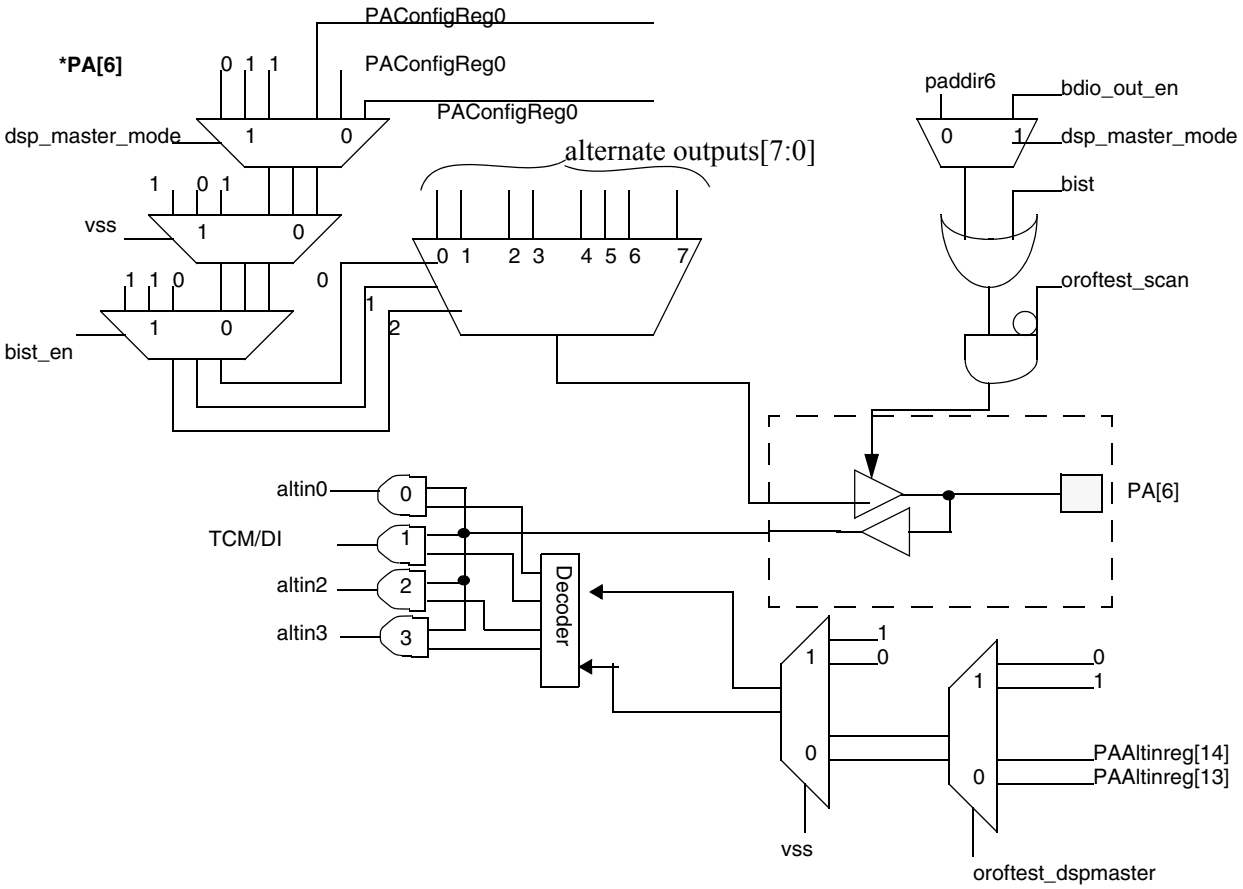


Figure 23-18. * gpio_porta_altin[2] configured for scan_mode

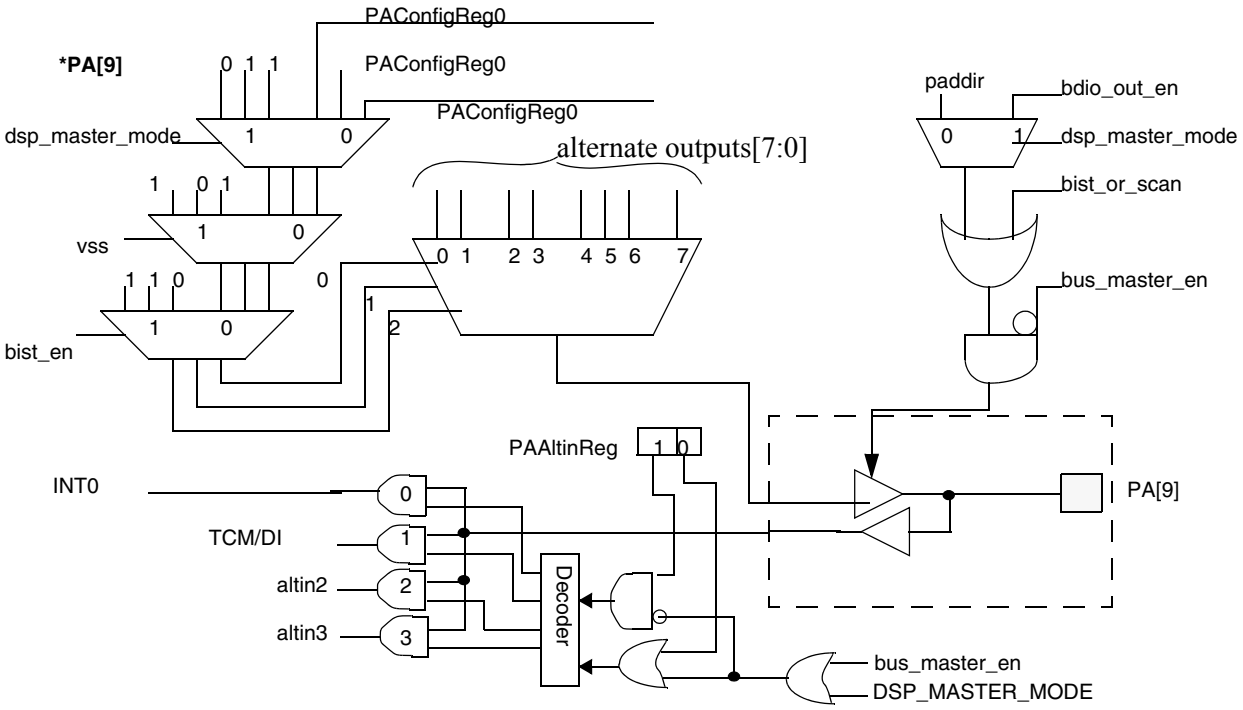


Figure 23-19. * gpio_porta_funcout[3] configured for scan_mode, refer Appendix

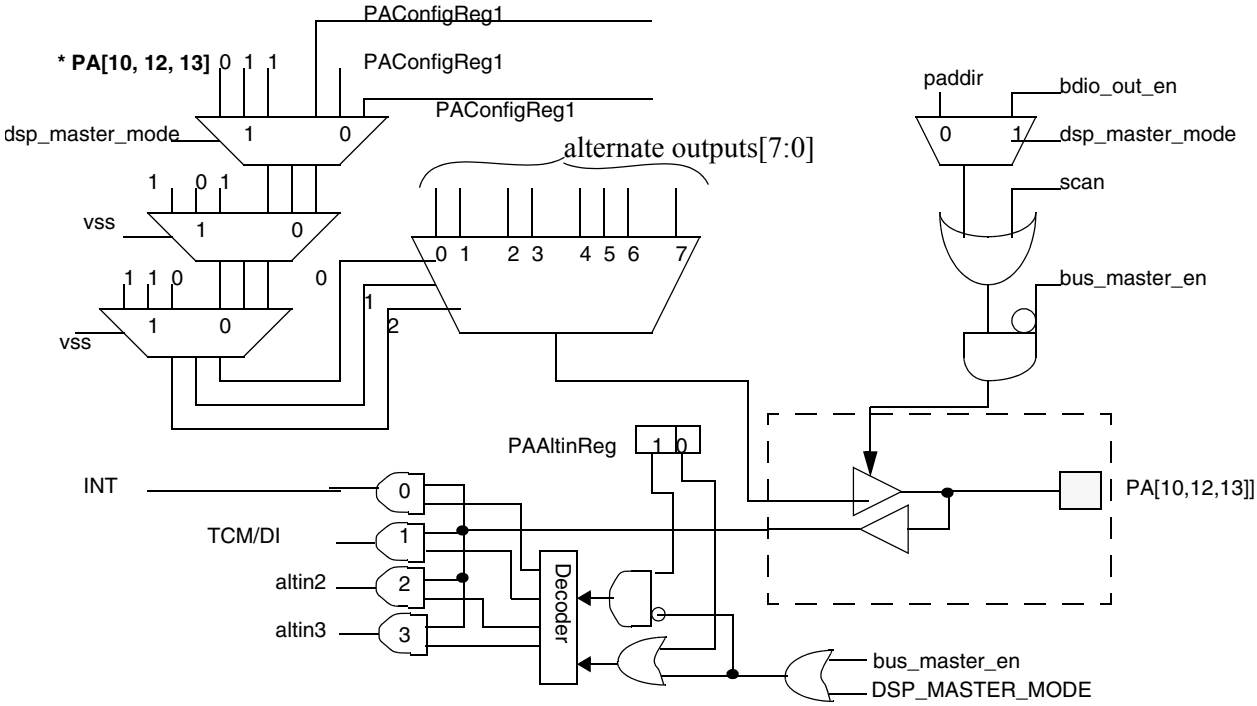


Figure 23-20. * gpio_porta_funcout[3] configured for scan_mode

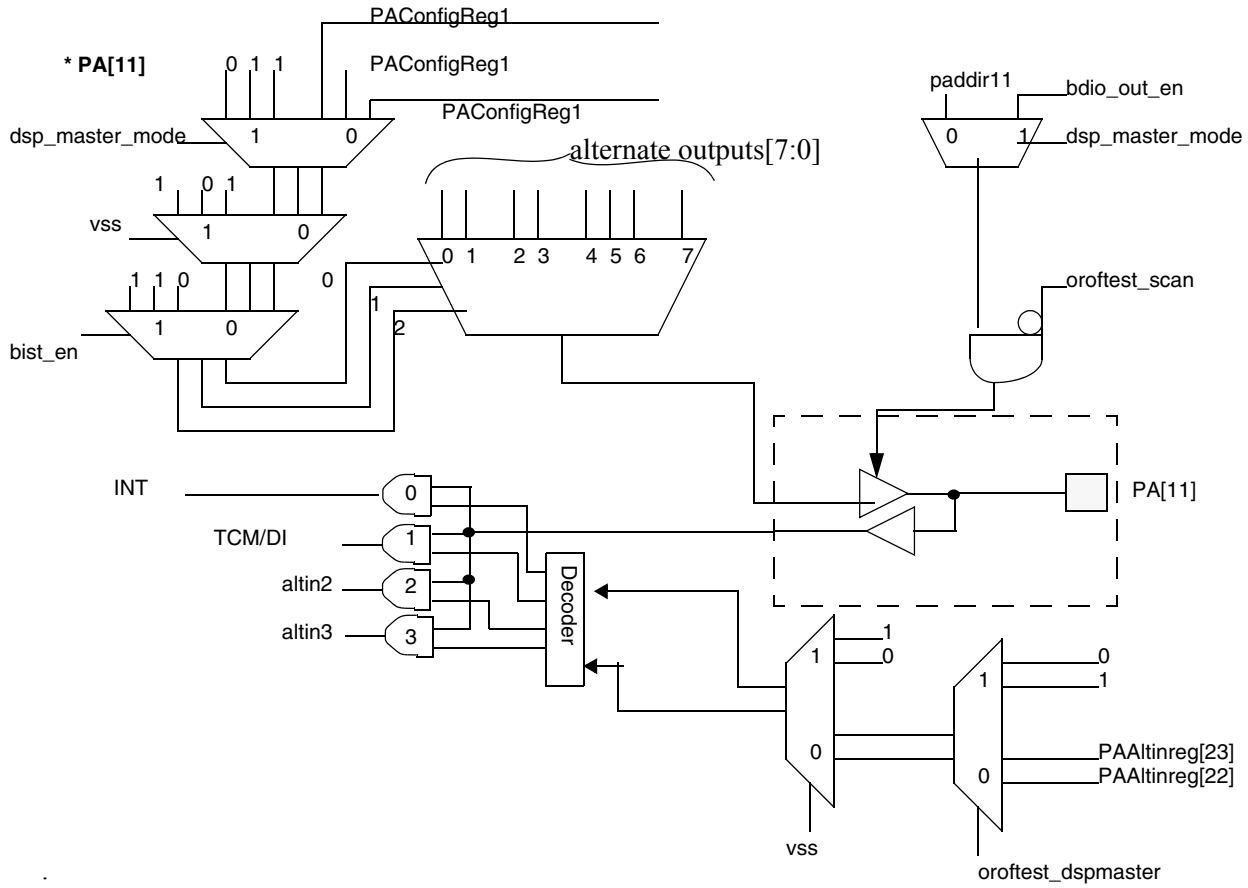


Figure 23-21. * gpio_porta_altin[2] configured for scan_mode

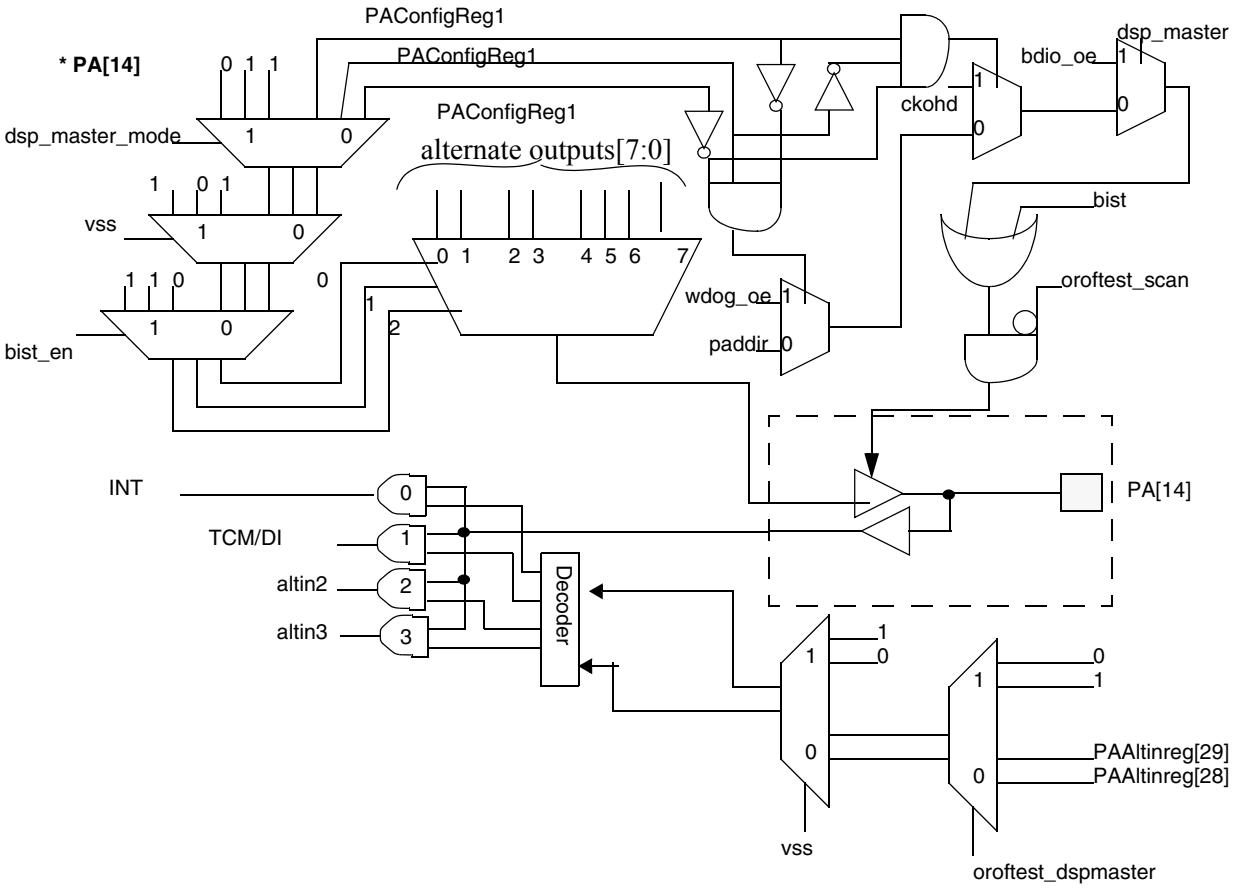


Figure 23-22. * gpio_porta_altin[2] configured for scan_mode

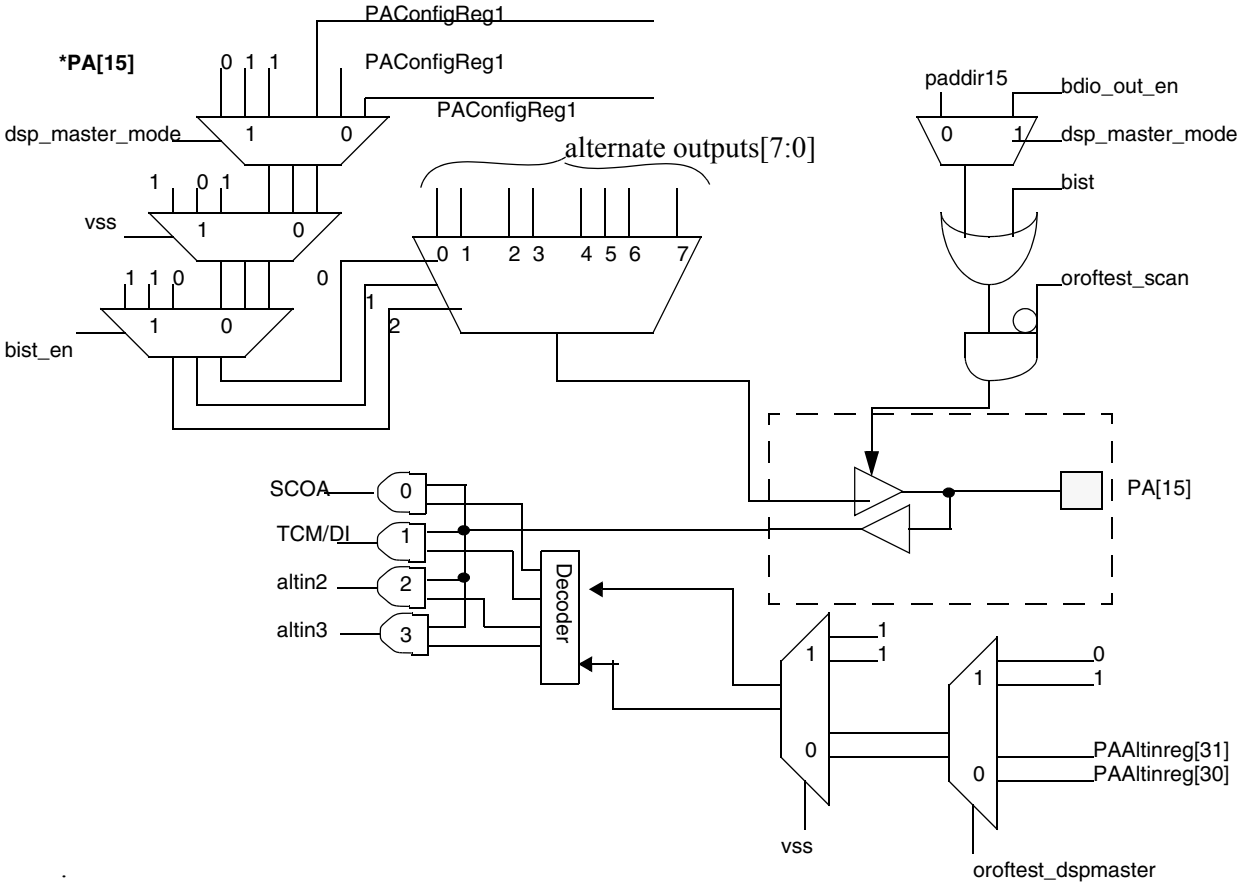


Figure 23-23. * gpio_porta_altin[3] configured for scan_mode

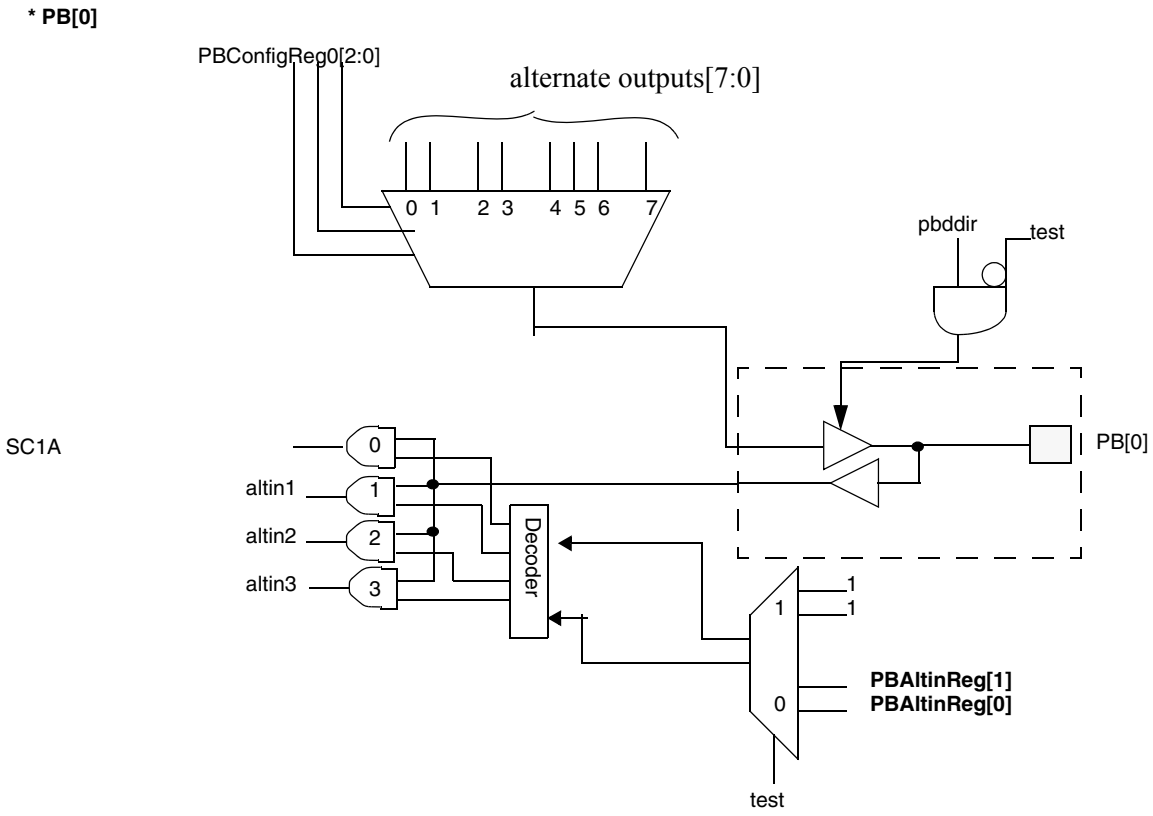


Figure 23-24. * gpio_portb_altin[3] configured for scan_mode

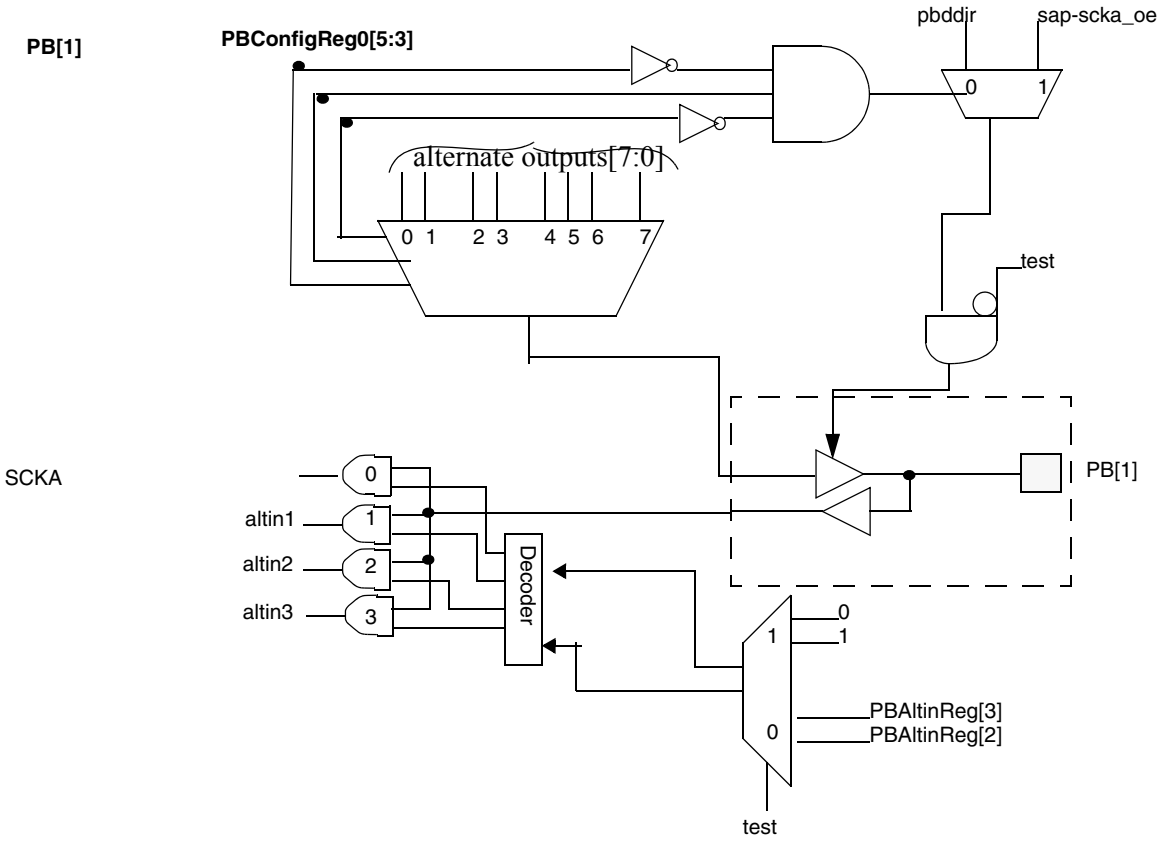

PB[1]
PBConfigReg0[5:3]
pbddir
sap-scka_oe
0
1
alternate outputs[7:0]
0 1 2 3 4 5 6 7
test
SCKA
0
altin1
1
altin2
2
altin3
3
Decoder
PB[1]
0
1
1
0
PBAltinReg[3]
PBAltinReg[2]
test

Figure 23-25.

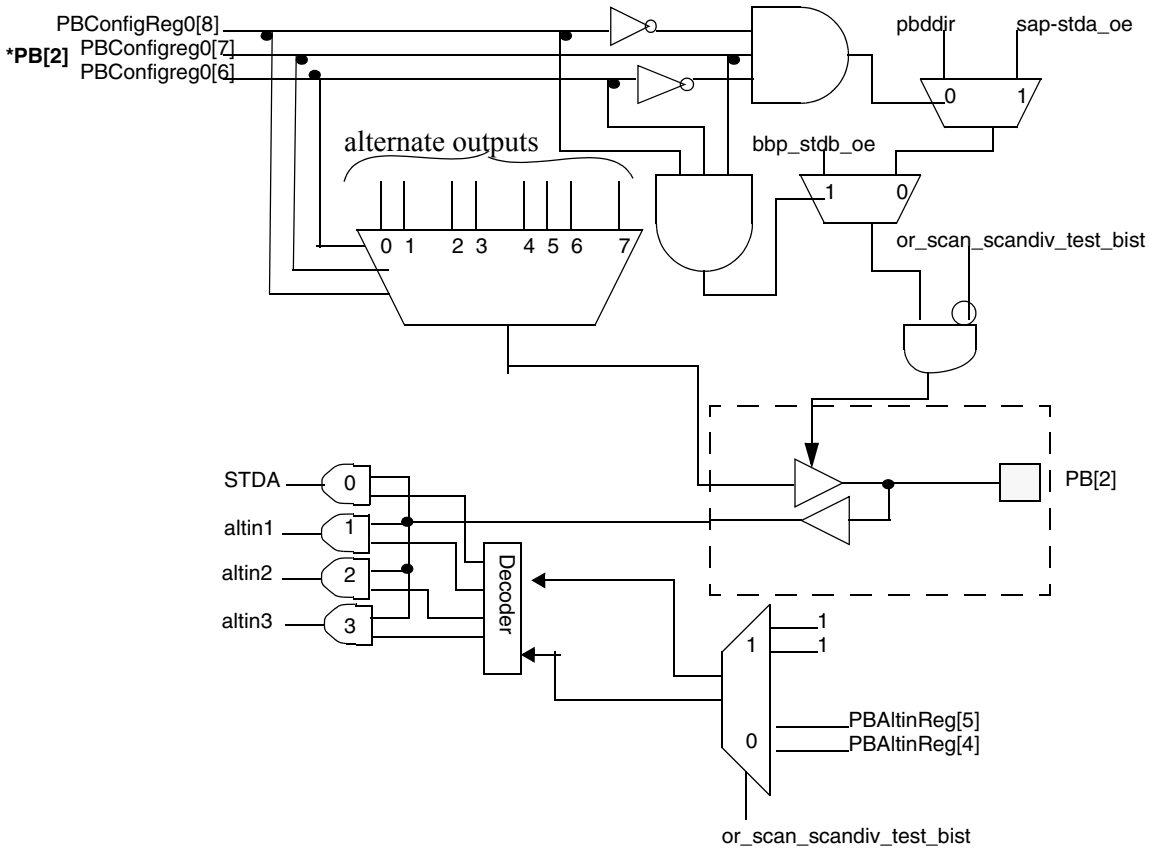


Figure 23-26. * gpio_portb_altin[3] configured for scan_mode

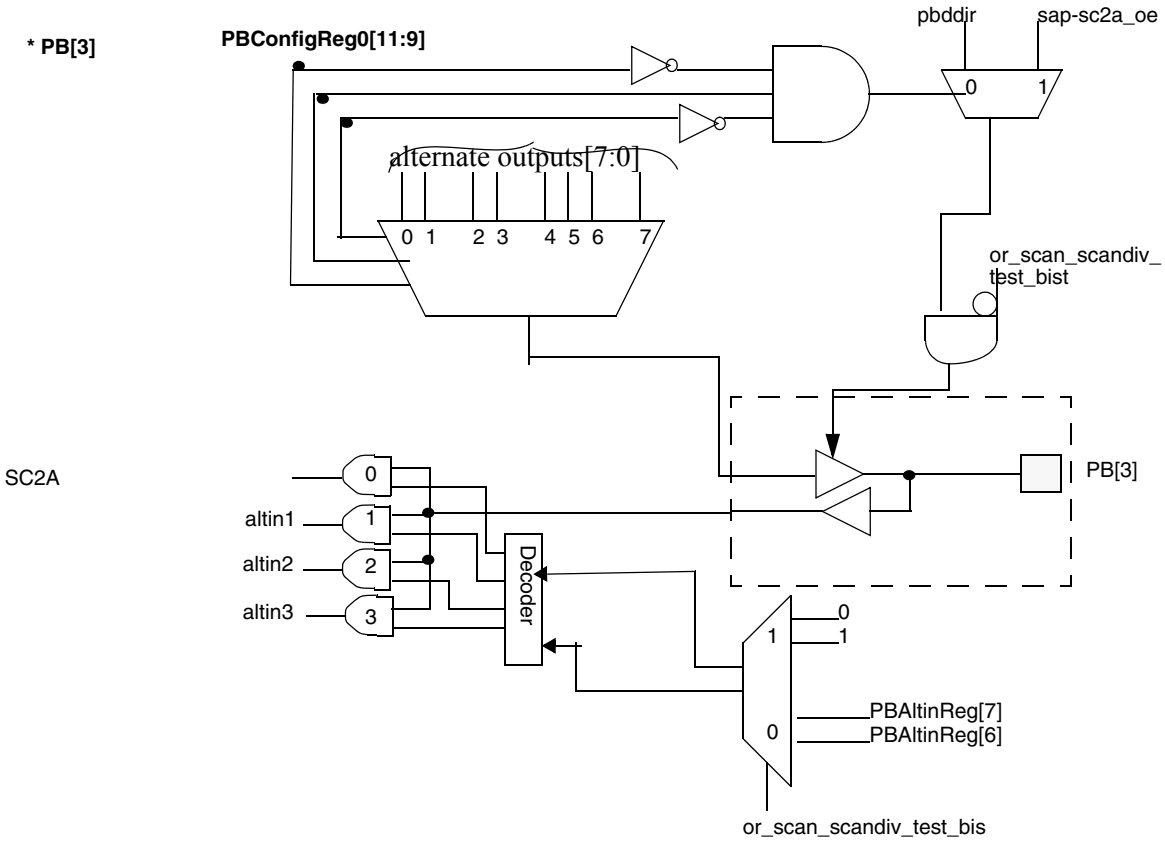


Figure 23-27. * gpio_portb_altin[3] configured for scan_mode

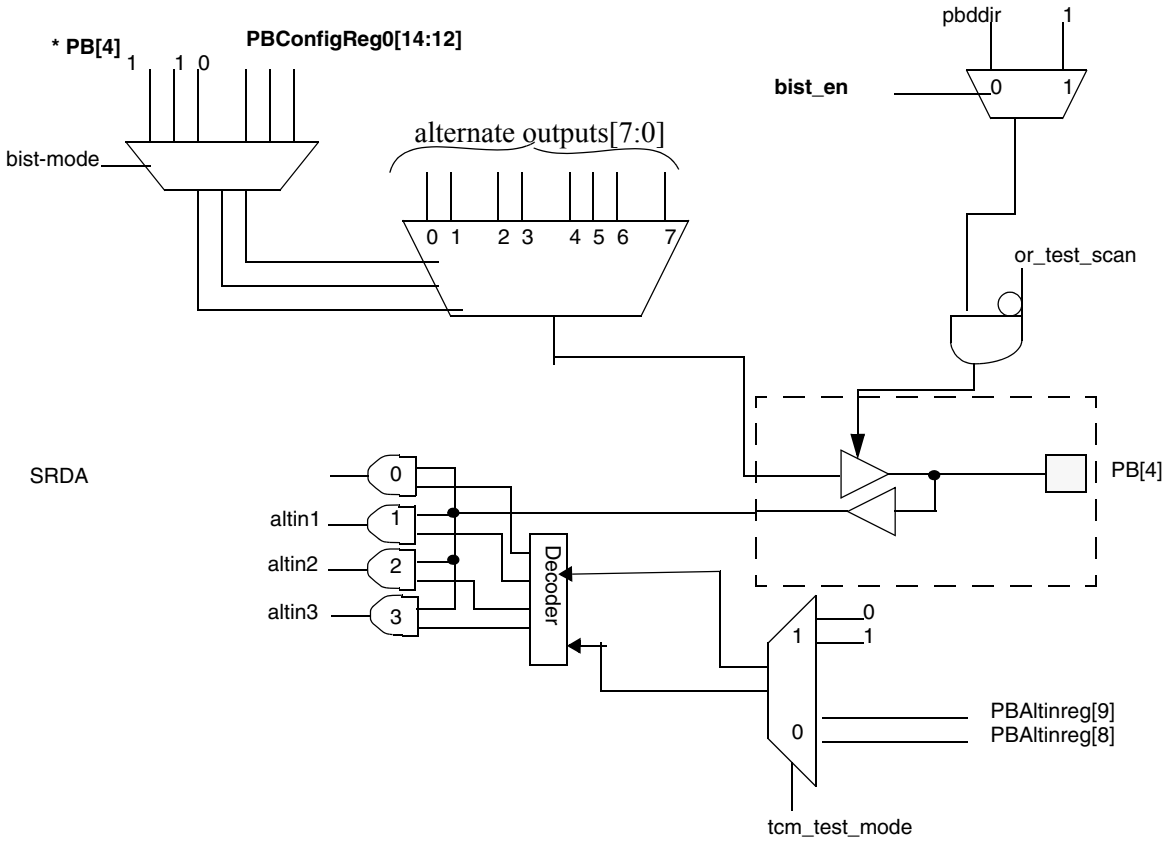


Figure 23-28. * gpio_portb_altin[3] configured for scan_mode

PB [5:6]

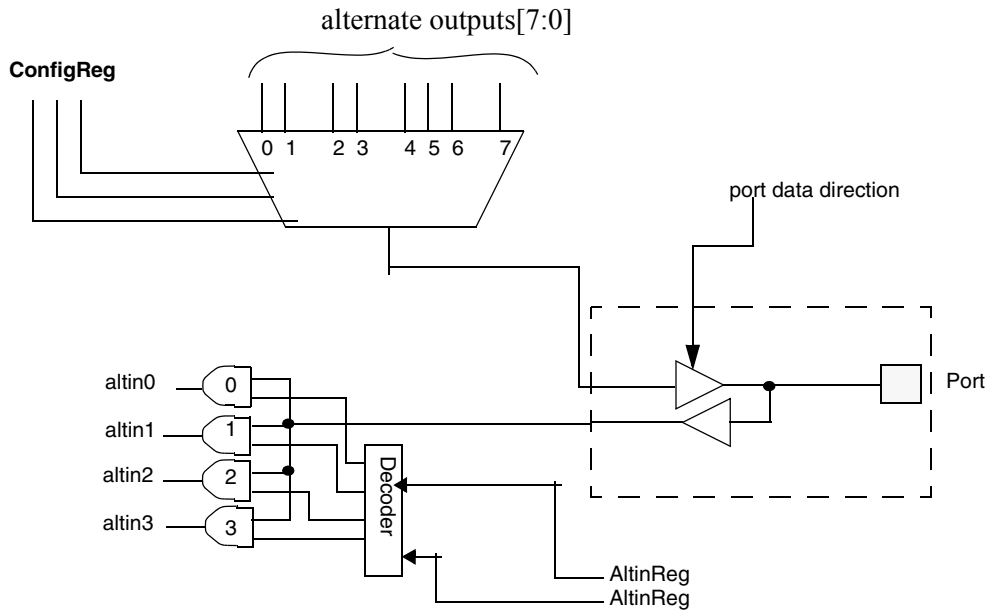


Figure 23-29.

*** PB[7, 8, 9,11]**

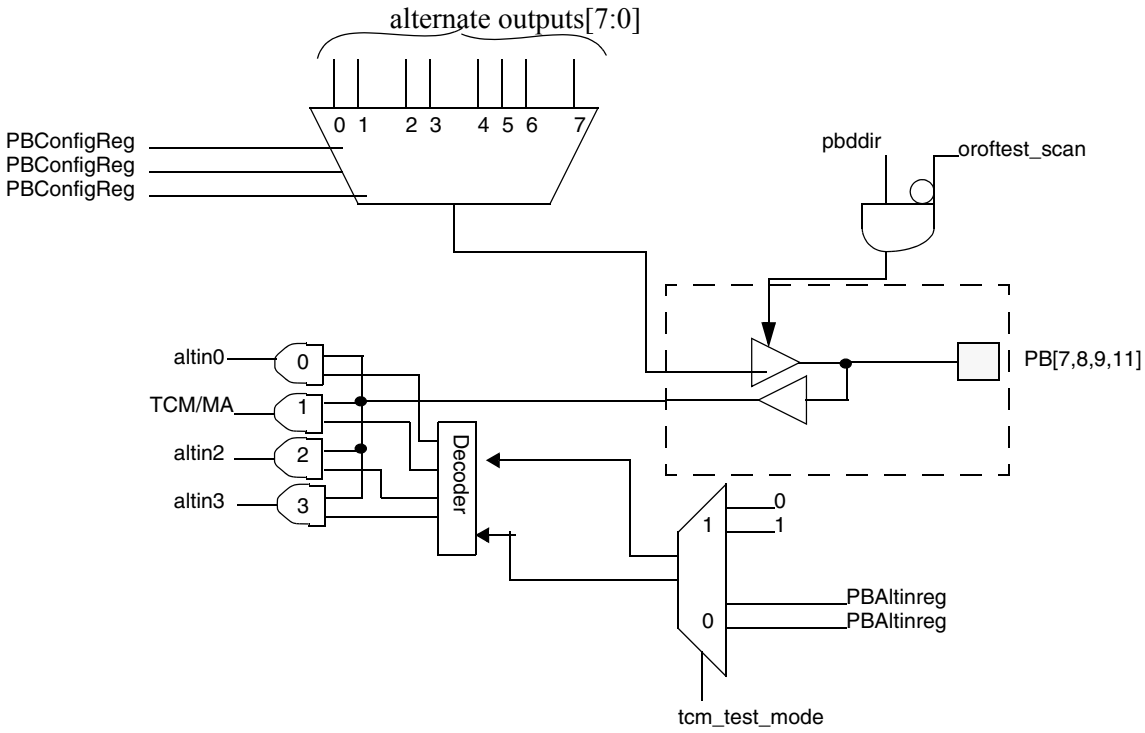


Figure 23-30. * gpio_portb_altin[2] configured for scan_mode

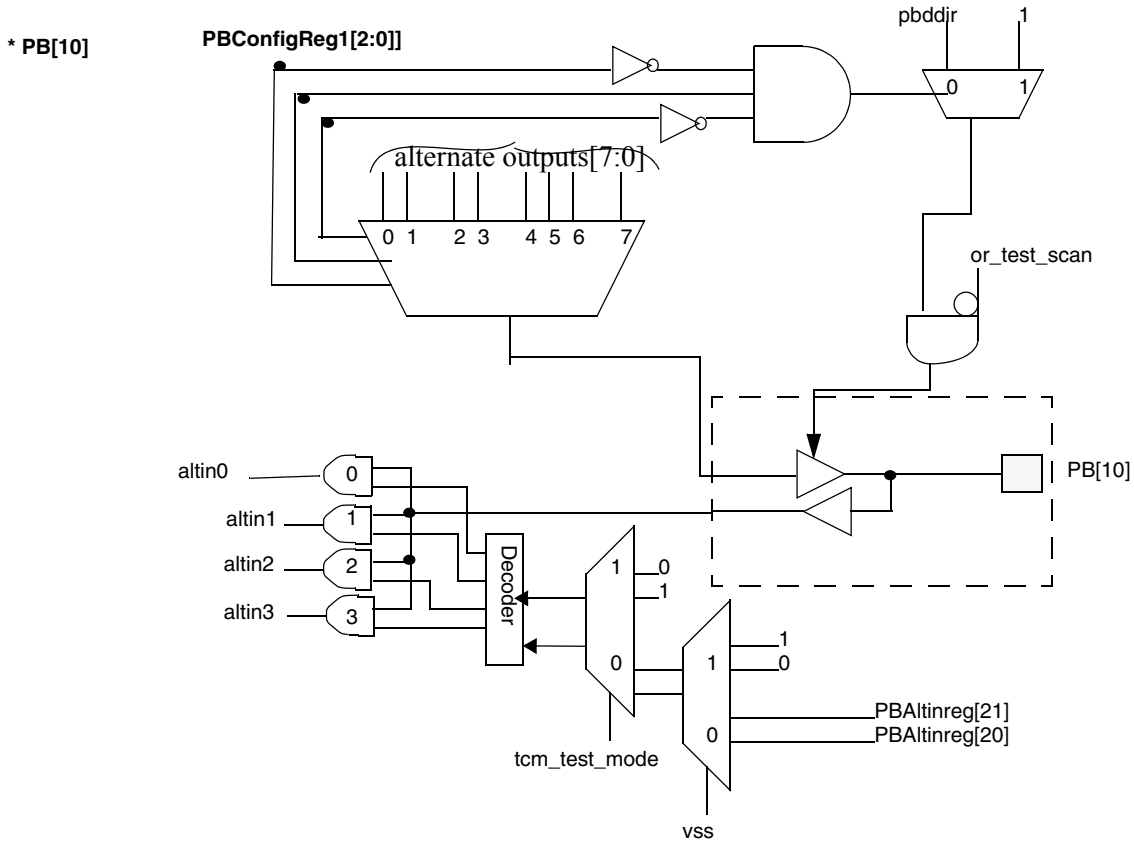


Figure 23-31. * gpio_portb_altin[2] configured for scan_mode

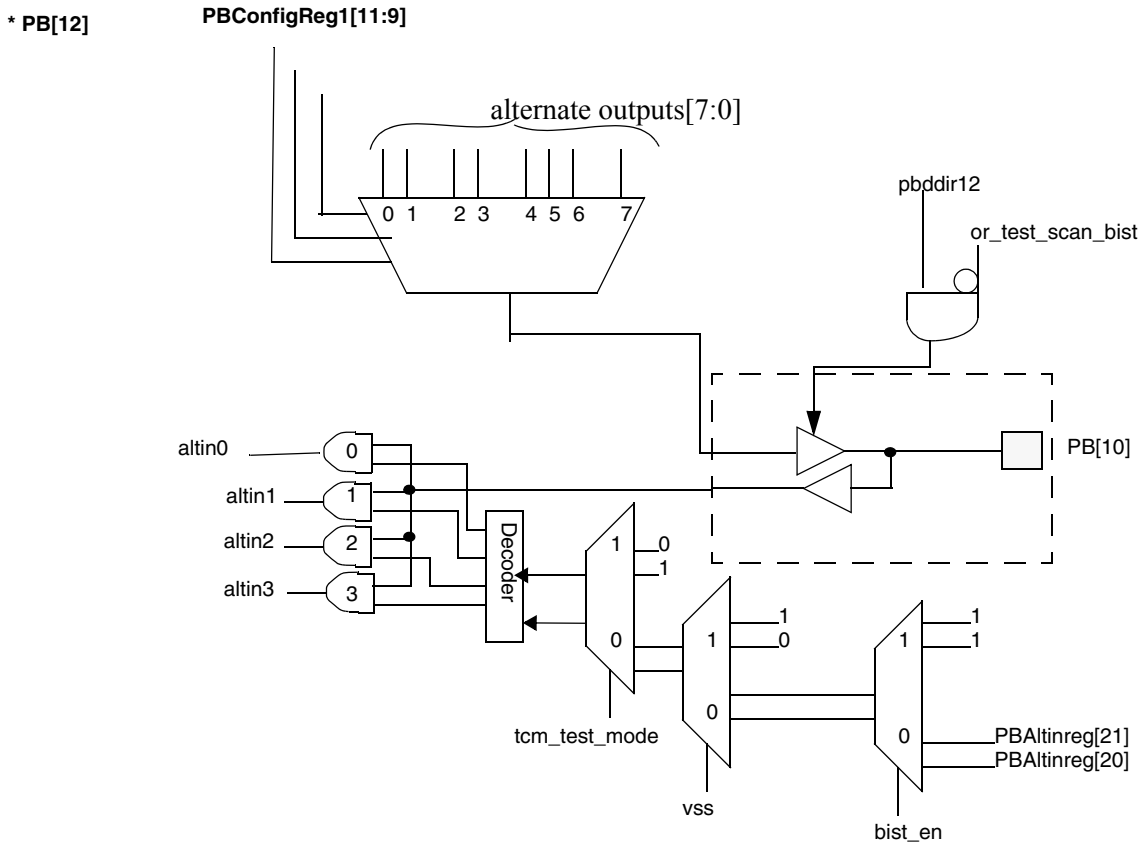


Figure 23-32. * gpio_portb_altin[2] configured for scan_mode

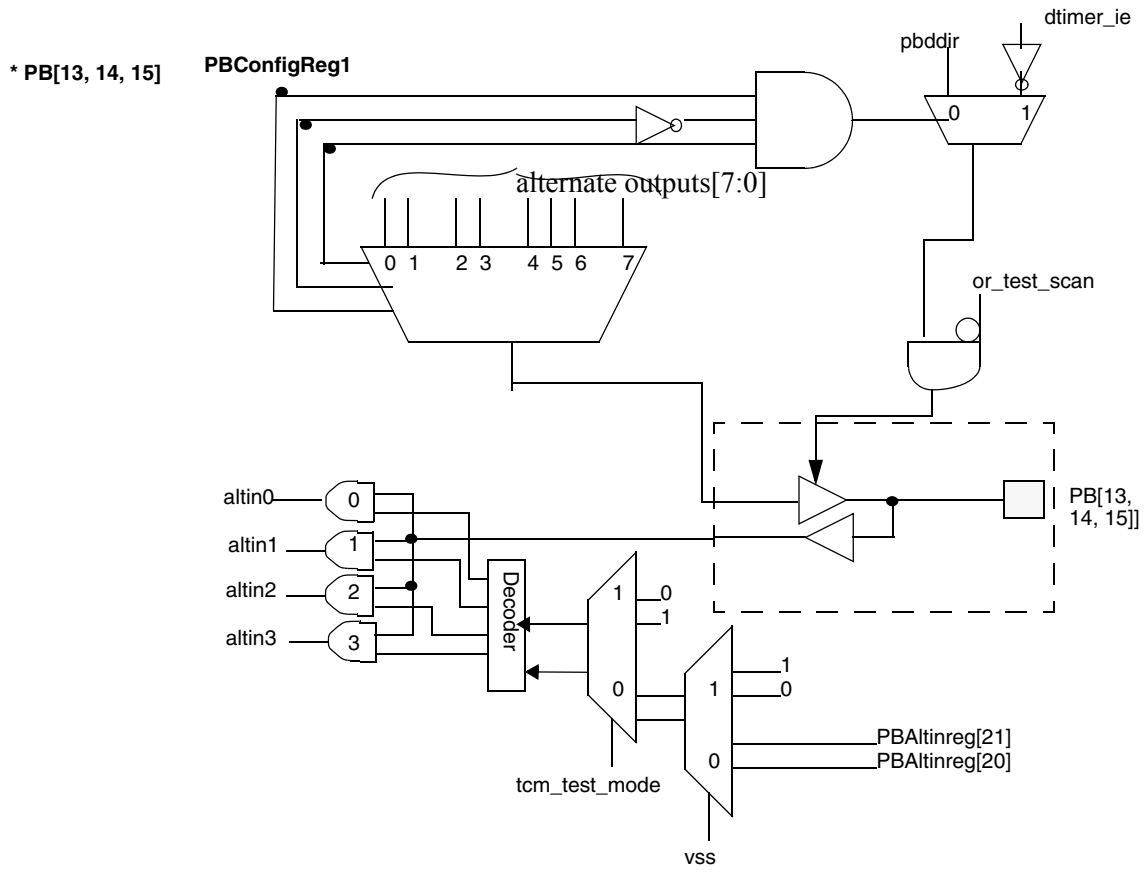


Figure 23-33. * gpio_portb_altin[2] configured for scan_mode

PC [0,15]

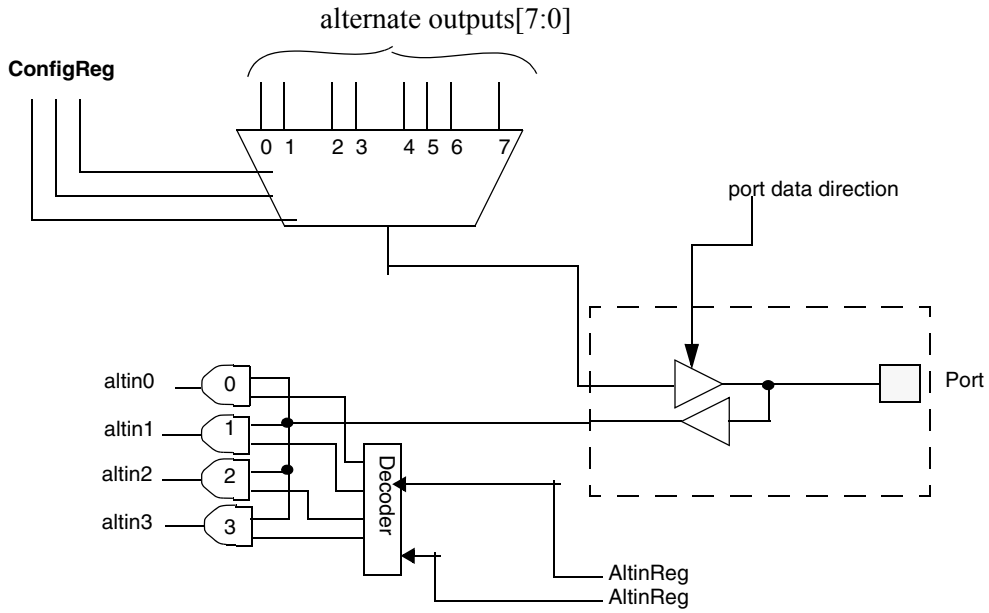


Figure 23-34.

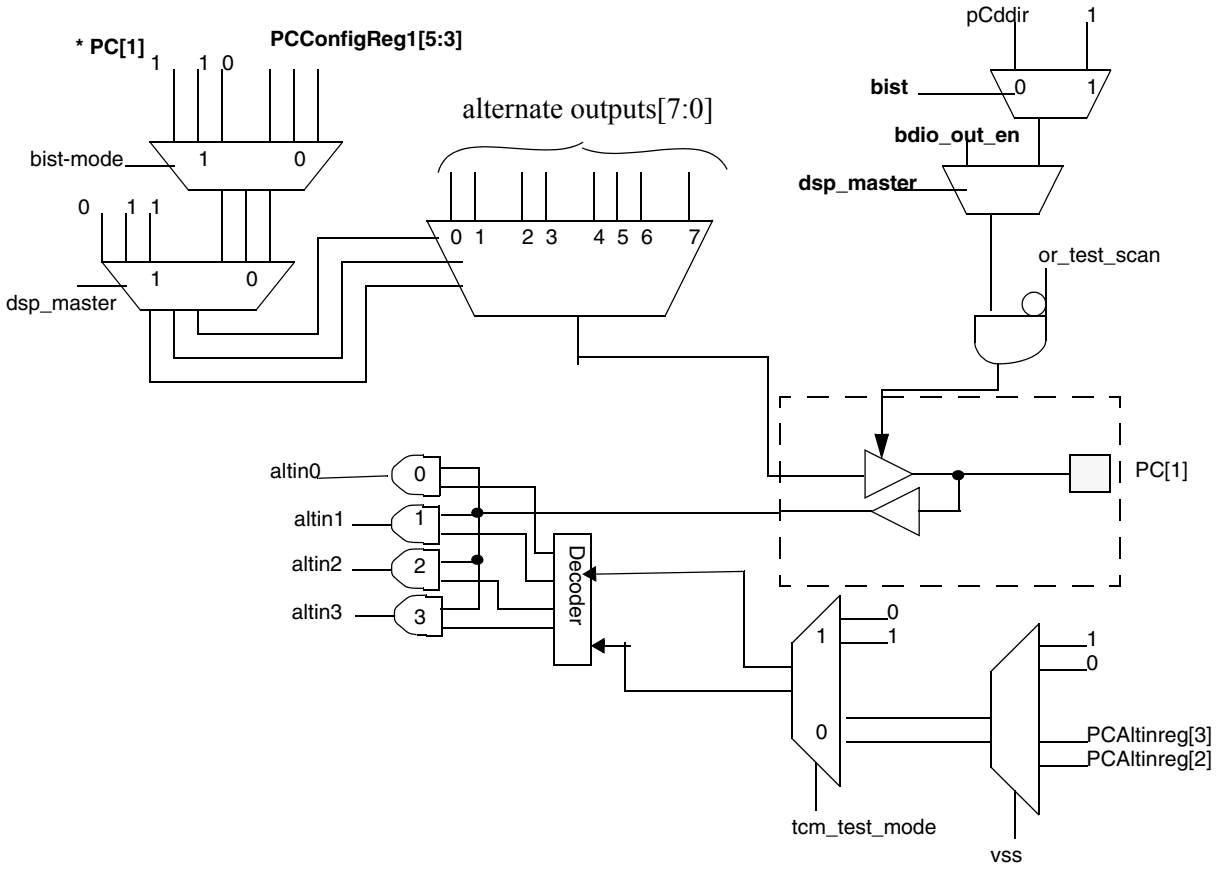


Figure 23-35. * gpio_portc_altin[2] configured for scan_mode

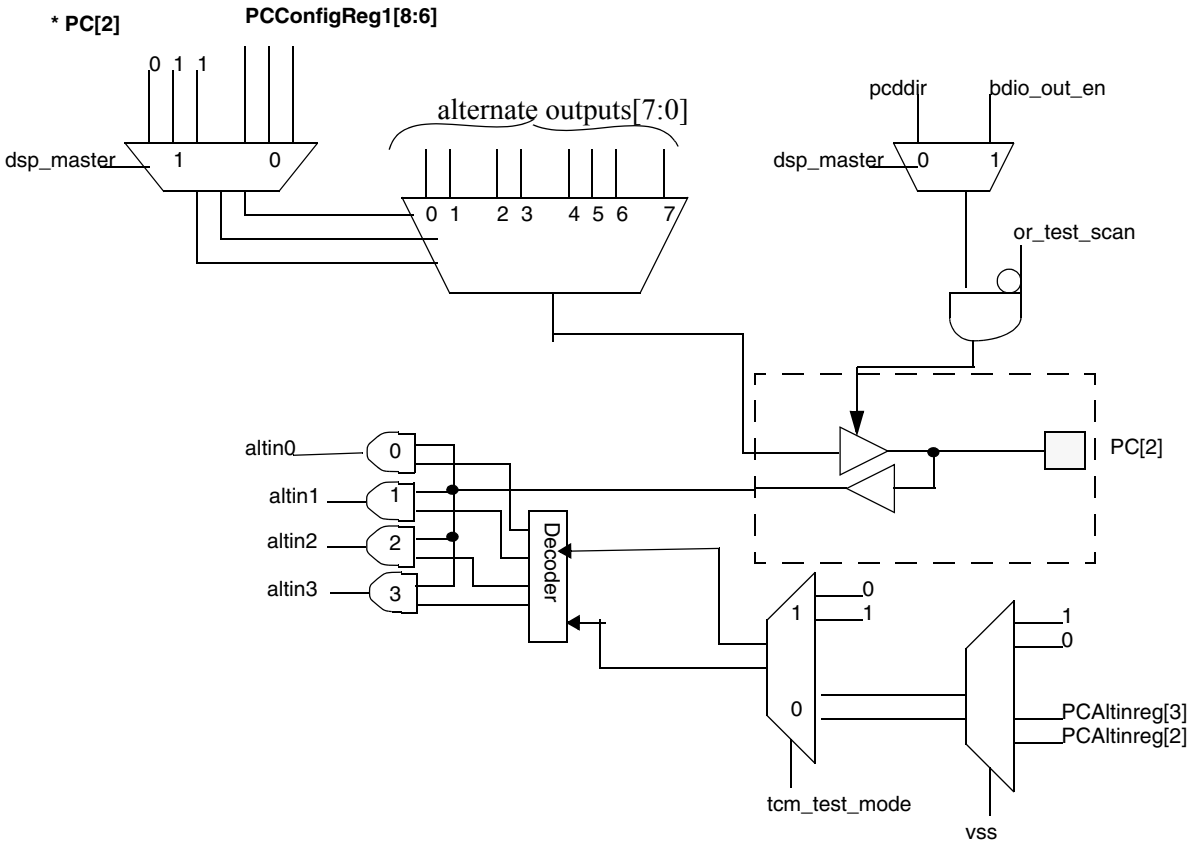


Figure 23-36. * gpio_portc_altin[2] configured for scan_mode

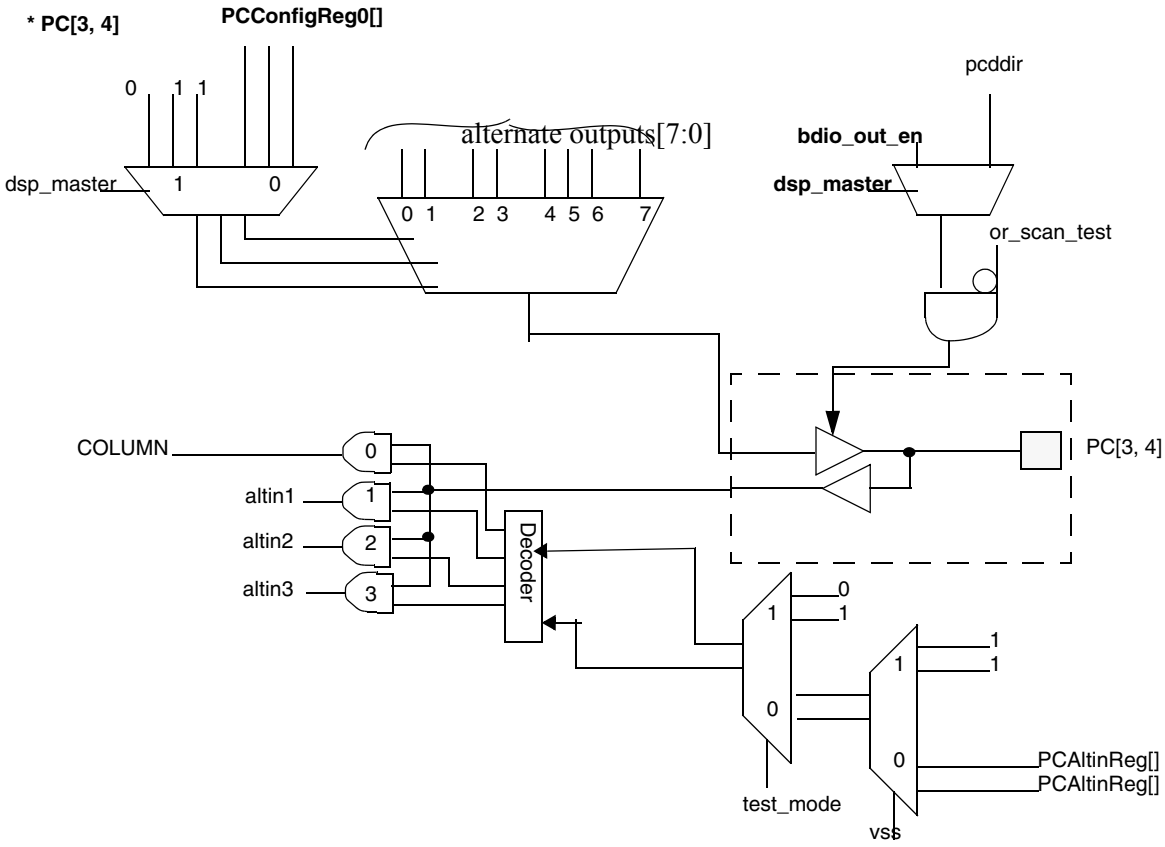


Figure 23-37. * gpio_portc_altin[2] configured for scan_mode

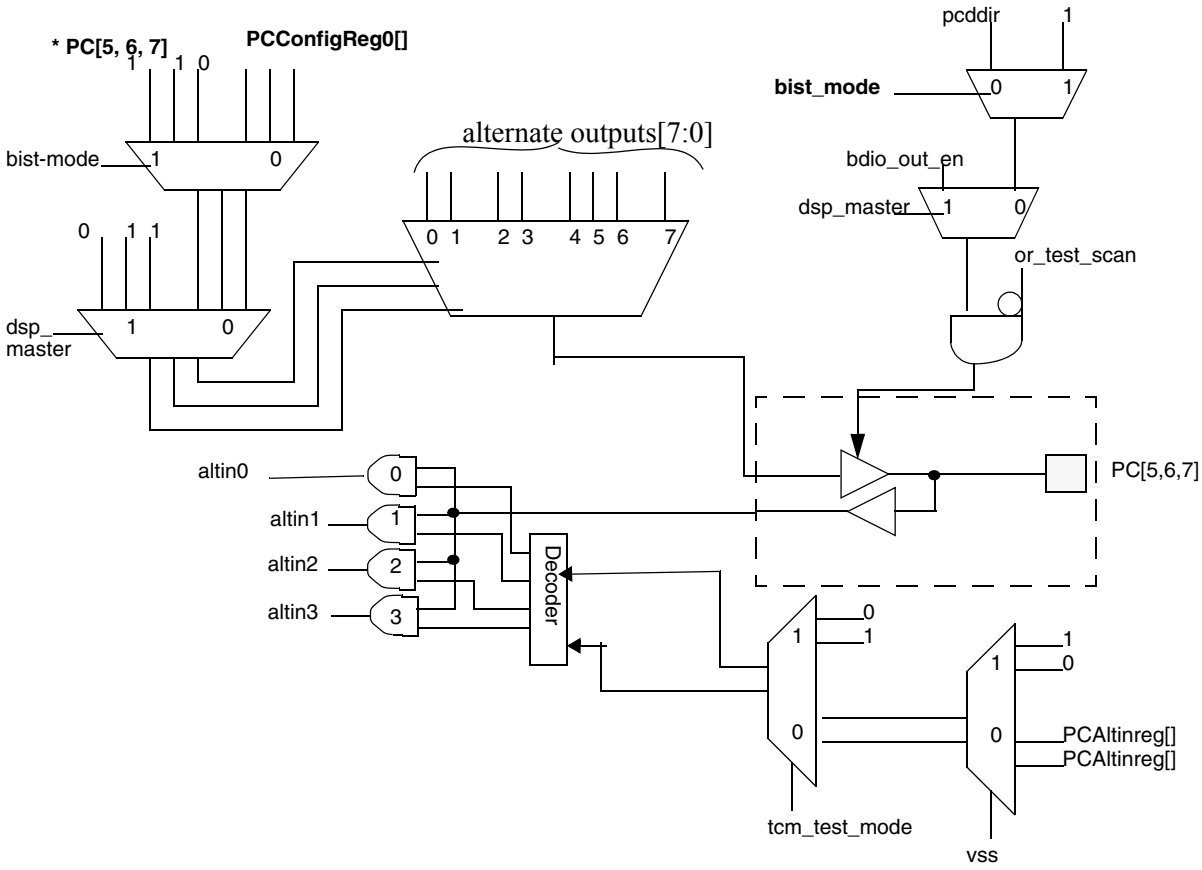


Figure 23-38. * gpio_portc_altin[2] configured for scan_mode

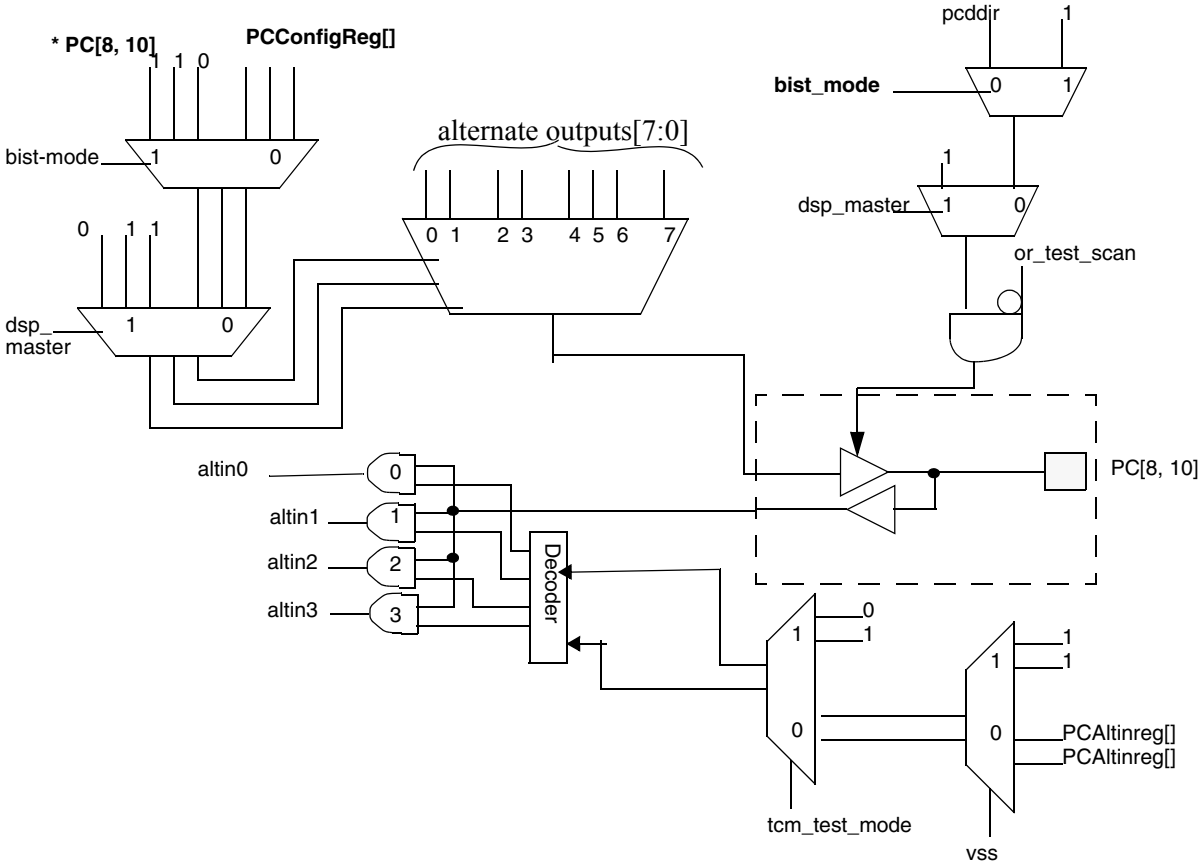


Figure 23-39. * gpio_portc_altin[3] configured for scan_mode

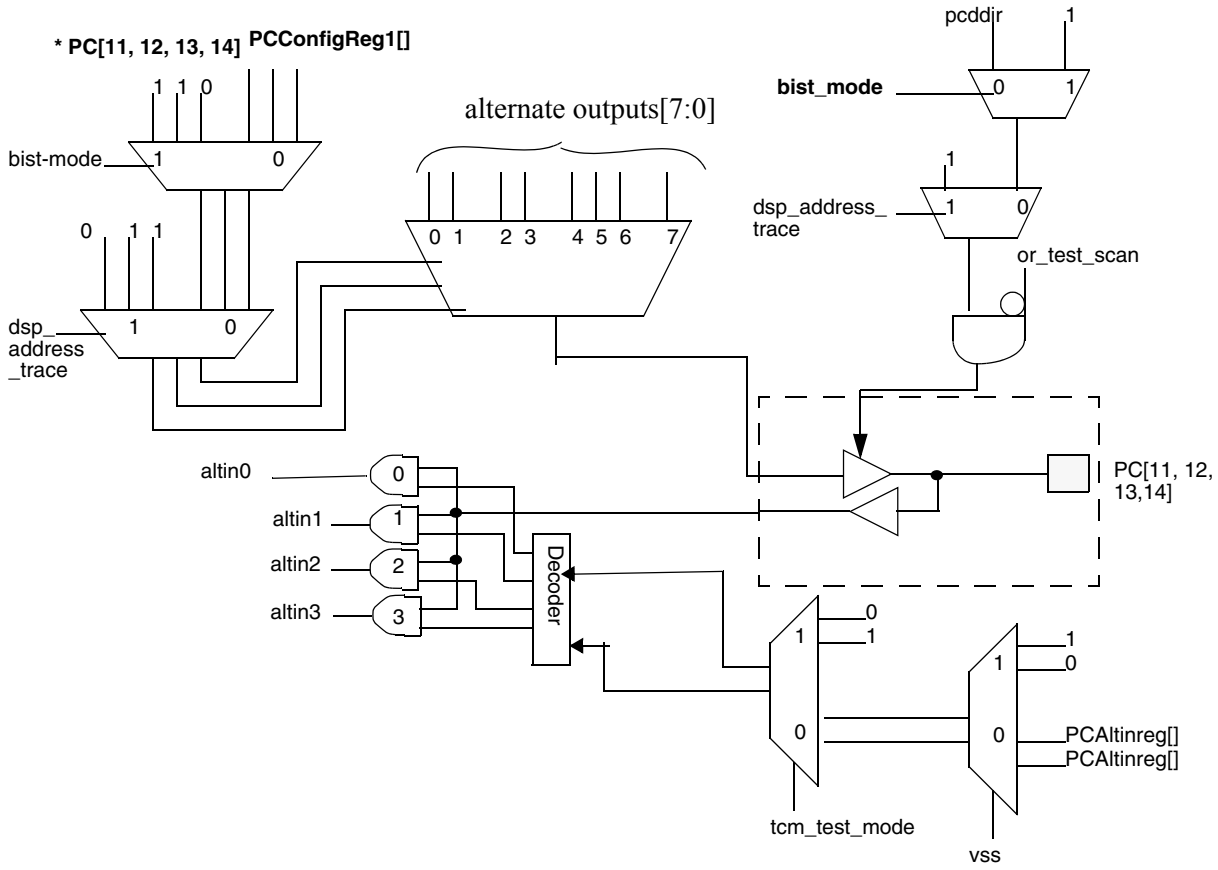


Figure 23-40. * gpio_portc_altin[2] configured for scan_mode

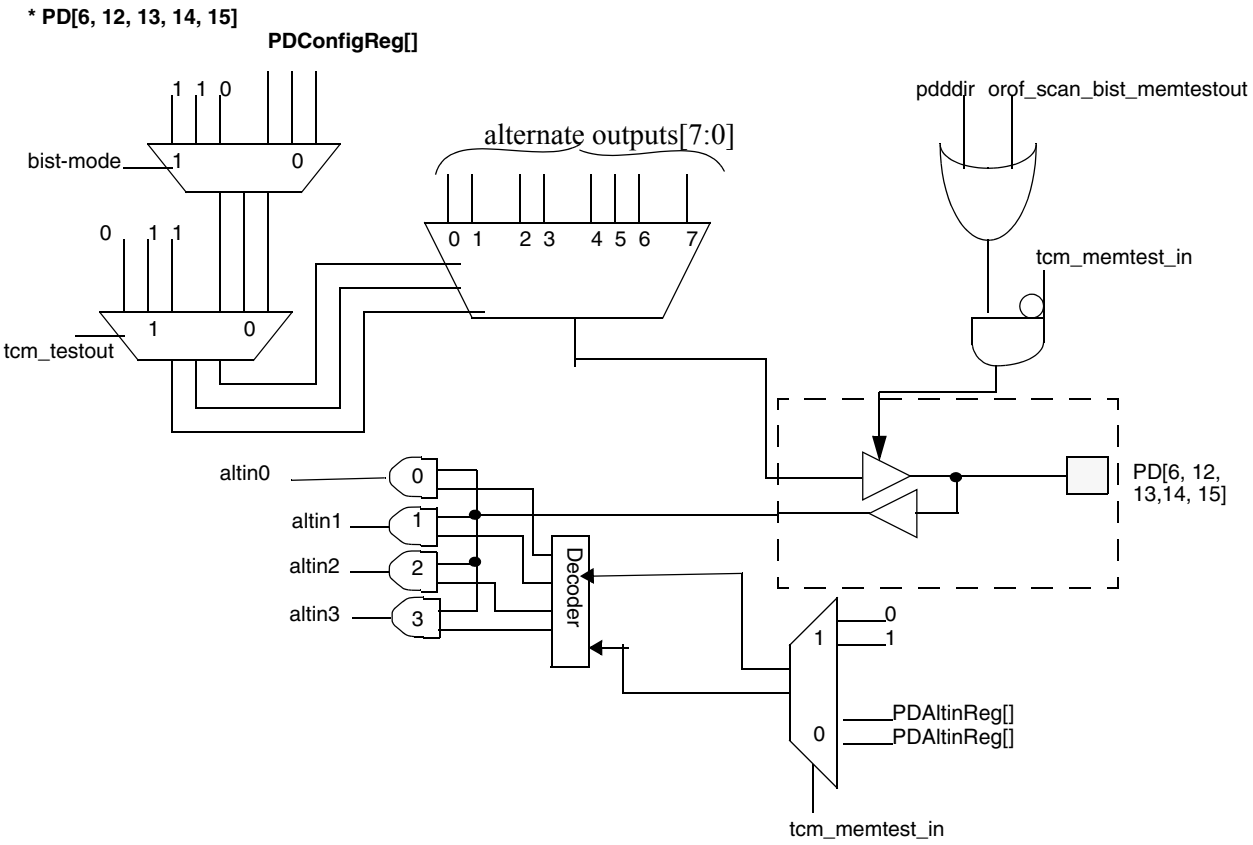


Figure 23-41. * gpio_portd_funcout[1] configured for scan_mode

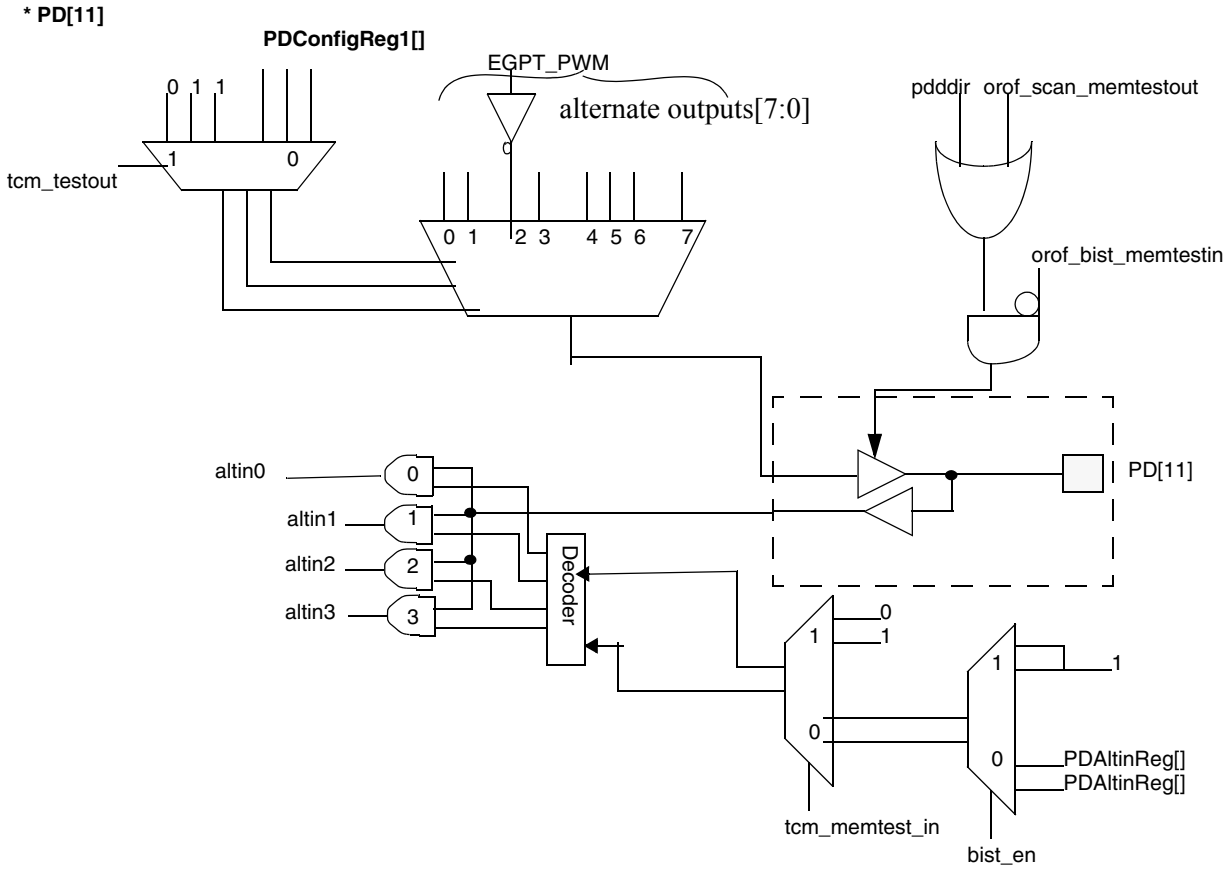


Figure 23-42. * gpio_portd_funcout[1] configured for scan_mode

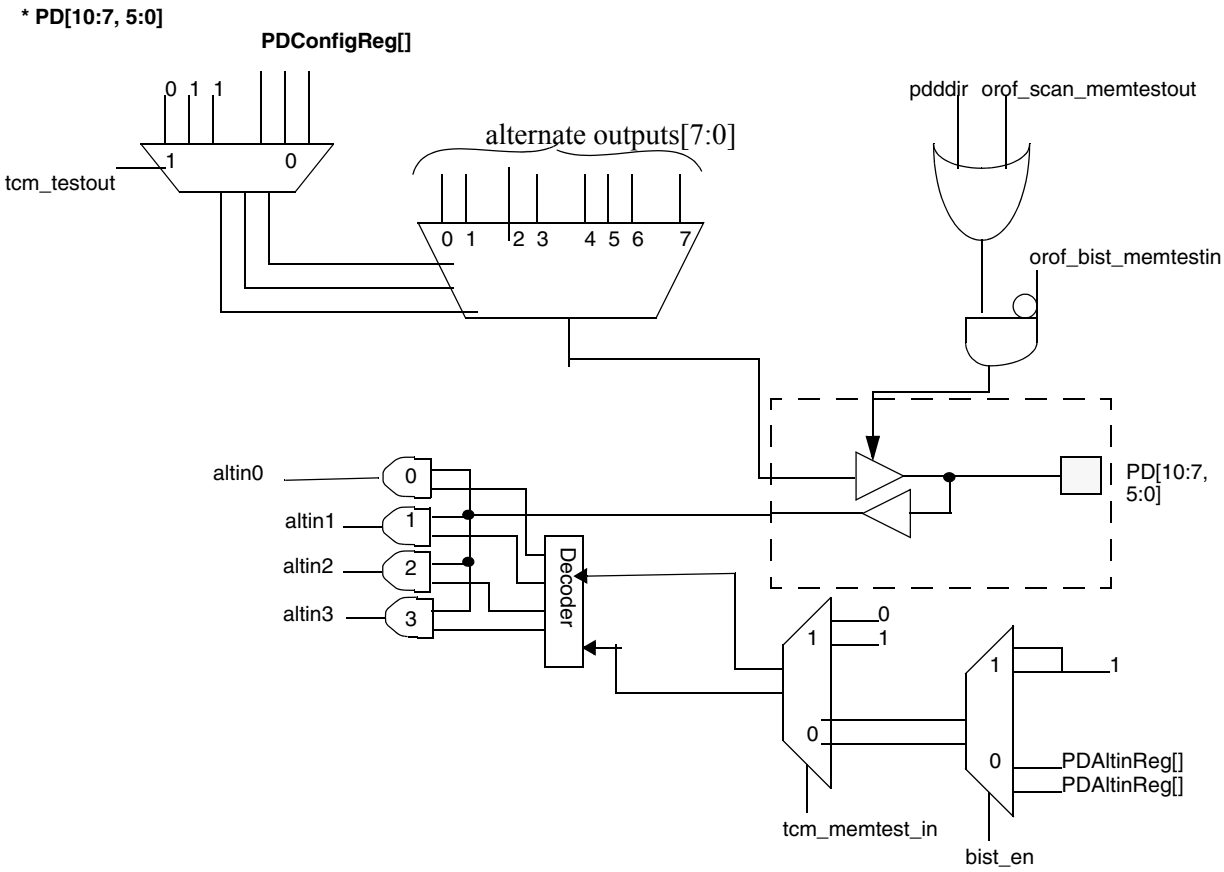


Figure 23-43. * gpio_portd_funcout[1] configured for scan_mode

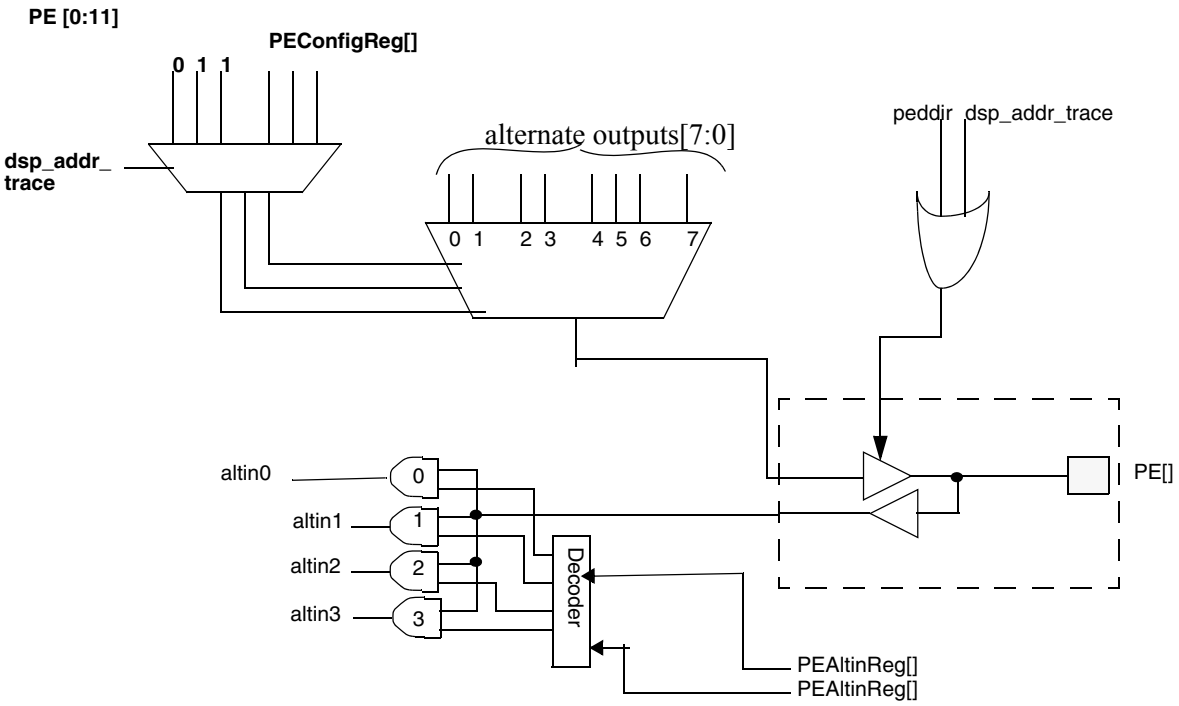

PE [0:11]
PEConfigReg[]
0 1 1
dsp_addr_
trace
alternate outputs[7:0]
0 1 2 3 4 5 6 7
peddir
dsp_addr_trace
altin0
altin1
altin2
altin3
0
1
2
3
Decoder
PE[]
PEAltinReg[]
PEAltinReg[]

Figure 23-44.

PE [15:12]

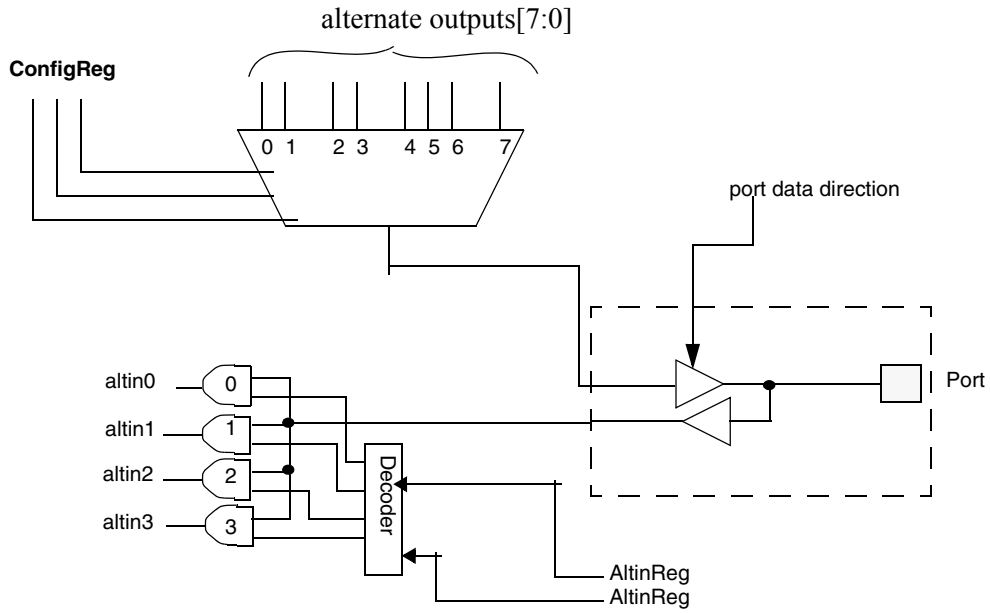


Figure 23-45.

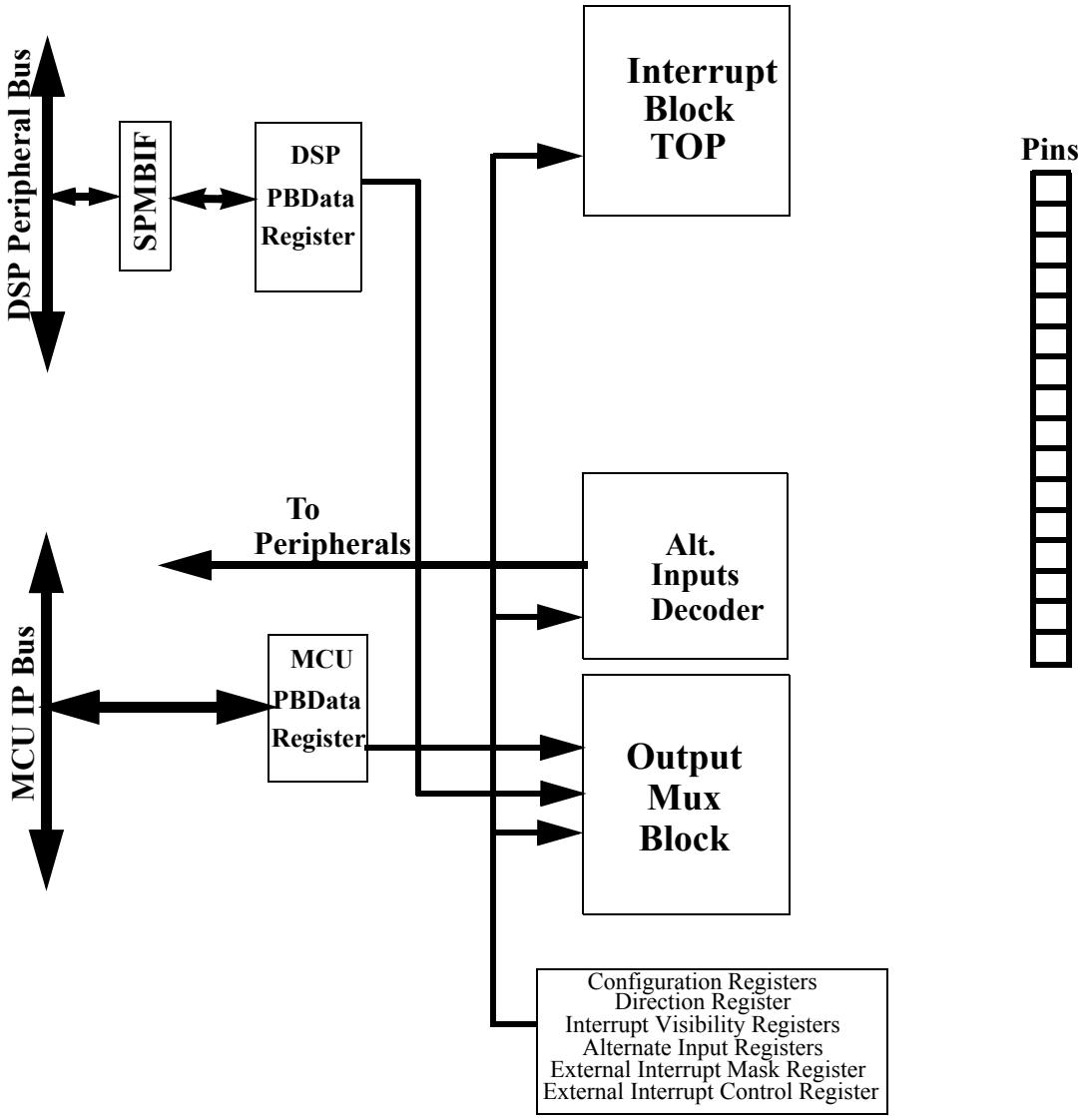


Figure 23-46. Port B Top Level Block Diagram

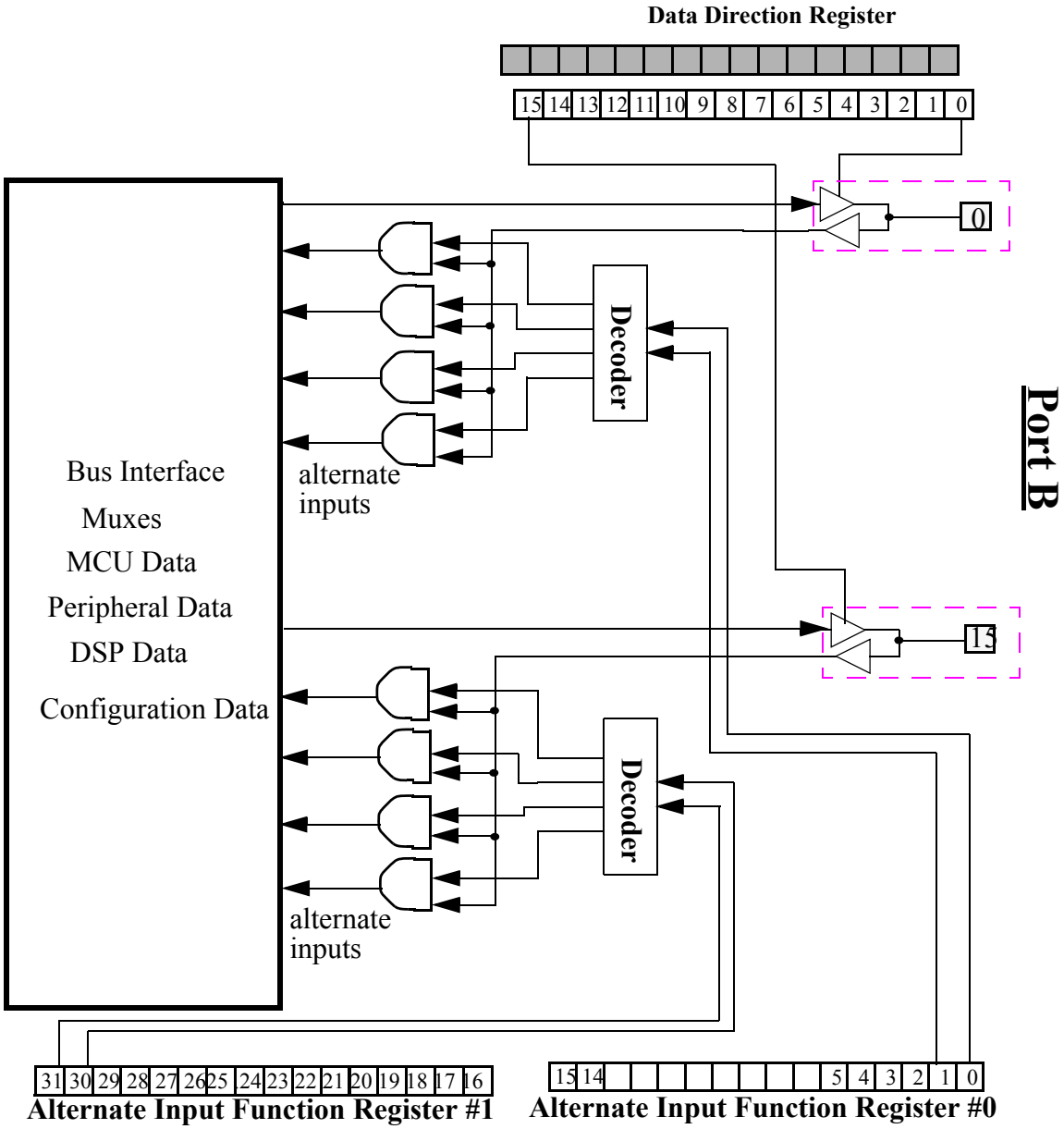


Figure 23-47. Alternate Input Functionality Block Diagram

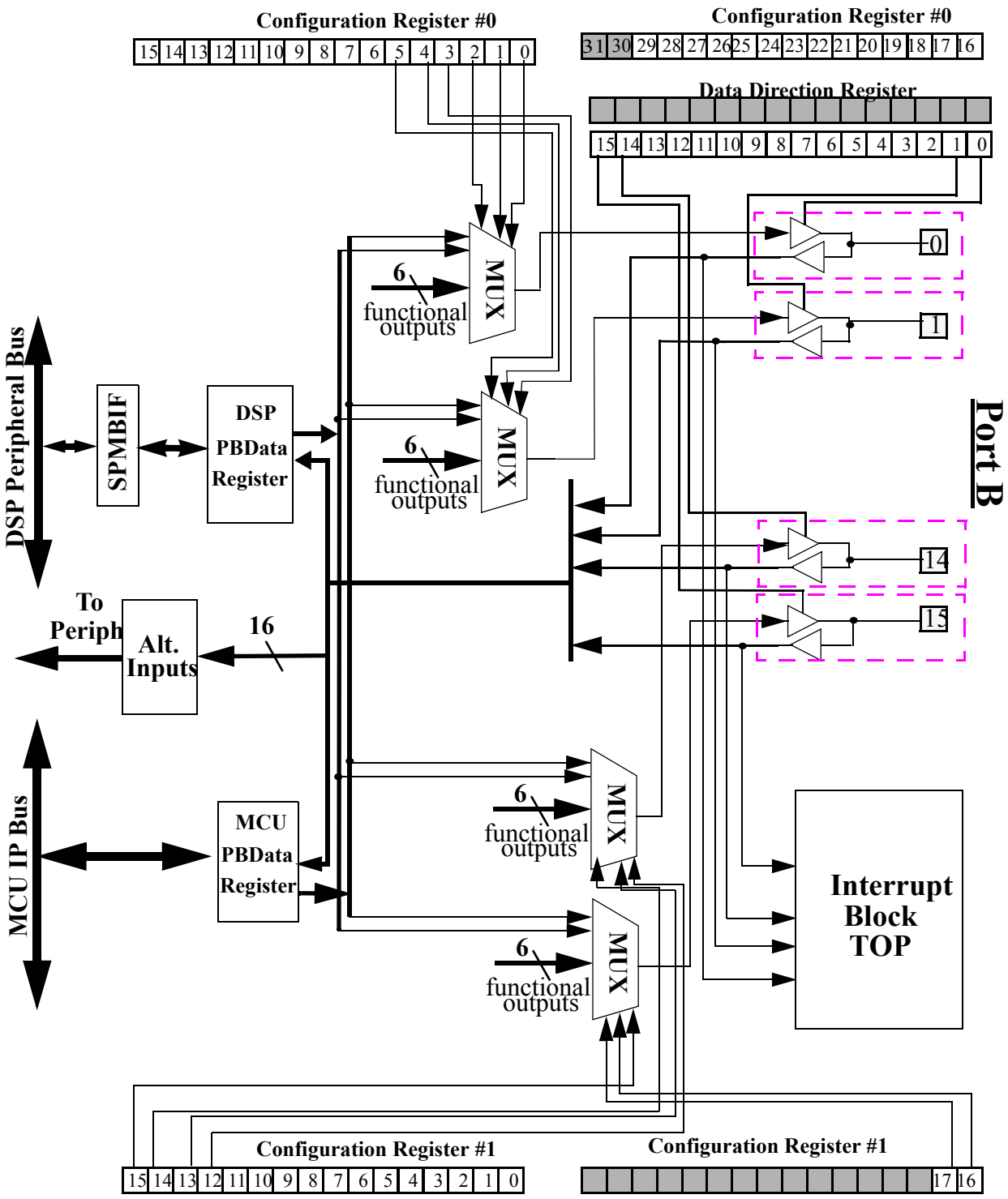


Figure 23-48. Port B Output Configuration Block Diagram

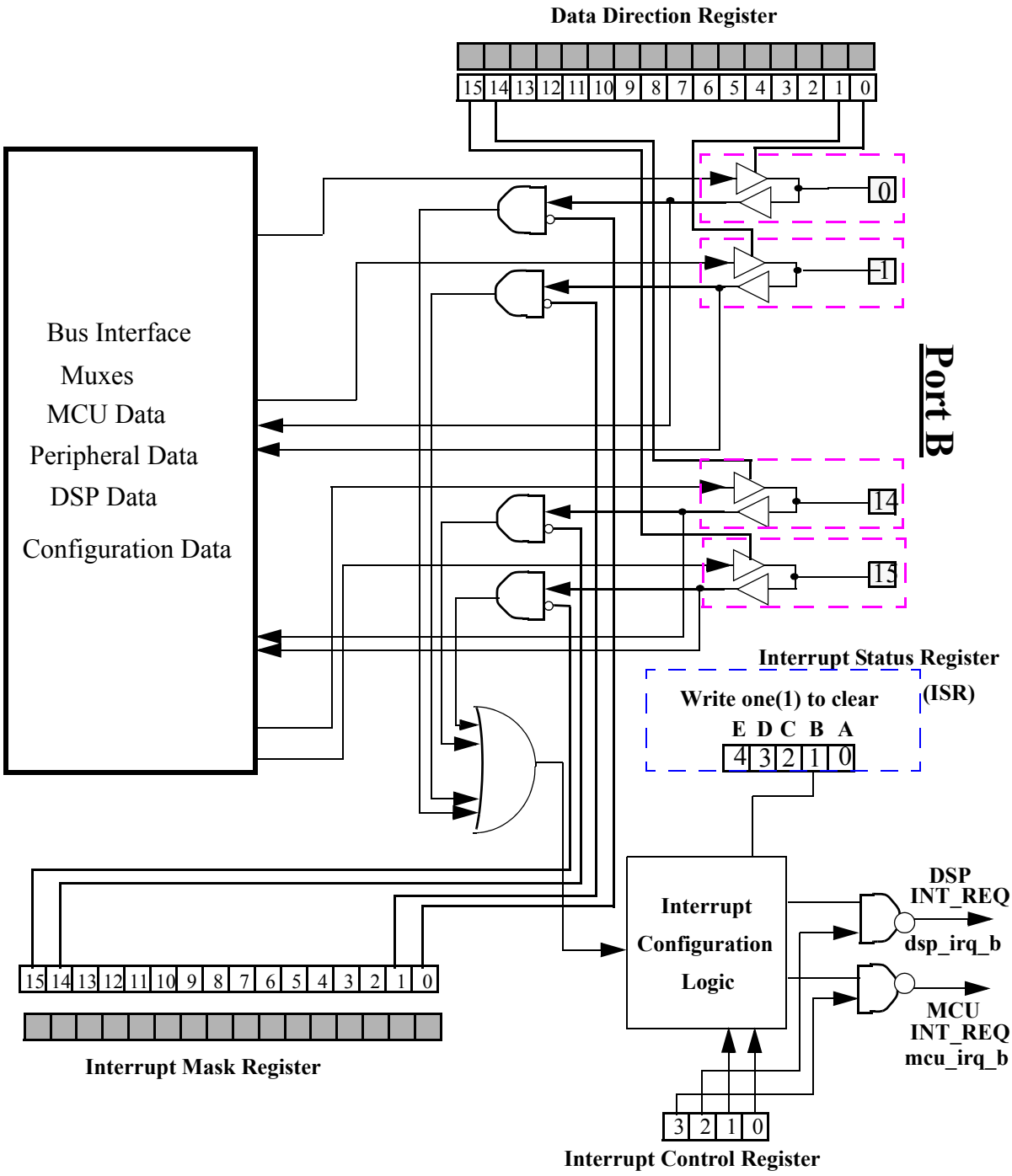


Figure 23-49. External Interrupt Control Block Diagram Port B

23.14.2 Port C Block Diagrams

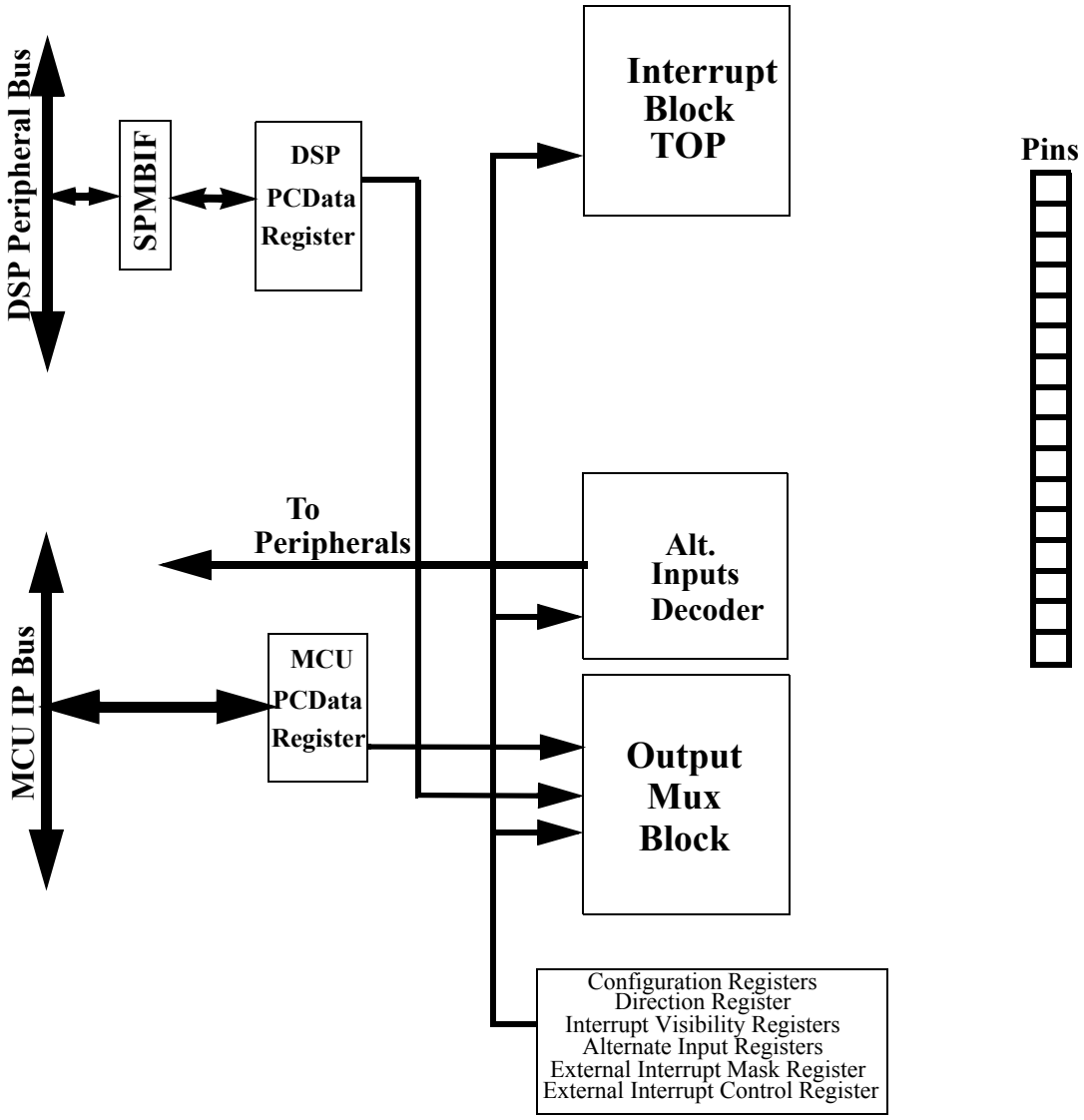


Figure 23-50. Port C Top Level Block Diagram

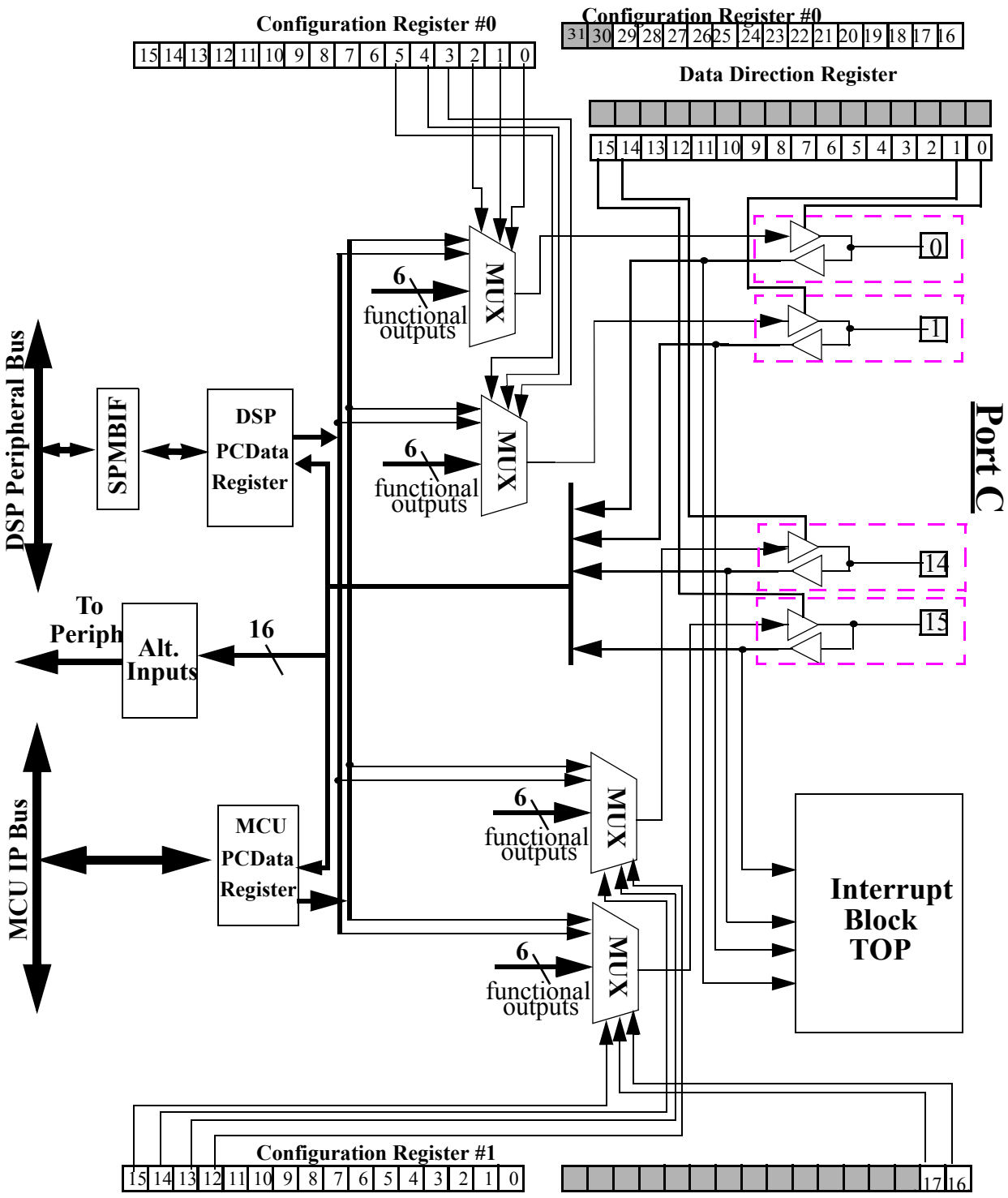


Figure 23-51. Port C Output Configuration Block Diagram

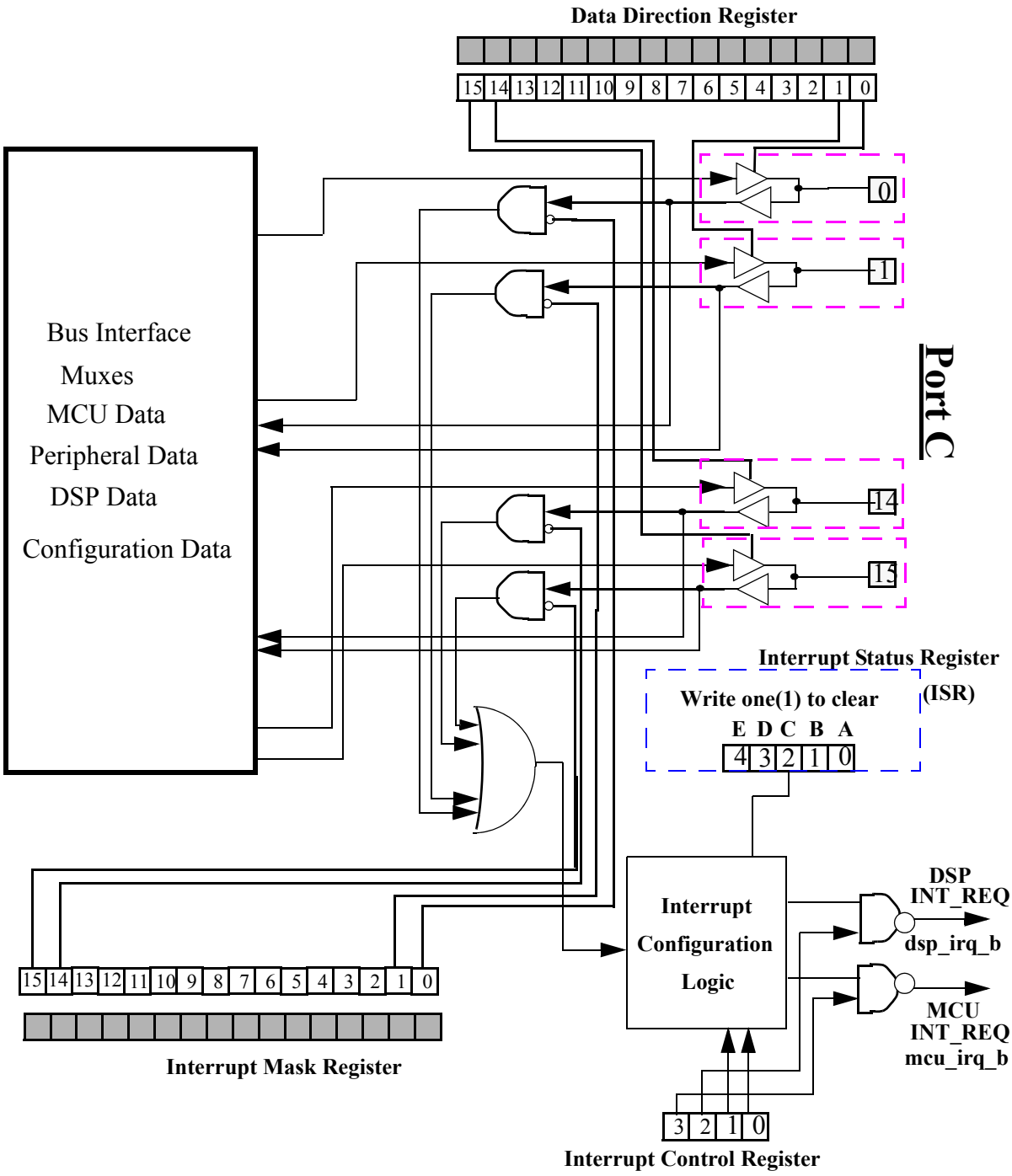


Figure 23-52. External Interrupt Control Block Diagram Port C

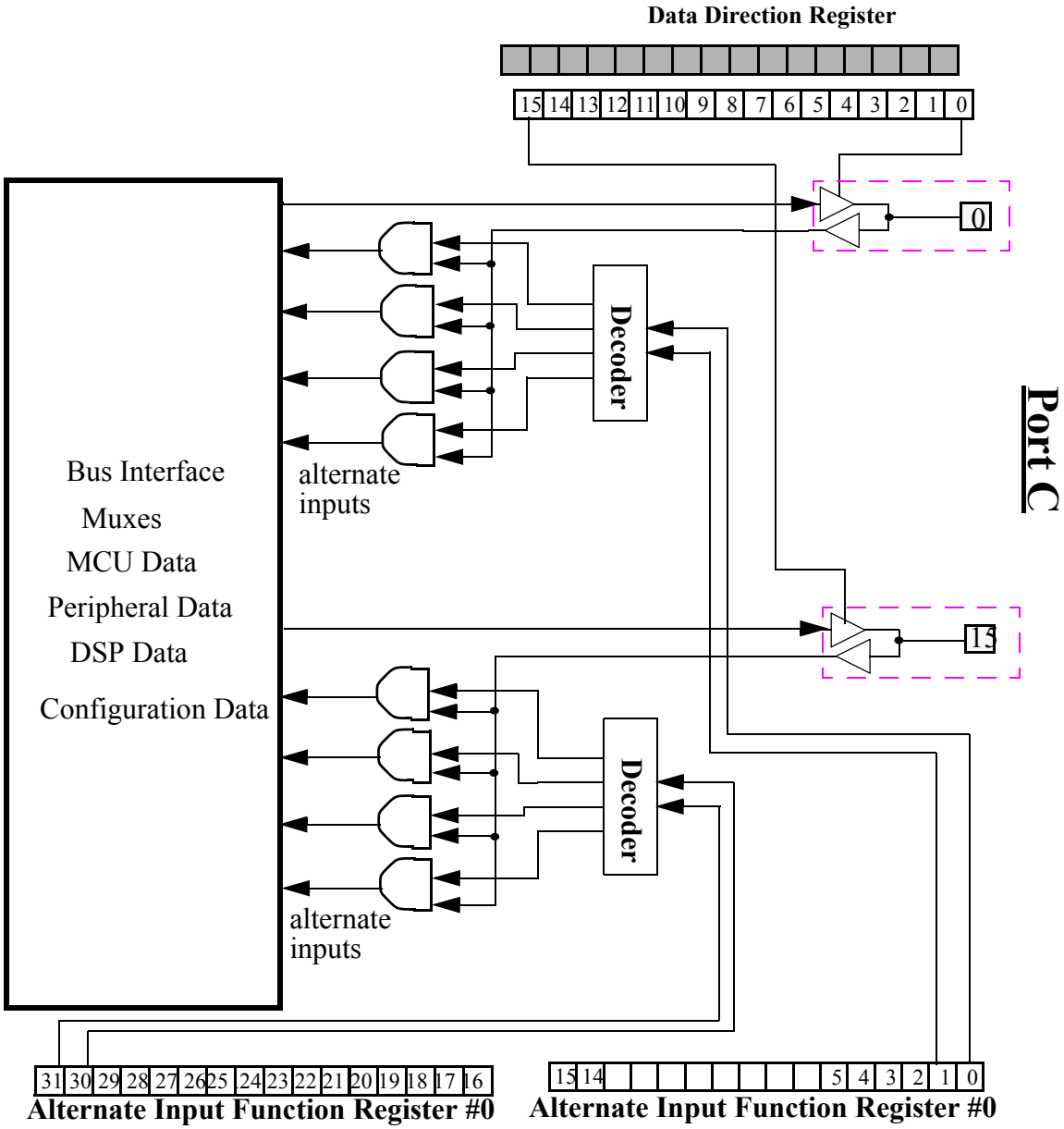


Figure 23-53. Alternate Input Functionality Block Diagram Port C

23.14.3 Port D Block Diagrams

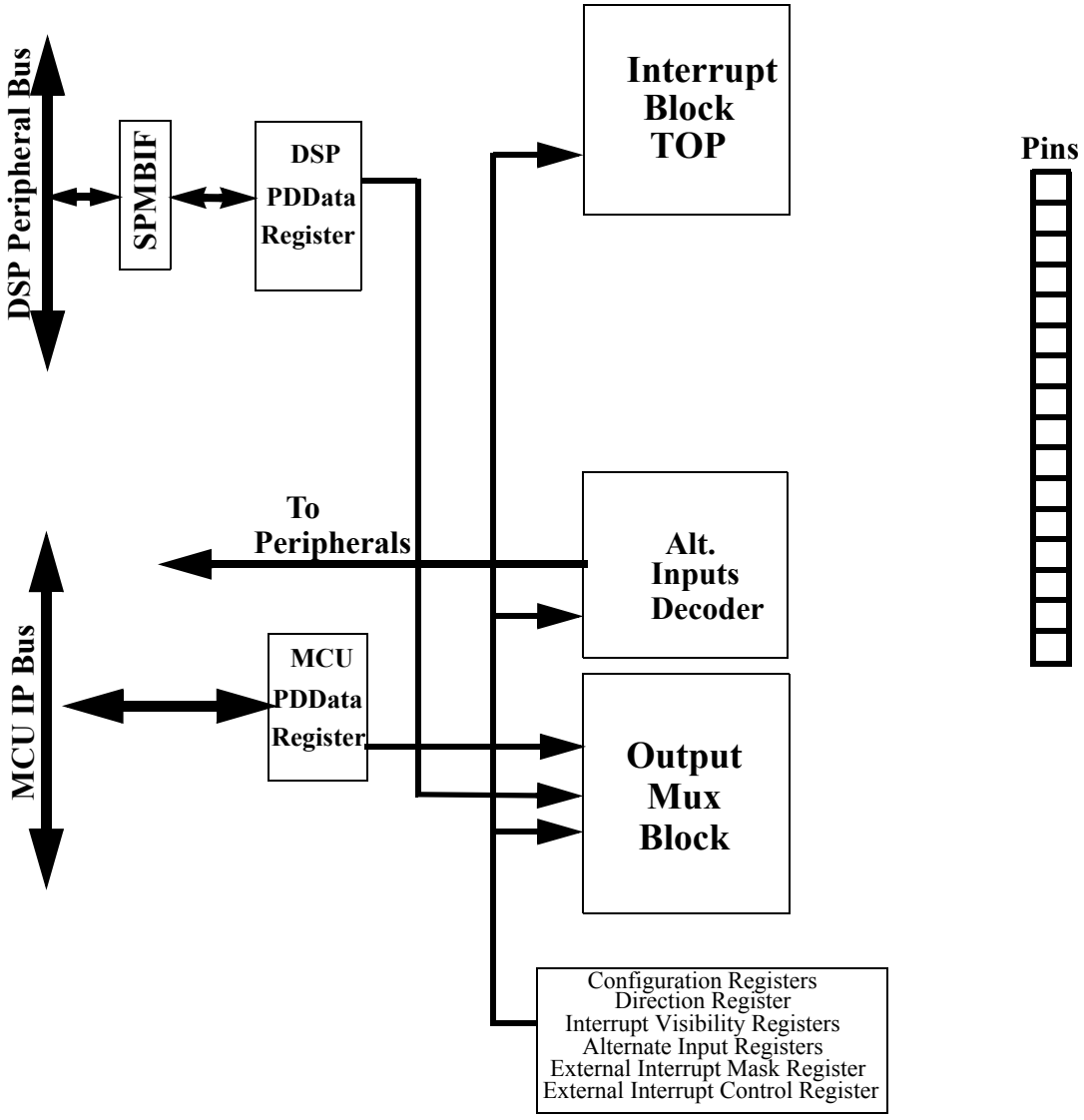


Figure 23-54. Port D Top Level Block Diagram

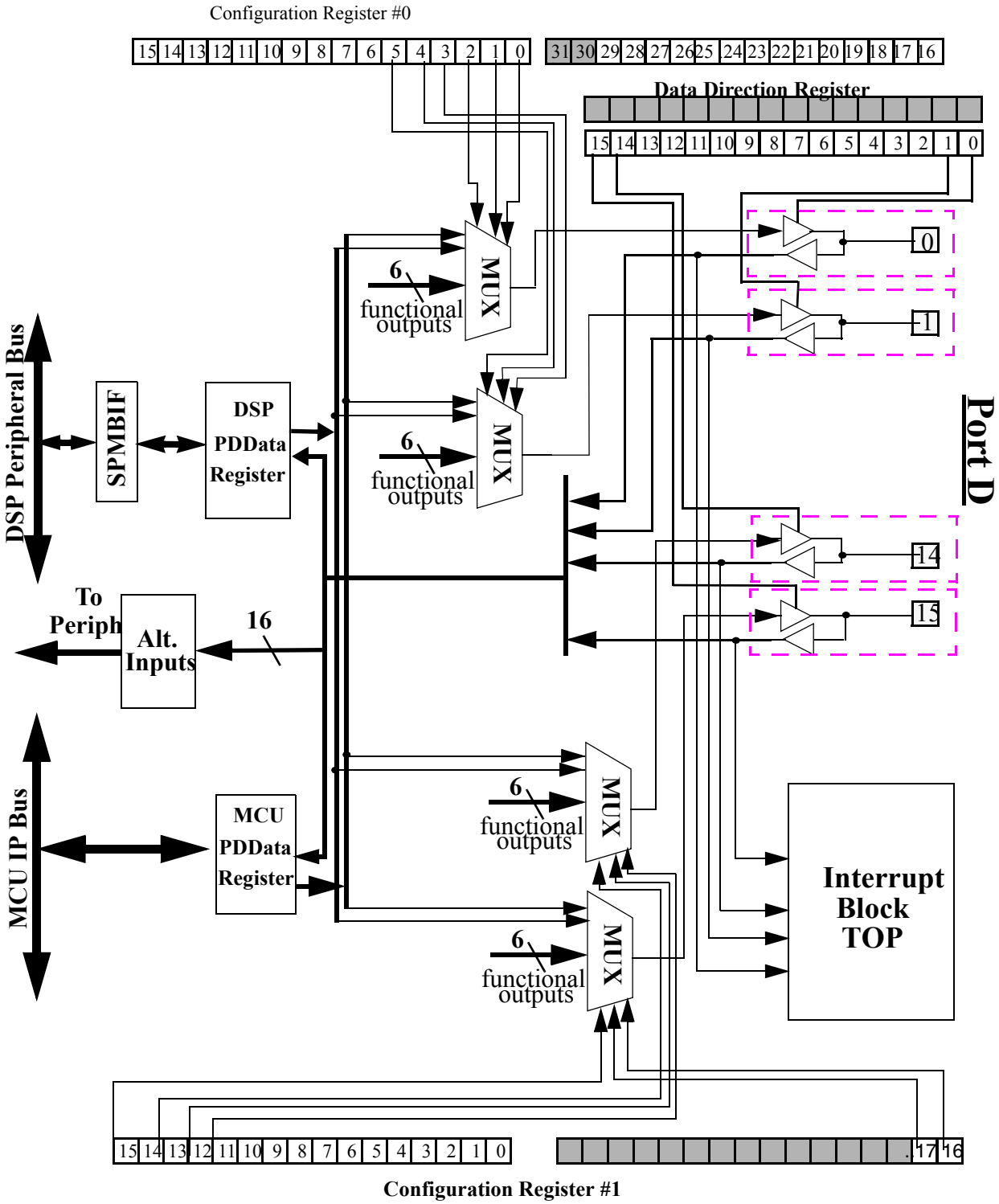


Figure 23-55. Port D Output Configuration Block Diagram

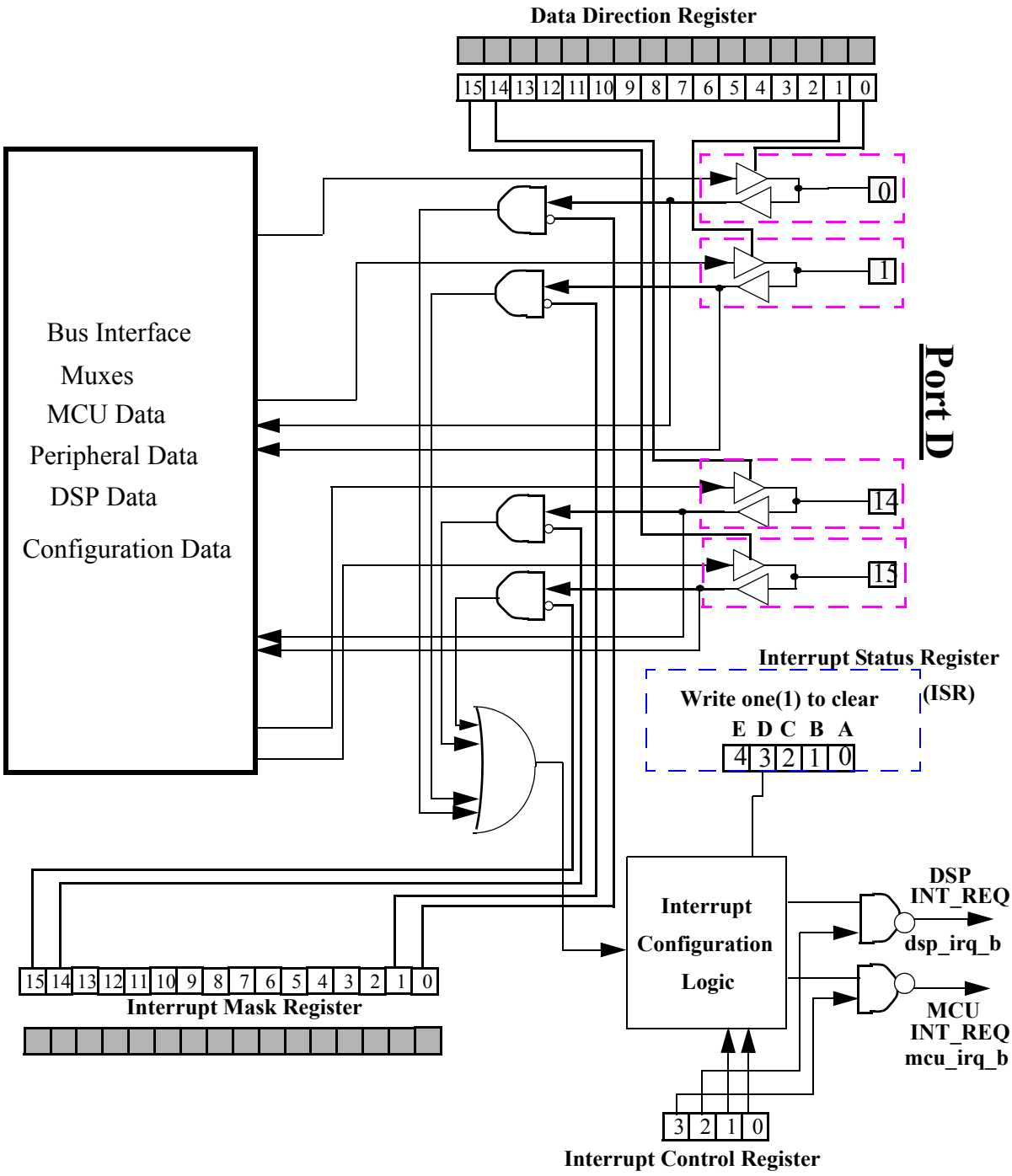


Figure 23-56. External Interrupt Control Block Diagram Port D

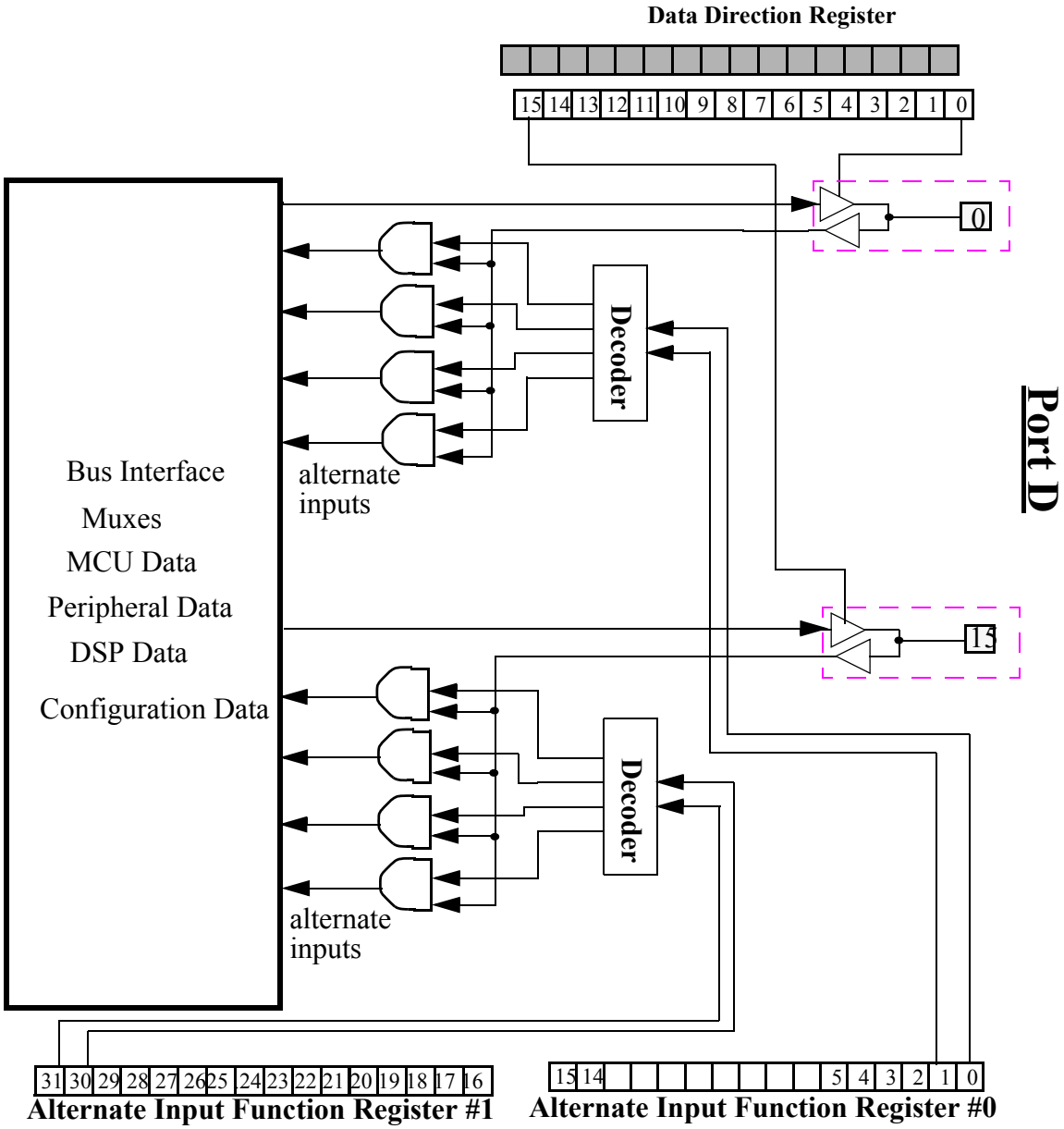


Figure 23-57. Alternate Input Functionality Block Diagram Port D

23.14.4 Port E Block Diagrams

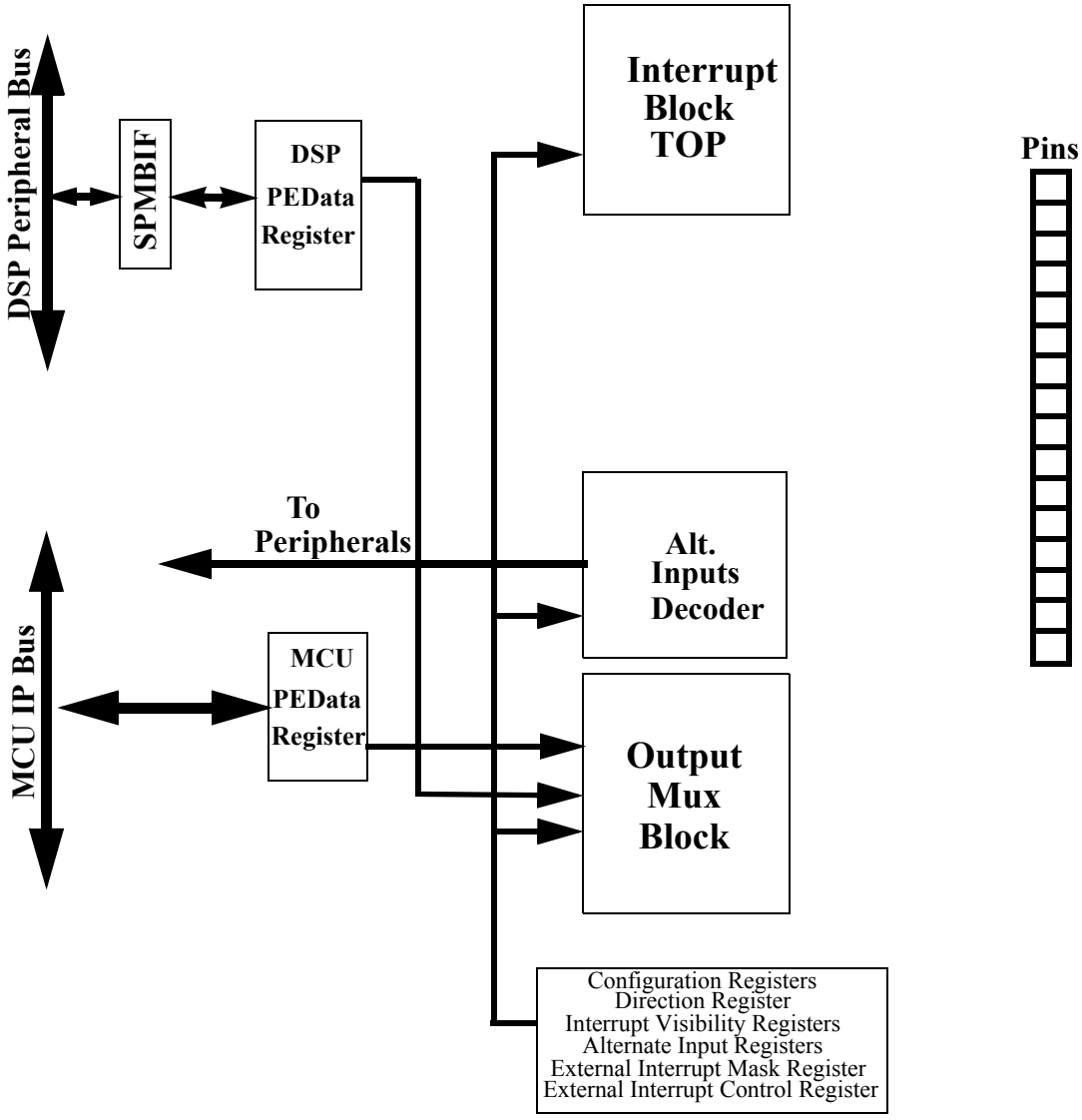


Figure 23-58. Port E Top Level Block Diagram

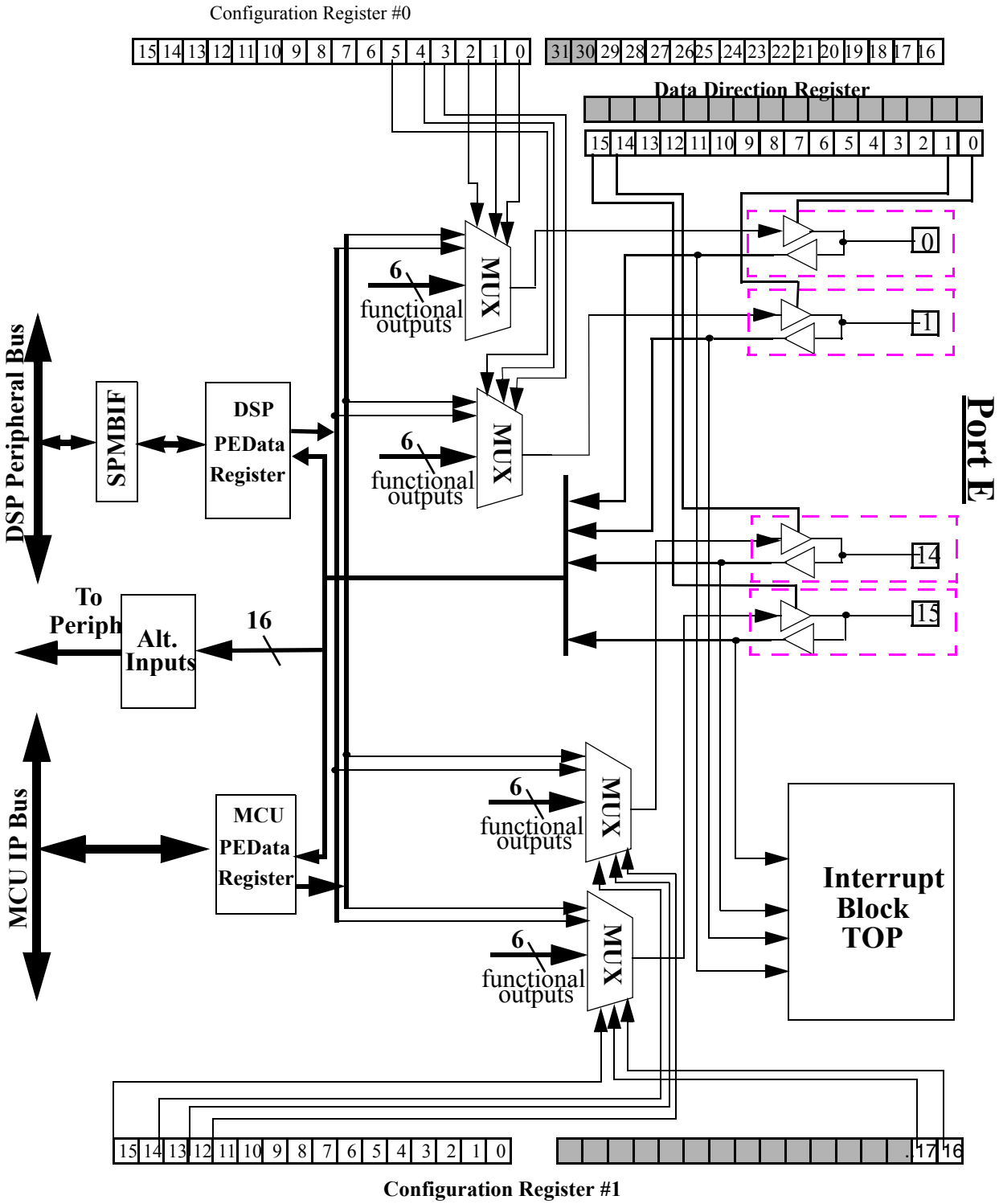


Figure 23-59. Port E Output Configuration Block Diagram

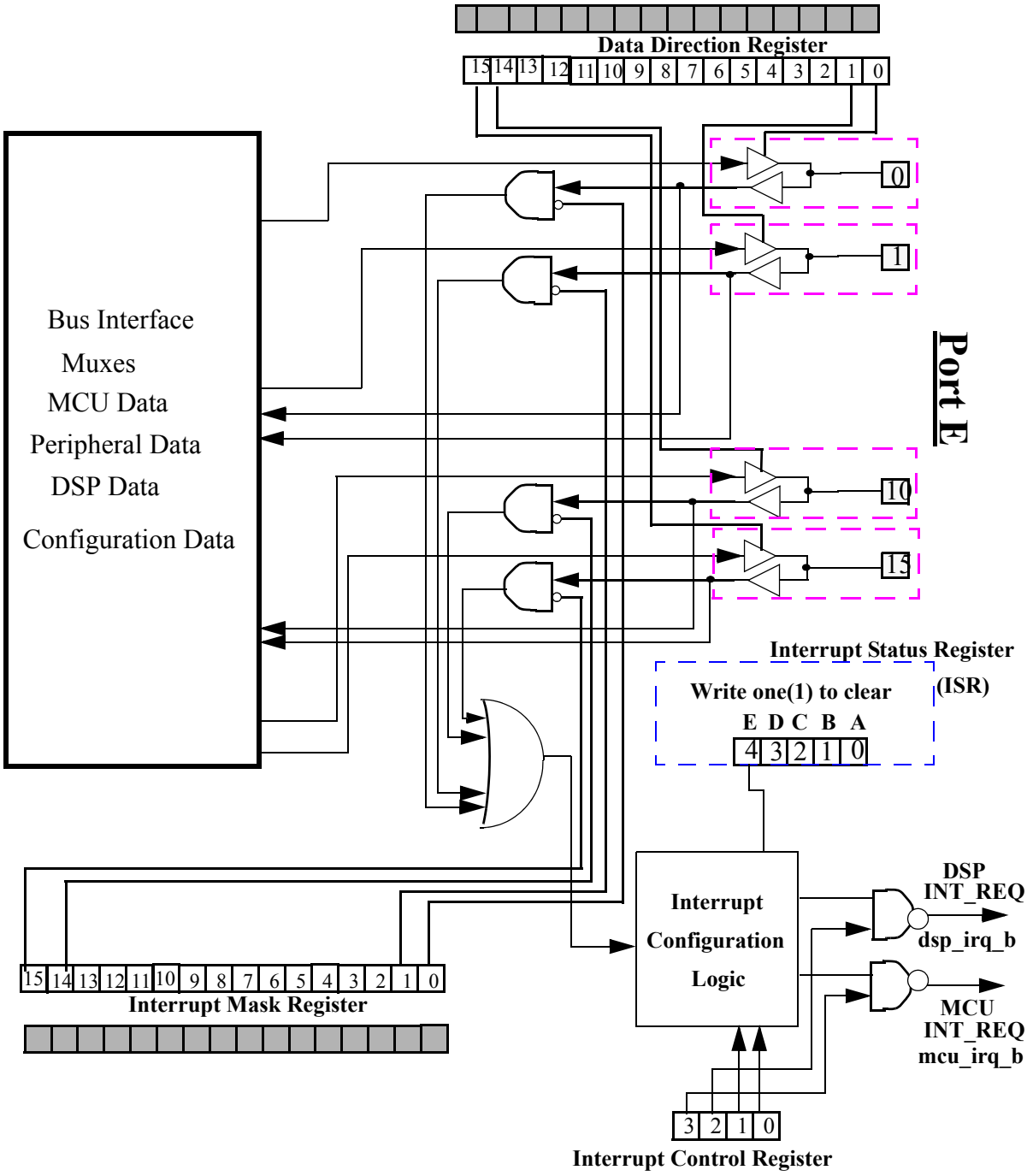


Figure 23-60. External Interrupt Control Block Diagram Port E

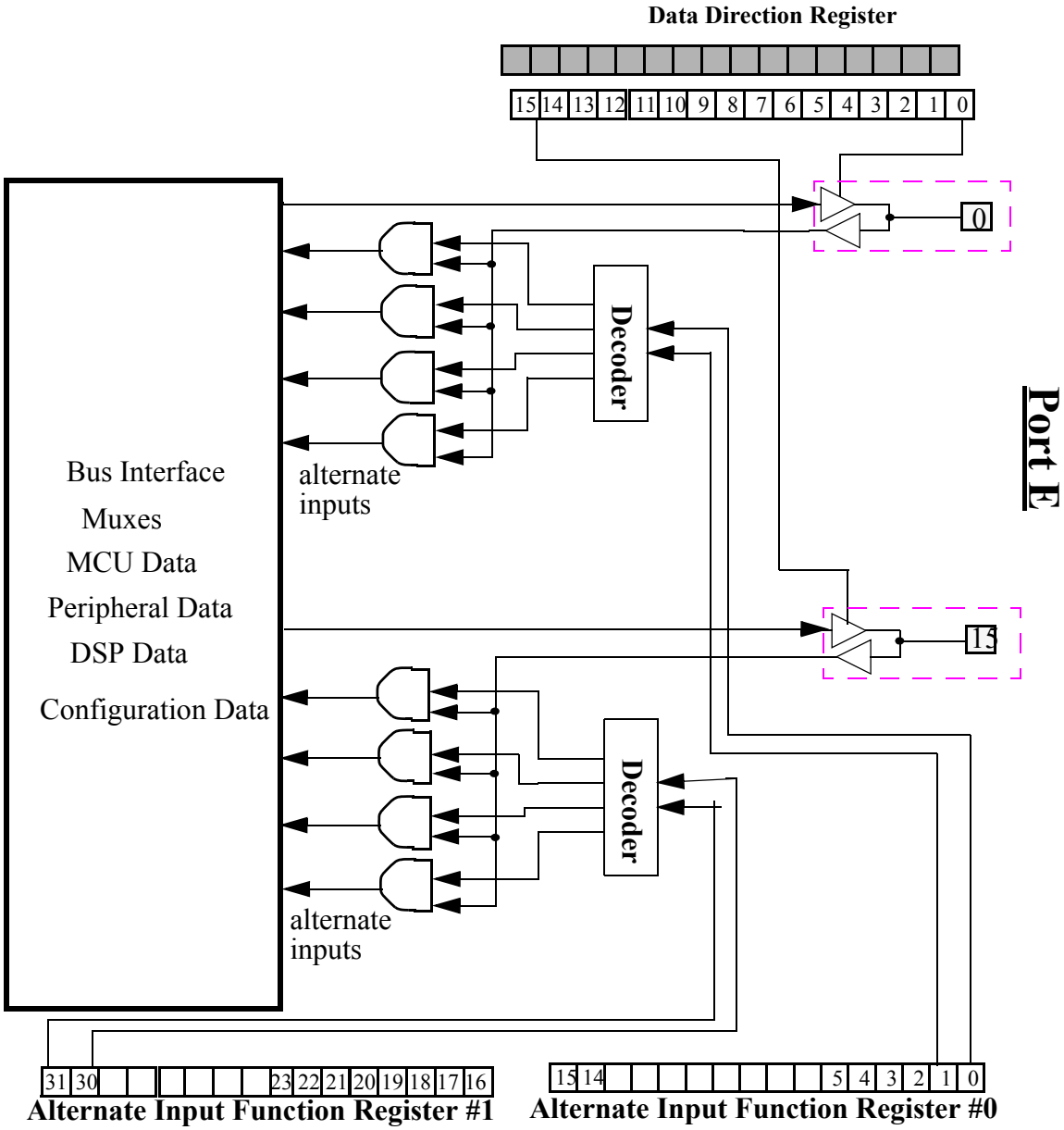


Figure 23-61. Alternate Input Functionality Block Diagram Port E

23.15 Detailed Register Descriptions (Port A-E)

The following registers show register names, addresses, and size of all registers used to configure the bi-directional Ports A-E. **GPIO supports only word writes to the registers. None of the GPIO registers are byte accessible.**

The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and is cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

23.15.1 Port A Register Description

This 32 bit register is used to select one of eight possible outputs for pins 9-0 of Port A.

READ: No restrictions

WRITE: No restrictions

PAConfReg0 Port A Configuration Register 0 **Addr $2484_1000**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			P9_S2	P9_S1	P9_S0	P8_S2	P8_S1	P8_S0	P7_S2	P7_S1	P7_S0	P6_S2	P6_S1	P6_S0	P5_S2	P5_S1
TYPE	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	1
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	P5_S0	P4_S2	P4_S1	P4_S0	P3_S2	P3_S1	P3_S0	P2_S2	P2_S1	P2_S0	P1_S2	P1_S1	P1_S0	P0_S2	P0_S1	P0_S0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0

Table 23-3. PAConfReg0 Description

Name	Description	Settings
Bit 31-30	**Reserved bit**	N/A
P9_S[2:0] Bits 29-27	**Pin 9 Select —** Select one of eight possible outputs for PortA pin 9.	**P9_S[2:0]** / **PortA, Pin9 Output Number**: 000 → 0; 001 → 1; ... → ...; 111 → 7
P8_S[2:0] Bits 26-24	**Pin 8 Select —** Select one of eight possible outputs for PortA pin 8.	**P8_S[2:0]** / **PortA, Pin8 Output Number**: 000 → 0; 001 → 1; ... → ...; 111 → 7

Table 23-3. PAConfReg0 Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>P7_S[2:0]
Bits 23-21</td><td>Pin 7 Select — Select one of eight possible outputs for PortA pin 7.</td><td>
<table>
<tr><th>P7_S[2:0]</th><th>PortA, Pin7 Output Number</th></tr>
<tr><td>000</td><td>0</td></tr>
<tr><td>001</td><td>1</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>7</td></tr>
</table>
</td></tr>
<tr><td>P6_S[2:0]
Bits 20-18</td><td>Pin 6 Select — Select one of eight possible outputs for PortA pin 6.</td><td>
<table>
<tr><th>P6_S[2:0]</th><th>PortA, Pin6 Output Number</th></tr>
<tr><td>000</td><td>0</td></tr>
<tr><td>001</td><td>1</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>7</td></tr>
</table>
</td></tr>
<tr><td>P5_S[2:0]
Bits 17-15</td><td>Pin 5 Select — Select one of eight possible outputs for PortA pin 5.</td><td>
<table>
<tr><th>P5_S[2:0]</th><th>PortA, Pin5 Output Number</th></tr>
<tr><td>000</td><td>0</td></tr>
<tr><td>001</td><td>1</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>7</td></tr>
</table>
</td></tr>
</table>

Table 23-3. PAConfReg0 Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>P4_S[2:0]
Bits 14-12</td><td>Pin 4 Select — Select one of eight possible outputs for PortA pin 4.</td><td>
<table>
<tr><th>P4_S[2:0]</th><th>PortA, Pin4 Output Number</th></tr>
<tr><td>000</td><td>0</td></tr>
<tr><td>001</td><td>1</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>7</td></tr>
</table>
</td></tr>
<tr><td>P3_S[2:0]
Bits 11-9</td><td>Pin 3 Select — Select one of eight possible outputs for PortA pin 3.</td><td>
<table>
<tr><th>P3_S[2:0]</th><th>PortA, Pin3 Output Number</th></tr>
<tr><td>000</td><td>0</td></tr>
<tr><td>001</td><td>1</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>7</td></tr>
</table>
</td></tr>
<tr><td>P2_S[2:0]
Bits 8-6</td><td>Pin 2 Select — Select one of eight possible outputs for PortA pin 2.</td><td>
<table>
<tr><th>P2_S[2:0]</th><th>PortA, Pin2 Output Number</th></tr>
<tr><td>000</td><td>0</td></tr>
<tr><td>001</td><td>1</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>7</td></tr>
</table>
</td></tr>
</table>

Table 23-3. PAConfReg0 Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>P1_S[2:0]
Bits 5-3</td><td>Pin 1 Select — Select one of eight possible outputs for PortA pin 1.</td><td>
<table>
<tr><th>P1_S[2:0]</th><th>PortA, Pin1 Output Number</th></tr>
<tr><td>000</td><td>0</td></tr>
<tr><td>001</td><td>1</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>7</td></tr>
</table>
</td></tr>
<tr><td>P0_S[2:0]
Bits 2-0</td><td>Pin 0 Select — Select one of eight possible outputs for PortA pin 0.</td><td>
<table>
<tr><th>P0_S[2:0]</th><th>PortA, Pin0 Output Number</th></tr>
<tr><td>000</td><td>0</td></tr>
<tr><td>001</td><td>1</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>7</td></tr>
</table>
</td></tr>
</table>

This 32 bit register is used to select one of eight possible outputs for pins 15-10 of Port A.

READ: No restrictions

WRITE: No restrictions

PAConfReg1 Port A Configuration Register 1 **Addr $2484_1004**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
															P15_S2	P15_S1
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	P15_S0	P14_S2	P14_S1	P14_S0	P13_S2	P13_S1	P13_S0	P12_S2	P12_S1	P12_S0	P11_S2	P11_S1	P11_S0	P10_S2	P10_S1	P10_S0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0

Table 23-4. PAConfReg1 Description

Name	Description	Settings
Bits 31-18	**Reserved bits**	N/A
P15_S[2:0] Bits 17-15	**Pin 15 Select—** Select one of eight possible outputs for PortA pin 15.	**P15_S[2:0]** / **PortA, Pin15 Output Number**: 000 → 0; 001 → 1; ... → ...; 111 → 7
P14_S[2:0] Bits 14-12	**Pin 14 Select —** Select one of eight possible outputs for PortA pin 14	**P14_S[2:0]** / **PortA, Pin14 Output Number**: 000 → 0; 001 → 1; ... → ...; 111 → 7

Table 23-4. PAConfReg1 Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>P13_S[2:0]
Bits 11-9</td><td>Pin 13 Select — Select one of eight possible outputs for PortA pin 13.</td><td>
<table>
<tr><th>P13_S[2:0]</th><th>PortA, Pin 13 Output Number</th></tr>
<tr><td>000</td><td>0</td></tr>
<tr><td>001</td><td>1</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>7</td></tr>
</table>
</td></tr>
<tr><td>P12_S[2:0]
Bits 8-6</td><td>Pin 12 Select — Select one of eight possible outputs for PortA pin 12.</td><td>
<table>
<tr><th>P12_S[2:0]</th><th>PortA, Pin12 Output Number</th></tr>
<tr><td>000</td><td>0</td></tr>
<tr><td>001</td><td>1</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>7</td></tr>
</table>
</td></tr>
<tr><td>P11_S[2:0]
Bits 5-3</td><td>Pin 11 Select — Select one of eight possible outputs for PortA pin 11.</td><td>
<table>
<tr><th>P11_S[2:0]</th><th>PortA, Pin11 Output Number</th></tr>
<tr><td>000</td><td>0</td></tr>
<tr><td>001</td><td>1</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>7</td></tr>
</table>
</td></tr>
</table>

Table 23-4. PAConfReg1 Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>P10_S[2:0]
Bits 2-0</td><td>Pin 10 Select — Select one of eight possible outputs for PortA pin 10.</td><td>
<table>
<tr><th>P10_S[2:0]</th><th>PortA, Pin10 Output Number</th></tr>
<tr><td>000</td><td>0</td></tr>
<tr><td>001</td><td>1</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>7</td></tr>
</table>
</td></tr>
</table>

This 32 bit register is used to configure the Port A pin as input or output. Each register bit corresponds to one Port A pin.

READ: No restrictions

WRITE: No restrictions

PADDirReg Port A Data Direction Register **Addr $2484_1008**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	DirP15	DirP14	DirP13	DirP12	DirP11	DirP10	DirP9	DirP8	DirP7	DirP6	DirP5	DirP4	DirP3	DirP2	DirP1	DirP0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1

Table 23-5. PADDirReg Description

Name	Description	Settings
Bits 31-16	**Reserved bits**	N/A
DirP[15:0] Bits 15-0	**Direction Pin—** Set direction for pin [15:0] of Port A.	0 = input 1 = output

This 32 bit register is used to screen an interrupt for each pin of Port A. Each register bit is the mask for each corresponding Port A pin.

READ: No restrictions

WRITE: No restrictions

PAIMaskReg Port A Interrupt Mask Register **Addr $2484_100C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Msk15	Msk14	Msk13	Msk12	Msk11	Msk10	Msk9	Msk8	Msk7	Msk6	Msk5	Msk4	Msk3	Msk2	Msk1	Msk0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-6. PAIMaskReg Description

Name	Description	Settings
Bits 31-16	**Reserved bits**	N/A
Msk[15:0] Bit 15-0	**Mask—** Mask bit for pin [15:0] of Port A.	0 = interrupt masked 1 = Interrupt not masked

This 32 bit register provides configuration of each of Port A's interrupt request signals to be high/low level sensitive and rising/falling edge. The DSP interrupt can only be masked by this register. The high/low level sensitive and rising/falling edge bits do not affect the DSP interrupts. The following table shows what each pin does.

READ: No restrictions

WRITE: No restrictions

PAIntCtrlReg Port A Interrupt Control Register **Addr $2484_1010**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
													MCU Mask	DSP Mask	Edge/ Level	Rise/fall High/low
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-7. PAIntCtrlReg Description

Name	Description	Settings
Bits 31-4	**Reserved**	N/A
MCU Mask Bit 3	**MCU Mask—** MCU interrupt mask bit.	0 = MCU interrupt disabled 1 = MCU interrupt enabled
DSP Mask Bit 2	**DSP Mask**— DSP interrupt mask bit.	0 = DSP interrupt disabled 1 = DSP interrupt enabled
Edge/Level Bit 1	**Edge/Level —** Edge/level interrupt selection control bit. Only valid for MCU interrupt.	0 = Edge sensitive 1 = Level sensitive
Rise/fall High/low Bit 0	**Rise/fall High/low**— rising/falling edge control or high or low level interrupt configuration bit. Only valid for MCU interrupt.	

Edge/ Level	Rise/fall High/low	Interrupt mode
0	0	rising edge
	1	falling edge
1	0	level high
	1	level low

This 32 bit register provides the 2 bit input code for the decoder used to determine which of the 4 possible devices will receive the input from any one of Port A pins 15-0. The register bits are grouped into twos to correspond to one of the 16 Port A pins.

READ: No restrictions

WRITE: No restrictions

PAAltInReg Port A Alternate Input Register **Addr $2484_1014**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Alt15 S1	Alt15 S0	Alt14 S1	Alt14 S0	Alt13 S1	Alt13 S0	Alt12 S1	Alt12 S0	Alt11 S1	Alt11 S0	Alt10 S1	Alt10 S0	Alt9S1	Alt9S0	Alt8S1	Alt8S0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Alt7S1	Alt7S0	Alt6S1	Alt6S0	Alt5S1	Alt5S0	Alt4S1	Alt4S0	Alt3S1	Alt3S0	Alt2S1	Alt2S0	Alt1S1	Alt1S0	Alt0S1	Alt0S0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-8. PAAltInReg Description

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>Alt15S[1:0]
Bits 31-30</td><td>Pin 15 Select — Select one of four possible devices to receive input from PortA pin 15.</td><td>
<table>
<tr><th>Alt15S[1:0]</th><th>PortA, Pin15 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Alt14S[1:0]
Bits 29-28</td><td>Pin 14 Select — Select one of four possible devices to receive input from PortA pin 14.</td><td>
<table>
<tr><th>Alt14S[1:0]</th><th>PortA, Pin14 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
</table>

Table 23-8. PAAltInReg Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>Alt13S[1:0]
Bits 27-26</td><td>Pin 13 Select — Select one of four possible devices to receive input from PortA pin 13.</td><td>
<table>
<tr><th>Alt13S[1:0]</th><th>PortA, Pin13 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Alt12S[1:0]
Bits 25-24</td><td>Pin 12 Select — Select one of four possible devices to receive input from PortA pin 12.</td><td>
<table>
<tr><th>Alt12S[1:0]</th><th>PortA, Pin12 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Alt11S[1:0]
Bits 23-22</td><td>Pin 11 Select — Select one of four possible devices to receive input from PortA pin 11.</td><td>
<table>
<tr><th>Alt11S[1:0]</th><th>PortA, Pin11 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Alt10S[1:0]
Bits 21-20</td><td>Pin 10 Select — Select one of four possible devices to receive input from PortA pin 10.</td><td>
<table>
<tr><th>Alt10S[1:0]</th><th>PortA, Pin10 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
</table>

Table 23-8. PAAltInReg Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>Alt9S[1:0]
Bits 19-18</td><td>Pin 9 Select — Select one of four possible devices to receive input from PortA pin 9.</td><td>
<table>
<tr><th>Alt9S[1:0]</th><th>PortA, Pin9 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Alt8S[1:0]
Bits 17-16</td><td>Pin 8 Select — Select one of four possible devices to receive input from PortA pin 8.</td><td>
<table>
<tr><th>Alt8S[1:0]</th><th>PortA, Pin8 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Alt7S[1:0]
Bits 15-14</td><td>Pin 7 Select — Select one of four possible devices to receive input from PortA pin 7.</td><td>
<table>
<tr><th>Alt7S[1:0]</th><th>PortA, Pin7 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
</table>

Table 23-8. PAAltInReg Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>Alt6S[1:0]
Bits 13-12</td><td>Pin 6 Select — Select one of four possible devices to receive input from PortA pin 6.</td><td>
<table>
<tr><th>Alt6S[1:0]</th><th>PortA, Pin6 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Alt5S[1:0]
Bits 11-10</td><td>Pin 5 Select — Select one of four possible devices to receive input from PortA pin 5.</td><td>
<table>
<tr><th>Alt5S[1:0]</th><th>PortA, Pin5 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Alt4S[1:0]
Bits 9-8</td><td>Pin 4 Select — Select one of four possible devices to receive input from PortA pin 4.</td><td>
<table>
<tr><th>Alt4S[1:0]</th><th>PortA, Pin4 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
</table>

Table 23-8. PAAltInReg Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>Alt3S[1:0]
Bits 7-6</td><td>Pin 3 Select — Select one of four possible devices to receive input from PortA pin 3.</td><td>
<table>
<tr><th>Alt3S[1:0]</th><th>PortA, Pin3 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Alt2S[1:0]
Bits 5-4</td><td>Pin 2 Select — Select one of four possible devices to receive input from PortA pin 2.</td><td>
<table>
<tr><th>Alt2S[1:0]</th><th>PortA, Pin2 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Alt1S[1:0]
Bits 3-2</td><td>Pin 1 Select — Select one of four possible devices to receive input from PortA pin 1.</td><td>
<table>
<tr><th>Alt1S[1:0]</th><th>PortA, Pin1 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
</table>

Table 23-8. PAAltInReg Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>Alt0S[1:0]
Bits 1-0</td><td>Pin 0 Select — Select one of four possible devices to receive input from PortA pin 0.</td><td>
<table>
<tr><th>Alt0S[1:0]</th><th>PortA, Pin0 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
</table>

DSPPADataReg DSP Port A Data Register **Addr Y:$FFD0**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	DSPPAData[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-9. DSPPADataReg Description

Name	Description	Settings
DSPPAData[15:0] Bits 15-0	**DSP Port A Data —** This 16 bit register is used to handle data between the DSP Bus and the GPIO. When the pin is configured as an output, the data written to this register is written directly to pin. A read from this register will always return the value at the pin. READ: No restrictions WRITE: No restrictions	N/A

MCUPADataReg MCU Port A Data Register **Addr $2484_1080**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	MCUPXData[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-10. MCUPADataReg Description

Name	Description	Settings
Bits 31-16	**Reserved**	N/A
MCUPAData[15:0] Bits 15-0	**MCU Port A Data —**This 16 bit register is used to handle data between the MCU Bus and the GPIO Port A. When the pin is configured as an output, the data written to this register is written directly to pin. A read from this register will always return the value at the pin. READ: No restrictions WRITE: No restrictions	N/A

23.15.2 Port B-E Register Description

This 32 bit register is used to select one of eight possible outputs for pins 9-0 of Port X (where X = Ports B,C,D, E).

READ: No restrictions

WRITE: No restrictions

PXConfReg0 Port X Configuration Register 0 **Addr (Port B)$2484_1018 (Port C)$2484_1030 (Port D)$2484_1048 (Port E)$2484_1060**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			P9_S2	P9_S1	P9_S0	P8_S2	P8_S1	P8_S0	P7_S2	P7_S1	P7_S0	P6_S2	P6_S1	P6_S0	P5_S2	P5_S1
TYPE	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET Port B	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
RESET Port C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RESET Port D	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1
RESET Port E	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	P5_S0	P4_S2	P4_S1	P4_S0	P3_S2	P3_S1	P3_S0	P2_S2	P2_S1	P2_S0	P1_S2	P1_S1	P1_S0	P0_S2	P0_S1	P0_S0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET Port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RESET Port C	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
RESET Port D	0	0	0	0	0	1	0	0	1	0	0	1	0	0	1	0
RESET Port E	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-11. PXConfReg0 Description

Name	Description	Settings
Bit 31-30	**Reserved bit**	N/A

Table 23-11. PXConfReg0 Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>P9_S[2:0]
Bits 29-27</td><td>Pin 9 Select — Select one of eight possible outputs for PortX pin 9.</td><td>
<table>
<tr><th>P9_S[2:0]</th><th>PortX, Pin9 Output Number</th></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>2</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>8</td></tr>
</table>
</td></tr>
<tr><td>P8_S[2:0]
Bits 26-24</td><td>Pin 8 Select — Select one of eight possible outputs for PortX pin 8.</td><td>
<table>
<tr><th>P8_S[2:0]</th><th>PortX, Pin8 Output Number</th></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>2</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>8</td></tr>
</table>
</td></tr>
<tr><td>P7_S[2:0]
Bits 23-21</td><td>Pin 7 Select — Select one of eight possible outputs for PortX pin 7.</td><td>
<table>
<tr><th>P7_S[2:0]</th><th>PortX, Pin7 Output Number</th></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>2</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>8</td></tr>
</table>
</td></tr>
<tr><td>P6_S[2:0]
Bits 20-18</td><td>Pin 6 Select — Select one of eight possible outputs for PortX pin 6.</td><td>
<table>
<tr><th>P6_S[2:0]</th><th>PortX, Pin6 Output Number</th></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>2</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>8</td></tr>
</table>
</td></tr>
</table>

Table 23-11. PXConfReg0 Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>P5_S[2:0]
Bits 17-15</td><td>Pin 5 Select — Select one of eight possible outputs for PortX pin 5.</td><td>
<table>
<tr><th>P5_S[2:0]</th><th>PortX, Pin5 Output Number</th></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>2</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>8</td></tr>
</table>
</td></tr>
<tr><td>P4_S[2:0]
Bits 14-12</td><td>Pin 4 Select — Select one of eight possible outputs for PortX pin 4.</td><td>
<table>
<tr><th>P4_S[2:0]</th><th>PortX, Pin4 Output Number</th></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>2</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>8</td></tr>
</table>
</td></tr>
<tr><td>P3_S[2:0]
Bits 11-9</td><td>Pin 3 Select — Select one of eight possible outputs for PortX pin 3.</td><td>
<table>
<tr><th>P3_S[2:0]</th><th>PortX, Pin3 Output Number</th></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>2</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>8</td></tr>
</table>
</td></tr>
<tr><td>P2_S[2:0]
Bits 8-6</td><td>Pin 2 Select — Select one of eight possible outputs for PortX pin 2.</td><td>
<table>
<tr><th>P2_S[2:0]</th><th>PortX, Pin2 Output Number</th></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>2</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>8</td></tr>
</table>
</td></tr>
</table>

Table 23-11. PXConfReg0 Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>P1_S[2:0]
Bits 5-3</td><td>Pin 1 Select — Select one of eight possible outputs for PortX pin 1.</td><td>
<table>
<tr><th>P1_S[2:0]</th><th>PortX, Pin1 Output Number</th></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>2</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>8</td></tr>
</table>
</td></tr>
<tr><td>P0_S[2:0]
Bits 2-0</td><td>Pin 0 Select — Select one of eight possible outputs for PortX pin 0.</td><td>
<table>
<tr><th>P0_S[2:0]</th><th>PortX, Pin0 Output Number</th></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>2</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>8</td></tr>
</table>
</td></tr>
</table>

This 32 bit register is used to select one of eight possible outputs for pins 15-10 of Port X. (where: X = B,C,D,E).

READ: No restrictions

WRITE: No restrictions

PXConfReg1 Port X Configuration Register 1

Addr
(Port B)$2484_101C
(Port C)$2484_1034
(Port D)$2484_104C
(Port E)$2484_1064

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
															P15_S2	P15_S1
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw
RESET Port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
RESET Port C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
RESET Port D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
RESET PoRTE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	P15_S0	P14_S2	P14_S1	P14_S0	P13_S2	P13_S1	P13_S0	P12_S2	P12_S1	P12_S0	P11_S2	P11_S1	P11_S0	P10_S2	P10_S1	P10_S0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET Port B	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
RESET Port C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RESET Port D	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
RESET Port E	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Table 23-12. PXConfReg1 Description

Name	Description	Settings
Bit 31-18	**Restricted**	N/A

Table 23-12. PXConfReg1 Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>P15_S[2:0]
Bit 17-15</td><td>Pin 15 Select—Select one of eight possible outputs for PortX pin 15.</td><td>
<table>
<tr><th>P15_S[2:0]</th><th>PortX, Pin15 Output Number</th></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>2</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>8</td></tr>
</table>
</td></tr>
<tr><td>P14_S[2:0]
Bits 14-12</td><td>Pin 14 Select — Select one of eight possible outputs for PortX pin 14.</td><td>
<table>
<tr><th>P14_S[2:0]</th><th>PortX, Pin14 Output Number</th></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>2</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>8</td></tr>
</table>
</td></tr>
<tr><td>P13_S[2:0]
Bits 11-9</td><td>Pin 13 Select — Select one of eight possible outputs for PortX pin 13.</td><td>
<table>
<tr><th>P13_S[2:0]</th><th>PortX, Pin13 Output Number</th></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>2</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>8</td></tr>
</table>
</td></tr>
</table>

Table 23-12. PXConfReg1 Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>P12_S[2:0]
Bits 8-6</td><td>Pin 12 Select — Select one of eight possible outputs for PortX pin 12.</td><td>
<table>
<tr><th>P12_S[2:0]</th><th>PortX, Pin12 Output Number</th></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>2</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>8</td></tr>
</table>
</td></tr>
<tr><td>P11_S[2:0]
Bits 5-3</td><td>Pin 11 Select — Select one of eight possible outputs for PortX pin 11.</td><td>
<table>
<tr><th>P11_S[2:0]</th><th>PortX, Pin11 Output Number</th></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>2</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>8</td></tr>
</table>
</td></tr>
<tr><td>P10_S[2:0]
Bits 2-0</td><td>Pin 10 Select — Select one of eight possible outputs for PortX pin 10.</td><td>
<table>
<tr><th>P10_S[2:0]</th><th>PortX, Pin10 Output Number</th></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>2</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111</td><td>8</td></tr>
</table>
</td></tr>
</table>

PXDDirReg Port X Data Direction Register

Addr
(Port B)$2484_1020
(Port C)$2484_1038
(Port D)$2484_1050
(Port E)$2484_1068

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	DirP15	DirP14	DirP13	DirP12	DirP11	DirP10	DirP9	DirP8	DirP7	DirP6	DirP5	DirP4	DirP3	DirP2	DirP1	DirP0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET Port B	1	1	1	1	1	1	1	0	1	0	1	0	0	0	0	0
RESET Port C	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
RESET Port D	1	1	0	0	0	1	1	1	1	1	1	0	1	1	1	1
RESET Port E	0	0	1	0	0	**0**	1	1	1	1	1	1	0	1	1	0

Table 23-13. PXDDirReg Description

Name	Description	Settings
Bit 31-16	**Restricted**	N/A
DirP[15:0] Bits 15-0	**Direction Pin—** Set direction for pin [15:0] of Port X. This 16 bit register is used to configure the Port X pin as input or output. Each register bit corresponds to one Port X pin. (where: X = B:E) READ: No restrictions WRITE: No restrictions	0 = input 1 = output

PXIMaskReg Port X Interrupt Mask Register

Addr
(Port B)$2484_1024
(Port C)$2484_103C
(Port D)$2484_1054
(Port E)$2484_106C

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Msk15	Msk14	Msk13	Msk12	Msk11	Msk10	Msk9	Msk8	Msk7	Msk6	Msk5	Msk4	Msk3	Msk2	Msk1	Msk0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET Port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RESET Port C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RESET Port D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RESET Port E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-14. PXIMaskReg Description

Name	Description	Settings
Bit 31-16	**Restricted**	N/A
Msk[15:0] Bits 15-0	**Mask—** Mask bit for pin [15:0] of Port X. This 16 bit register is used to screen an interrupt for each pin of Port X. Each register bit is the mask for each corresponding Port X pin. (where: X = B:E). READ: No restrictions WRITE: No restrictions	0 = interrupt masked 1 = interrupt not masked

This 32 bit register provides configuration of each of Port B-E's interrupt request signals to be high/low level sensitive and rising/falling edge. The DSP interrupt can only be masked by this register. The high/low level sensitive and rising/falling edge bits do not affect the DSP interrupts. The following table shows what each pin does:

READ: No restrictions

WRITE: No restrictions

PXIntCtrlReg Port X Interrupt Control Register

Addr
(Port B)$2484_1028
(Port C)$2484_1040
(Port D)$2484_1058
(Port E)$2484_1070

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
													MCU Mask	DSP Mask	Edge/ Level	Rise/fall High/low
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-15. PXIntCtrlReg Description

Name	Description	Settings
Bits 31-4	**Reserved**	N/A
MCU Mask Bit 3	MCU interrupt mask bit	0 = MCU interrupt disabled 1 = MCU interrupt enabled
DSP Mask Bit 2	DSP interrupt mask bit	0 = DSP interrupt disabled 1 = DSP interrupt enabled
Edge/ Level Bit 1	Edge/level interrupt selection control bit. Only valid for MCU interrupt.	0 = Edge sensitive 1 = Level sensitive

Table 23-15. PXIntCtrlReg Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>High/low
Rise/fall
Bit 0</td><td>Rising/falling edge control or high or low level interrupt configuration bit. Only valid for MCU interrupt.</td><td>
<table>
<tr><th>Edge/ Level</th><th>Rise/fall High/low</th><th>Interrupt mode</th></tr>
<tr><td rowspan="2">0</td><td>0</td><td>rising edge</td></tr>
<tr><td>1</td><td>falling edge</td></tr>
<tr><td rowspan="2">1</td><td>0</td><td>level high</td></tr>
<tr><td>1</td><td>level low</td></tr>
</table>
</td></tr>
</table>

This 32 bit register provides the 2 bit input code for the decoder used to determine which of the 4 possible devices will receive the input from any one of Port Xpins 15-0. The register bits are grouped into twos to correspond to one of the 16 Port X pins.

READ: No restrictions

WRITE: No restrictions

PXAltInReg Port XAlternate Input Register **Addr (Port B)$2484_102C (Port C)$2484_1044 (Port D)$2484_105C (Port E)$2484_1074**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Alt15 S1	Alt15 S0	Alt14 S1	Alt14 S0	Alt13 S1	Alt13 S0	Alt12 S1	Alt12 S0	Alt11 S1	Alt11 S0	Alt10 S1	Alt10 S0	Alt9S1	Alt9S0	Alt8S1	Alt8S0
RESET	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Alt7S1	Alt7S0	Alt6S1	Alt6S0	Alt5S1	Alt5S0	Alt4S1	Alt4S0	Alt3S1	Alt3S0	Alt2S1	Alt2S0	Alt1S1	Alt1S0	Alt0S1	Alt0S0
RESET	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-16. PXAltInReg Description

Name	Description	Settings
Alt15S[1:0] Bits 31-30	**Pin 15 Select —** Select one of four possible devices to receive input from PortX pin 15.	**Alt15S[1:0]** / **PortX, Pin15 Input Device**: 00 → 1; 01 → 2; 10 → 3; 11 → 4

Table 23-16. PXAltInReg Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>Alt14S[1:0]
Bits 29-28</td><td>Pin 14 Select — Select one of four possible devices to receive input from PortX pin 14.</td><td>
<table>
<tr><th>Alt14S[1:0]</th><th>PortX, Pin14 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Alt13S[1:0]
Bits 27-26</td><td>Pin 13 Select — Select one of four possible devices to receive input from PortX pin 13.</td><td>
<table>
<tr><th>Alt13S[1:0]</th><th>PortX, Pin13 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Alt12S[1:0]
Bits 25-24</td><td>Pin 12 Select — Select one of four possible devices to receive input from PortX pin 12.</td><td>
<table>
<tr><th>Alt12S[1:0]</th><th>PortX, Pin12 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
</table>

Table 23-16. PXAltInReg Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>Alt11S[1:0]
Bits 23-22</td><td>Pin 11 Select — Select one of four possible devices to receive input from PortX pin 11.</td><td>
<table>
<tr><th>Alt11S[1:0]</th><th>PortX, Pin11 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Alt10S[1:0]
Bits 21-20</td><td>Pin 10 Select — Select one of four possible devices to receive input from PortX pin 10.</td><td>
<table>
<tr><th>Alt10S[1:0]</th><th>PortX, Pin10 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Alt9S[1:0]
Bits 19-18</td><td>Pin 9 Select — Select one of four possible devices to receive input from PortX pin 9.</td><td>
<table>
<tr><th>Alt9S[1:0]</th><th>PortX, Pin9 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
</table>

Table 23-16. PXAltInReg Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>Alt8S[1:0]
Bits 17-16</td><td>Pin 8 Select — Select one of four possible devices to receive input from PortX pin 8.</td><td>
<table>
<tr><th>Alt8S[1:0]</th><th>PortX, Pin8 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Alt7S[1:0]
Bits 15-14</td><td>Pin 7 Select — Select one of four possible devices to receive input from PortX pin 7.</td><td>
<table>
<tr><th>Alt7S[1:0]</th><th>PortX, Pin7 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Alt6S[1:0]
Bits 13-12</td><td>Pin 6 Select — Select one of four possible devices to receive input from PortX pin 6.</td><td>
<table>
<tr><th>Alt6S[1:0]</th><th>PortX, Pin6 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
</table>

Table 23-16. PXAltInReg Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>Alt5S[1:0]
Bits 11-10</td><td>Pin 5 Select — Select one of four possible devices to receive input from PortX pin 5.</td><td>
<table>
<tr><th>Alt5S[1:0]</th><th>PortX, Pin5 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Alt4S[1:0]
Bits 9-8</td><td>Pin 4 Select — Select one of four possible devices to receive input from PortXpin 4.</td><td>
<table>
<tr><th>Alt4S[1:0]</th><th>PortX, Pin4 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Alt3S[1:0]
Bits 7-6</td><td>Pin 3 Select — Select one of four possible devices to receive input from PortX pin 3.</td><td>
<table>
<tr><th>Alt3S[1:0]</th><th>PortX, Pin3 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
</table>

Table 23-16. PXAltInReg Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>Alt2S[1:0]
Bits 5-4</td><td>Pin 2 Select — Select one of four possible devices to receive input from PortXpin 2.</td><td>
<table>
<tr><th>Alt2S[1:0]</th><th>PortX, Pin2 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Alt1S[1:0]
Bits 3-2</td><td>Pin 1 Select — Select one of four possible devices to receive input from PortX pin 1.</td><td>
<table>
<tr><th>Alt1S[1:0]</th><th>PortX, Pin1 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Alt0S[1:0]
Bits 1-0</td><td>Pin 0 Select — Select one of four possible devices to receive input from PortX pin 0.</td><td>
<table>
<tr><th>Alt0S[1:0]</th><th>PortX, Pin0 Input Device</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
</td></tr>
</table>

IntSelReg provides the 6 select bits for pins 0 to 4 of Port B and pins 4 to 8 of Port E used for the 64 to 1 muxes which select which internal interrupts will be visible on these pins.

READ: No restrictions

WRITE: No restrictions

IntSelReg Interrupt Select Register **Addr $2484_1078**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			IV4S5	IV4S4	IV4S3	IV4S2	IV4S1	IV4S0	IV3S5	IV3S4	IV3S3	IV3S2	IV3S1	IV3S0	IV2S5	IV2S4
TYPE	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	IV2S3	IV2S2	IV2S1	IV2S0	IV1S5	IV1S4	IV1S3	IV1S2	IV1S1	IV1S0	IV0S5	IV0S4	IV0S3	IV0S2	IV0S1	IV0S0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-17. IntSelReg Description

Name	Description	Settings
Bits 31-30	Reserved	N/A
IV4S[5:0] Bits 29-24	**Interrupt Visibility Pin 4 Select —** Select one of sixty-four (64) possible internal interrupts to be visible on Port B, Pin 0 and Port E, Pin 8.	**IV4S[5:0]** / **PortB, Pin0** Port E, PIn8 Interrupt Number 000000 — 0 000001 — 1 ... — ... 111111 — 63
IV3S[5:0] Bits 23-18	**Interrupt Visibility Pin 3 Select —** Select one of sixty-four (64) possible internal interrupts to be visible on Port B, Pin 1 and Port E, Pin 7.	**IV3S[5:0]** / **PortB, Pin1** Port E, pin 7 Interrupt Number 000000 — 0 000001 — 1 ... — ... 111111 — 63

Table 23-17. IntSelReg Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>IV2S[5:0]
Bits 17-12</td><td>Interrupt Visibility Pin 2 Select— Select one of sixty-four (64) possible internal interrupts to be visible on Port B, Pin 2 and Port E, Pin 6.</td><td>
<table>
<tr><th>IV2S[5:0]</th><th>PortB, Pin2
Port E, Pin 6
Interrupt Number</th></tr>
<tr><td>000000</td><td>0</td></tr>
<tr><td>000001</td><td>1</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111111</td><td>63</td></tr>
</table>
</td></tr>
<tr><td>IV1S[5:0]
Bits 11-6</td><td>Interrupt Visibility Pin 1 Select — Select one of sixty-four (64) possible internal interrupts to be visible on Port B, Pin 3 and Port E, Pin 5.</td><td>
<table>
<tr><th>IV1S[5:0]</th><th>PortB, Pin3
Port E, Pin 5
Interrupt Number</th></tr>
<tr><td>000000</td><td>0</td></tr>
<tr><td>000001</td><td>1</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111111</td><td>63</td></tr>
</table>
</td></tr>
<tr><td>IV0S[5:0]
Bits 5-0</td><td>Interrupt Visibility Pin 0 Select— Select one of sixty-four (64) possible internal interrupts to be visible on Port B, Pin 4and Port E, Pin 4.</td><td>
<table>
<tr><th>IV0S[5:0]</th><th>PortB, Pin4
Port E, Pin 4
Interrupt Number</th></tr>
<tr><td>000000</td><td>0</td></tr>
<tr><td>000001</td><td>1</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111111</td><td>63</td></tr>
</table>
</td></tr>
</table>

This 32 bit register provides 5 interrupt status signals, one for each of the 5 I/O Ports A-E.

READ: No restrictions

WRITE: write 1 to clear

IntStatReg Interrupt Status Register **Addr $2484_107C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
												IStatE	IStatD	IStatC	IStatB	IStatA
TYPE	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-18. IntStatReg Description

Name	Description	Settings
Bit 31-5	**Reserved**	N/A
IStat[E:A] Bits 4-0	**Interrupt Status**— Status bit for Port E Interrupt - Port A Interrupt, respectively	0 = Interrupt not active 1 = Interrupt active

MCUPXDataReg MCU Port X Data Register

Addr
(Port B)$2484_1084
(Port C)$2484_1088
(Port D)$2484_108C
(Port E)$2484_1090

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	MCUPXData[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-19. MCUPXDataReg Description

Name	Description	Settings
Bit 31-16	**Reserved**	N/A
MCUPXData[15:0] Bits 15-0	**MCU Port X Data —**This 16 bit register is used to handle data between the MCU Bus and the GPIO (Port X = B:E). When the pin is configured as an output, the data written to this register is written directly to pin. A read from this register will always return the value at the pin. READ: No restrictions WRITE: No restrictions	

DSPPXDataReg DSP Port X Data Register

Addr
(Port B) Y:$FFD1
(Port C) Y:$FFD2
(Port D) Y:$FFD3
(Port E) Y:$FFD4

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	DSPPXData[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-20. DSPPXDataReg Description

Name	Description	Settings
DSPPXData[15:0] Bits 15-0	**DSP Port X Data —** This 16 bit register is used to handle data between the DSP Bus and the GPIO. When the pin is configured as an output, the data written to this register is written directly to pin. A read from this register will always return the value at the pin (Port X = B:E). READ: No restrictions WRITE: No restrictions	N/A

23.15.3 Other Registers Description

MCUBGref MCU BandGap Reference Register **Addr $2484_1094**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
								Vrefen								BGrdy
TYPE	r	r	r	r	r	r	r	rw	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-21. MCUBGref Register Description

Name	Description	Settings
Bit 31-16	**Reserved**	N/A
Vrefen Bits 8	**Vref Enable —** This bit enables the BandGap Reference analog block from the MCU side; the Band-Gap can be enabled by either the MCU or the DSP READ: No restrictions WRITE: No restrictions	0 = Band-Gap is not enabled by the MCU 1 = Enable the BandGap Reference
BGrdy Bits 0	**Vref Enable —** This status bit indicates to the MCU that the Band-Gap is ready for enabling other analog modules that needs its operation READ: No restrictions	0 = Band-Gap is not ready 1 = Band-Gap is ready

Note: The MCU bandgap reference register is not used for Neptune.

DSPBGref DSP BandGap Reference Register **Addr Y:$FFD7**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
								Vrefen								BGrdy
TYPE	r	r	r	r	r	r	r	rw	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-22. DSPBGref Register Description

Name	Description	Settings
Vrefen Bits 8	**Vref Enable —** This bit enables the BandGap Reference analog block from the DSP side; the Band-Gap can be enabled by either the MCU or the DSP READ: No restrictions WRITE: No restrictions	0 = Band-Gap is not enabled by the DSP 1 = Enable the BandGap Reference
BGrdy Bits 0	**Vref Enable —** This status bit indicates to the DSP that the Band-Gap is ready for enabling other analog modules that needs its operation READ: No restrictions	0 = Band-Gap is not ready 1 = Band-Gap is ready

Note: The DSP bandgap reference register is not used for Neptune.

23.16 GPIO Test Mode Functions

GPIO supports the following test modes:

23.16.1 Bus Master Mode

In Bus Master mode, GPIO provides 22 address bits, 5 control signals, 32 data signals to the TCM for bit mapping and debugging internal RAM and ROM. Depending on the tcm_16bit_mode_en control from the TCM, we have 16 data_in and 16 data_out or 32 data_in signals muxed on the GPIO. These are spread out in Ports A to D. In AHB Test Mode, the GPIO turns control of several bi-directional pins over to the Alternate Master Test Block. The various test mode control signals are provided by the TCM to the GPIO. In memtest mode, port A is configured for input, port B for address and controls, Port C for address and controls and Port D for input or output depending on TCM control signals.

23.16.2 Scan Mode

In Scan mode, the Port A - D are configured to provide 31 scan inputs, 31 scan outputs and 5 control signals to the TCM.

23.16.3 Bist Mode

Refer to latest BIST xls sheet for the ports configurations of GPIO in BIST mode.

23.16.4 DSP Master Mode

In DSP master mode, GPIO routes bdio[23-0] to and from the DSP and the DSP read, write signals from the DSP on portA, port B and Port C.

23.16.5 DSP Address Trace Mode

In DSP address trace mode, PortA[13], PortB[2], PortB[6-4], PortB[15], PortC[5-3], Port C[7], PortC[12-10], PortC[14], PortE[11-10], and PortE[15-13] are configured to mux dsp address bus.

The test mode overrides any of the configuration values written to the ports' alternate input registers, configuration registers and the data direction registers.

23.17 GPIO Test Mode Muxing

Table 23-23. Test Mode Muxing

Port	Bus Master mode	Scan mode	DSP master mode	DSP address trace mode
A0	AHBT_Data_in0	scan_out1	bdio0	N/A
A1	AHBT_Data_in1	scan_out2	bdio1	N/A
A2	AHBT_Data_in2	scan_out3	bdio2	N/A
A3	AHBT_Data_in3	scan_out4	bdio3	N/A
A4	AHBT_Data_in4	scan_out5	bdio4	N/A

Table 23-23. Test Mode Muxing

Port	Bus Master mode	Scan mode	DSP master mode	DSP address trace mode
A5	AHBT_Data_in5	scan_in1	bdio5	N/A
A6	AHBT_Data_in6	scan_in2	bdio6	N/A
A7	AHBT_Data_in7	scan_out6	bdio7	N/A
A8	AHBT_Data_in8	scan_out7	bdio8	N/A
A9	AHBT_Data_in9	scan_out8	bdio9	N/A
A10	AHBT_Data_in10	scan_out9	bdio10	N/A
A11	AHBT_Data_in11	scan_in3	bdio11	N/A
A12	AHBT_Data_in12	scan_out10	bdio12	N/A
A13	AHBT_Data_in13	scan_out11	bdio13	DSP_PGAD4
A14	AHBT_Data_in14	scan_in4	bdio14	N/A
A15	AHBT_Data_in15	scan_in5	bdio15	N/A
B0	gpio_mcu_clk	gpio_mcu_clk	N/A	N/A
B1	master	master	master	master
B2	TRES_b	TRES_b	N/A	DSP_PGAD5
B3	test_RW	gpio_scan_test_clk	N/A	N/A
B4	AHBT_Addr0	scan_in6	N/A	DSP_PGAD3
B5	N/A	scan_in28	N/A	DSP_PGAD1
B6	N/A	scan_in29	N/A	DSP_PGAD2
B7	AHBT_Addr1	scan_in7	N/A	N/A
B8	AHBT_Addr2	scan_in8	N/A	N/A
B9	AHBT_Addr3	scan_in9	N/A	N/A
B10	AHBT_Addr4	scan_in10	N/A	N/A
B11	AHBT_Addr5	scan_in11	N/A	N/A
B12	AHBT_Addr6	scan_in12	bdio16	N/A
B13	AHBT_Addr7	scan_in13	N/A	N/A
B14	AHBT_Addr8	scan_in14	N/A	N/A
B15	AHBT_Addr9	scan_in15	N/A	DSP_PGAD13
C0	N/A	scan_clock	N/A	N/A
C1	AHBT_Addr10	scan_in16	bdio17	N/A
C2	AHBT_Addr11	scan_in17	bdio18	N/A
C3	AHBT_Addr12	scan_in18	bdio19	DSP_PGAD8
C4	AHBT_Addr13	scan_in19	bdio20	DSP_PGAD9

Table 23-23. Test Mode Muxing

Port	Bus Master mode	Scan mode	DSP master mode	DSP address trace mode
C5	AHBT_Addr14	scan_in20	bdio21	DSP_PGAD14
C6	AHBT_Addr15	scan_in21	bdio22	N/A
C7	EXEC	scan_enable	bdio23	DSP_PGAD6
C8	AHBT_Addr21	scan_in22	dsp_rd_b	N/A
C9	N/A	N/A	N/A	N/A
C10	AHBT_Addr20	scan_in23	dsp_wr_b	DSP_PGAD7
C11	AHBT_Addr16	scan_in24	N/A	DSP_PGAD15
C12	AHBT_Addr17	scan_in25	N/A	DSP_PGAD12
C13	AHBT_Addr18	scan_in26	N/A	N/A
C14	AHBT_Addr19	scan_in27	N/A	DSP_PGAD14/DBG_PAGE2
C15	N/A	N/A	N/A	N/A
D0	AHBT_DO0/DI16	scan_out12	N/A	N/A
D1	AHBT_DO1/DI17]	scan_out13	N/A	N/A
D2	AHBT_DO2/DI18	scan_out14	N/A	N/A
D3	AHBT_DO3/DI19	scan_out15	N/A	N/A
D4	AHBT_DO4/DI20	scan_out16	N/A	N/A
D5	AHBT_DO5/DI21	scan_out17	N/A	N/A
D6	AHBT_DO6/DI22	scan_out18	N/A	N/A
D7	AHBT_DO7/DI23	scan_out19	N/A	N/A
D8	AHBT_DO8/DI24	scan_out20	N/A	N/A
D9	AHBT_DO9/DI25	scan_out21	N/A	N/A
D10	AHBT_DO10/DI26	scan_out22	N/A	N/A
D11	AHBT_DO11/DI27	scan_out23	N/A	N/A
D12	AHBT_DO12/DI28	scan_out24	N/A	N/A
D13	AHBT_DO13/DI29	scan_out25	N/A	N/A
D14	AHBT_DO14/DI30	scan_out26	N/A	N/A
D15	AHBT_DO15/DI31	scan_out27	N/A	N/A
E0	N/A	N/A	N/A	N/A
E1	N/A	N/A	N/A	N/A
E2	N/A	N/A	N/A	N/A
E3	N/A	N/A	N/A	N/A

Table 23-23. Test Mode Muxing

Port	Bus Master mode	Scan mode	DSP master mode	DSP address trace mode
E4	N/A	N/A	N/A	N/A
E5	N/A	N/A	N/A	N/A
E6	N/A	N/A	N/A	N/A
E7	N/A	N/A	N/A	N/A
E8	N/A	N/A	N/A	N/A
E9	N/A	N/A	N/A	N/A
E10	N/A	scan_in30	N/A	DSP_PGAD10
E11	N/A	scan_in31	N/A	DSP_PGAD11
E12	N/A	scan_out28	N/A	N/A
E13	N/A	scan_out29	N/A	DSP_PGAD12
E14	N/A	scan_out30	N/A	N/A
E15	TCM_GPIO_OUT[16]	scan_out31	N/A	DSP_PGAD0

23.18 GPIO BIST Mode Muxing:

Table 23-24. DSP BIST Mode Muxing Table

DSP BIST	Pins	TestName Pin	Dir.	Mode
D10	USB_TXENB	DSP BIST Clock(GCLKW)	Input	Prod+Bit
E3	EBW	ipt_dsp_bist_done	Ouput	Prod+Bit
E10	DSP_PGAD10	ipt_dsp_bist_fail	Output	Prod+Bit
E11	DSP_PGAD11	ipt_dsp_cfg	Output	Prod+Bit
A0	CKO (Bitmap Mode)	sab_bmap_compr[0]	Output	Bit Map
A1	RX_ EN_OUT	sab_bmap_compr[1]	Output	Bit Map
A2	L1T /TOUT5	sab_bmap_compr[2]	Output	Bit Map
A3	A2D I/GSM_DCS	sab_bmap_compr[3]	Output	Bit Map
A4	L1T /TOUT7	sab_bmap_compr[4]	Output	Bit Map
A5	L1T /TOUT8	sab_bmap_compr[5]	Output	Bit Map
A6	L1T /TOUT9	sab_bmap_compr[6]	Output	Bit Map
A7	L1T /TOUT10	sab_bmap_compr[7]	Output	Bit Map
A8	L1T /TOUT11	sab_bmap_compr[8]	Output	Bit Map

Table 23-24. DSP BIST Mode Muxing Table

DSP BIST	Pins	TestName Pin	Dir.	Mode
A9	INT 0	sab_bmap_compr[9]	Output	Bit Map
A10	INT1	sab_bmap_compr[10]	Output	Bit Map
A11	USB/SUSPEND	sab_bmap_compr[11]	Output	Bit Map
C0	UAR T/CTS1	sab_bmap_compr[12]	Output	Bit Map
A13	INT4	sab_bmap_compr[13]	Output	Bit Map
A14	WDOG	sab_bmap_compr[14]	Output	Bit Map
B6	UAR T2/RXD2	sab_bmap_compr[15]	Output	Bit Map
B7	SIM _RST	sab_bmap_compr[16]	Output	Bit Map
B8	SIM _PD	sab_bmap_compr[17]	Output	Bit Map
B9	SIM _CLK	sab_bmap_compr[18]	Output	Bit Map
B10	SIM_DIO	sab_bmap_compr[19]	Output	Bit Map
B11	SIM_VCCEN	sab_bmap_compr[20]	Output	Bit Map
B13	LDC_CLK	sab_bmap_compr[21]	Output	Bit Map
B14	LCD_CS	sab_bmap_compr[22]	Output	Bit Map
B15	LCD_RS	sab_bmap_compr[23]	Output	Bit Map
C2	ADC _SYNC	sab_bmap_addr[0]	Output	Bit Map
C3	COL UMN0	sab_bmap_addr[1]	Output	Bit Map
C4	COL UMN1	sab_bmap_addr[2]	Output	Bit Map
D3	MQS PI/QSCKA	sab_bmap_addr[3]	Output	Bit Map
D4	MQS PI/MISOA	sab_bmap_addr[4]	Output	Bit Map
D6	SPI _CS0	sab_bmap_addr[5]	Output	Bit Map
D14	USB_VPOUT	sab_bmap_addr[6]	Output	Bit Map
D7	SPI_CS1	sab_bmap_addr[7]	Output	Bit Map
D8	SPI_CS2	sab_bmap_addr[8]	Output	Bit Map
D9	A2DIGL/DM_CS1	sab_bmap_addr[9]	Output	Bit Map
D10	USB_TXENB	DSP BIST Clock (GCLKW)	In	Prod
E3	EBW	sio_din	In	Prod
E15	DSP_PGAD0	sio_wr	In	Prod
E11	DSP_PGAD11	sio_rd	In	Prod

Table 23-24. DSP BIST Mode Muxing Table

DSP BIST	Pins	TestName Pin	Dir.	Mode
E13	DSP_PGAD12	sio_en_shift	In	Prod
E14	MISOB(Alt)	sio_dout	Out	Prod

Table 23-25. Analog Test BIST Mode Muxing Table

Analog Test BIST Mode	Pin Name	TestPin Name	Dir	Mode
A0	CKO	GPADC/A2D_OUT[0]	Out	Prod Mode
A1	RX_EN_OUT	GPADC/A2D_OUT[1]	Out	Prod Mode
A2	L1T/TOUT5	GPADC/A2D_OUT[2]	Out	Prod Mode
A3	A2DI/GSM_DCS	GPADC/A2D_OUT[3]	Out	Prod Mode
A4	L1T/TOUT7	GPADC/A2D_OUT[4]	Out	Prod Mode
A5	L1T/TOUT8	GPADC/A2D_OUT[5]	Out	Prod Mode
A6	L1T/TOUT9	GPADC/A2D_OUT[6]	Out	Prod Mode
A7	L1T/TOUT10	GPADC/A2D_OUT[7]	Out	Prod Mode
A8	L1T/TOUT11	GPADC/A2D_OUT[8]	Out	Prod Mode
A9	INT0	GPADC/A2D_OUT[9]	Out	Prod Mode
A15	SAP/SC0A	rxcproc/sclk_out	Out	Prod Mode
B2	SAP/STDA	Test Reset	In	Prod Mode
B3	SAP/SC2A	Frame Sync	In	Prod Mode
B4	SAP/SRDA	RxCPROC/SRDx	Out	Prod Mode
B12	DMAC/SDATA	A2DIGL/STDB1	In	Prod Mode
C1	RTCK	GPADC/END_ADC	Out	Prod Mode
D0	A2DIGL/RF_CLK	FVMAIN	In	Prod Mode
D3	MQSPI/QSCKA	A2DIGL/SPI_CLK	In	Prod Mode
D4	MQSPI/MISOA	A2DIGL/SPI_DATA1	In	Prod Mode
D5	MQSPI/MOSIA	A2DIGL/TCONVERT	In	Prod Mode
D7	MQSPI/SPI_CS1	A2DIGL/CE1	In	Prod Mode
D8	MQSPI/SPI_CS2	A2DIGL/RXACQ1	In	Prod Mode
D9	MQSP/SPI_CS3	A2DIGL/DMCS1	In	Prod Mode
D13	USB_XRXD	rxcproc/sdfs	Out	Prod Mode

Table 23-25. Analog Test BIST Mode Muxing Table

Analog Test BIST Mode	Pin Name	TestPin Name	Dir	Mode
D15	USB_VMOUT	a2digl/TX_CLK	Out	Prod Mode

Table 23-26.

Codec BISTPRODUCTION MODE	Pins	Test Name Pin	Dir	Mode
A10	INT1	CODEC0	I/O	Prod Mode
A11	USB/SUSPEND	CODEC1	I/O	Prod Mode
C0	UART/CTS1	CODEC2	I/O	Prod Mode
A13	INT4	CODEC3	I/O	Prod Mode
A14	WDOG	CODEC4	I/O	Prod Mode
B7	SIM_RST	CODEC5	I/O	Prod Mode
B8	SIM_PD	CODEC6	I/O	Prod Mode
B9	SIM_CLK	CODEC7	I/O	Prod Mode
B10	SIM_DIO	CODEC8	I/O	Prod Mode
B11	SIM_VCCEN	CODEC9	I/O	Prod Mode
B13	LCD_CLK	CODEC10	I/O	Prod Mode
B14	LCD_CS	CODEC11	I/O	Prod Mode
B15	LCD_RS	CODEC12	I/O	Prod Mode
C2	ADC_SYNC	CODEC13	I/O	Prod Mode
C3	COLUMN0	CODEC14	I/O	Prod Mode
C4	COLUMN1	CODEC15	I/O	Prod Mode
D6	MQSP/SPI_CS0	CODEC_STROBE	Out	Prod Mode
D14	USB_VPOUT	CODEC_DIR	In	Prod Mode
E14	MISOB(Alt)	CODEC_BIST_CLOCK	In	Prod Mode

Table 23-27. MCU BIST Mode Muxing Table

MCU BIST MODE	Pins	TestName Pin	Dir	Mode
C5	ROW0	ipt_mem_bist_fail	Out	Prod+Bit
C6	ROW1	ipt_mem_bist_data_out[0]	Out	Bit Map
C7	ROW4	ipt_mem_bist_data_out[1]	Out	Bit Map
C8	ROW5	ipt_mem_bist_data_out[2]	Out	Bit Map
C10	ROW6	ipt_mem_bist_data_out[3]	Out	Bit Map
C11	ROW7	ipt_mem_bist_data_out[4]	Out	Bit Map
C12	PC12	ipt_mem_bist_data_out[5]	Out	Bit Map
C13	JTAG/SJC_MOD	ipt_mem_bist_data_out[6]	Out	Bit Map
C14	PC14	ipt_mem_bist_data_out[7]	Out	Bit Map
D12	USB_VMIN	ipt_mem_bist_done	Out	Prod+Bit
D1	A2D/RF_DATA	ipt_mem_bist_invoke	In	Prod+Bit
D2	A2D/RF_CS	ipt_mem_bist_reset	In	Prod+Bit

23.19 GPIO Module Pin List

Table 23-28 lists all the pins in the GPIO module

Table 23-28. GPIO Module Pin List

Pin Name	Direction	Description
IP signals		
ipg_clk	Input	IP clock
ips_wdata[31:0]	Input	IP write data bus
ipg_async_hard_reset_b	Input	IP reset
ips_rwb	Input	IP read/write
ips_byte_31_24	Input	IP byte [31-24] enable
ips_byte_23_16	Input	IP byte [23-16] enable
ips_byte_15_8	Input	IP byte [15-8] enable
ips_byte_7_0	Input	IP byte [7-0] enable
ips_addr[31:0]	Input	IP address bus
ips_module_en	Input	IP module enable

Table 23-28. GPIO Module Pin List

Pin Name	Direction	Description
ips_rdata[31:0]	Output	IP read data
ips_xfr_err	Output	IP transfer error signal
ips_xfr_wait	Output	IP transfer wait signal
SONYXU signals		
gdb_wr [15:0]	Input	DSP write data bus
pires	Input	DSP reset
gclkw_b	Input	DSP clock inverted
dsp_ngtm	Input	from dsp jtag register for dsp master mode and dsp address trace mode
mcab_wr	Input	peripheral I/O space write address
mcab_rd	Input	peripheral I/O space read address
mcdis0	Input	stall causes disable of t0 phase
mcdis1	Input	stall causes disable of t1 phase
mcsely_wr	Input	DSP Y memory I/O space write
mcsely_rd	Input	DSP Y memory I/O space read
dsp_nsertst	Input	DSP Serial Test Mode
bdatoe	Input	bdio output enable
bdio_rd[23:0]	Output	Port A readdata
gdb_rd[15:0]	Output	DSP peripheral I/O space read data
// GPIO signals		
gpio_porta_datain[15:0]	Input	data from pads to gpio port A[15:0]
gpio_portb_datain[15:0]	Input	data from pads to gpio port B[15:0]
gpio_portc_datain[15:0]	Input	data from pads to gpio port C[15:0]
gpio_portd_datain[15:0]	Input	data from pads to gpio port D[15:0]
gpio_porte_datain[15:0]	Input	data from pads to gpio port E[15:0]
gpio_porta0_funcout[5:0]	Input	signals from chip input to porta0 into 8 to 1 mux [5:0]
gpio_porta1_funcout[5:0]	Input	signals from chip input to porta1 into 8 to 1 mux [5:0]
gpio_porta2_funcout[5:0]	Input	signals from chip input to porta2 into 8 to 1 mux [5:0]
gpio_porta3_funcout[5:0]	Input	signals from chip input to porta3 into 8 to 1 mux [5:0]
gpio_porta4_funcout[5:0]	Input	signals from chip input to porta4 into 8 to 1 mux [5:0]

Table 23-28. GPIO Module Pin List

Pin Name	Direction	Description
gpio_porta5_funcout[5:0]	Input	signals from chip input to porta5 into 8 to 1 mux [5:0]
gpio_porta6_funcout[5:0]	Input	signals from chip input to porta6 into 8 to 1 mux [5:0]
gpio_porta7_funcout[5:0]	Input	signals from chip input to porta7 into 8 to 1 mux [5:0]
gpio_porta8_funcout[5:0]	Input	signals from chip input to porta8 into 8 to 1 mux [5:0]
gpio_porta9_funcout[5:0]	Input	signals from chip input to porta9 into 8 to 1 mux [5:0]
gpio_porta10_funcout[5:0]	Input	signals from chip input to porta10 into 8 to 1 mux [5:0]
gpio_porta11_funcout[5:0]	Input	signals from chip input to porta11 into 8 to 1 mux [5:0]
gpio_porta12_funcout[5:0]	Input	signals from chip input to porta12 into 8 to 1 mux [5:0]
gpio_porta13_funcout[5:0]	Input	signals from chip input to porta13 into 8 to 1 mux [5:0]
gpio_porta14_funcout[5:0]	Input	signals from chip input to porta14 into 8 to 1 mux [5:0]
gpio_porta15_funcout[5:0]	Input	signals from chip input to porta15 into 8 to 1 mux [5:0]
gpio_portb0_funcout[5:0]	Input	signals from chip input to portb0 into 8 to 1 mux [5:0]
gpio_portb1_funcout[5:0]	Input	signals from chip input to portb1 into 8 to 1 mux [5:0]
gpio_portb2_funcout[5:0]	Input	signals from chip input to portb2 into 8 to 1 mux [5:0]
gpio_portb3_funcout[5:0]	Input	signals from chip input to portb3 into 8 to 1 mux [5:0]
gpio_portb4_funcout[5:0]	Input	signals from chip input to portb4 into 8 to 1 mux [5:0]
gpio_portb5_funcout[5:0]	Input	signals from chip input to portb5 into 8 to 1 mux [5:0]
gpio_portb6_funcout[5:0]	Input	signals from chip input to portb6 into 8 to 1 mux [5:0]
gpio_portb7_funcout[5:0]	Input	signals from chip input to portb7 into 8 to 1 mux [5:0]
gpio_portb8_funcout[5:0]	Input	signals from chip input to portb8into 8 to 1 mux [5:0]
gpio_portb9_funcout[5:0]	Input	signals from chip input to portb9 into 8 to 1 mux [5:0]
gpio_portb10_funcout[5:0]	Input	signals from chip input to portb10 into 8 to 1 mux [5:0]
gpio_portb11_funcout[5:0]	Input	signals from chip input to portb11 into 8 to 1 mux [5:0]
gpio_portb12_funcout[5:0]	Input	signals from chip input to portb12 into 8 to 1 mux [5:0]
gpio_portb13_funcout[5:0]	Input	signals from chip input to portb13 into 8 to 1 mux [5:0]
gpio_portb14_funcout[5:0]	Input	signals from chip input to portb14 into 8 to 1 mux [5:0]
gpio_portb15_funcout[5:0]	Input	signals from chip input to portb15 into 8 to 1 mux [5:0]
gpio_portc0_funcout[5:0]	Input	signals from chip input to portc0 into 8 to 1 mux [5:0]
gpio_portc1_funcout[5:0]	Input	signals from chip input to portc1 into 8 to 1 mux [5:0]

Table 23-28. GPIO Module Pin List

Pin Name	Direction	Description
gpio_portc2_funcout[5:0]	Input	signals from chip input to portc2 into 8 to 1 mux [5:0]
gpio_portc3_funcout[5:0]	Input	signals from chip input to portc3 into 8 to 1 mux [5:0]
gpio_portc4_funcout[5:0]	Input	signals from chip input to portc4 into 8 to 1 mux [5:0]
gpio_portc5_funcout[5:0]	Input	signals from chip input to portc5 into 8 to 1 mux [5:0]
gpio_portc6_funcout[5:0]	Input	signals from chip input to portc6 into 8 to 1 mux [5:0]
gpio_portc7_funcout[5:0]	Input	signals from chip input to portc7 into 8 to 1 mux [5:0]
gpio_portc8_funcout[5:0]	Input	signals from chip input to portc8 into 8 to 1 mux [5:0]
gpio_portc9_funcout[5:0]	Input	signals from chip input to portc9 into 8 to 1 mux [5:0]
gpio_portc10_funcout[5:0]	Input	signals from chip input to portc10 into 8 to 1 mux [5:0]
gpio_portc11_funcout[5:0]	Input	signals from chip input to portc11 into 8 to 1 mux [5:0]
gpio_portc12_funcout[5:0]	Input	signals from chip input to portc12 into 8 to 1 mux [5:0]
gpio_portc13_funcout[5:0]	Input	signals from chip input to portc13 into 8 to 1 mux [5:0]
gpio_portc14_funcout[5:0]	Input	signals from chip input to portc14 into 8 to 1 mux [5:0]
gpio_portc15_funcout[5:0]	Input	signals from chip input to portc15 into 8 to 1 mux [5:0]
gpio_portd0_funcout[5:0]	Input	signals from chip input to portd0 into 8 to 1 mux [5:0]
gpio_portd1_funcout[5:0]	Input	signals from chip input to portd1 into 8 to 1 mux [5:0]
gpio_portd2_funcout[5:0]	Input	signals from chip input to portd2 into 8 to 1 mux [5:0]
gpio_portd3_funcout[5:0]	Input	signals from chip input to portd3 into 8 to 1 mux [5:0]
gpio_portd4_funcout[5:0]	Input	signals from chip input to portd4 into 8 to 1 mux [5:0]
gpio_portd5_funcout[5:0]	Input	signals from chip input to portd5 into 8 to 1 mux [5:0]
gpio_portd6_funcout[5:0]	Input	signals from chip input to portd6 into 8 to 1 mux [5:0]
gpio_portd7_funcout[5:0]	Input	signals from chip input to portd7 into 8 to 1 mux [5:0]
gpio_portd8_funcout[5:0]	Input	signals from chip input to portd8 into 8 to 1 mux [5:0]
gpio_portd9_funcout[5:0]	Input	signals from chip input to portd9 into 8 to 1 mux [5:0]
gpio_portd10_funcout[5:0]	Input	signals from chip input to portd10 into 8 to 1 mux [5:0]
gpio_portd11_funcout[5:0]	Input	signals from chip input to portd11 into 8 to 1 mux [5:0]
gpio_portd12_funcout[5:0]	Input	signals from chip input to portd12 into 8 to 1 mux [5:0]
gpio_portd13_funcout[5:0]	Input	signals from chip input to portd13 into 8 to 1 mux [5:0]
gpio_portd14_funcout[5:0]	Input	signals from chip input to portd14 into 8 to 1 mux [5:0]

Table 23-28. GPIO Module Pin List

Pin Name	Direction	Description
gpio_portd15_funcout[5:0]	Input	signals from chip input to portd15 into 8 to 1 mux [5:0]
gpio_porte0_funcout[5:0]	Input	signals from chip input to porte0 into 8 to 1 mux [5:0]
gpio_porte1_funcout[5:0]	Input	signals from chip input to porte1 into 8 to 1 mux [5:0]
gpio_porte2_funcout[5:0]	Input	signals from chip input to porte2 into 8 to 1 mux [5:0]
gpio_porte3_funcout[5:0]	Input	signals from chip input to porte3 into 8 to 1 mux [5:0]
gpio_porte4_funcout[5:0]	Input	signals from chip input to porte4 into 8 to 1 mux [5:0]
gpio_porte5_funcout[5:0]	Input	signals from chip input to porte5 into 8 to 1 mux [5:0]
gpio_porte6_funcout[5:0]	Input	signals from chip input to porte6 into 8 to 1 mux [5:0]
gpio_porte7_funcout[5:0]	Input	signals from chip input to porte7 into 8 to 1 mux [5:0]
gpio_porte8_funcout[5:0]	Input	signals from chip input to porte8 into 8 to 1 mux [5:0]
gpio_porte9_funcout[5:0]	Input	signals from chip input to porte9 into 8 to 1 mux [5:0]
gpio_porte10_funcout[5:0]	Input	signals from chip input to porte10 into 8 to 1 mux [5:0]
gpio_porte11_funcout[5:0]	Input	signals from chip input to porte11 into 8 to 1 mux [5:0]
gpio_porte12_funcout[5:0]	Input	signals from chip input to porte12 into 8 to 1 mux [5:0]
gpio_porte13_funcout[5:0]	Input	signals from chip input to porte13 into 8 to 1 mux [5:0]
gpio_porte14_funcout[5:0]	Input	signals from chip input to porte14 into 8 to 1 mux [5:0]
gpio_porte15_funcout[5:0]	Input	signals from chip input to porte15 into 8 to 1 mux [5:0]
bgrdy	Input	Bandgap ready signal
bbp_stdb_oe	Input	bbp_stdb output enable
interrupts[63:0]	Input	64 AITC interrupts
column0_ode	Input	Keypad column0 open drain enable
column0_oe	Input	Keypad column0 output enable
column1_ode	Input	Keypad column1 open drain enable
column1_oe	Input	Keypad column1 output enable
sap_scka_oe	Input	sap_scka output enable
sap_stda_oe	Input	sap_stda output enable
sap_sc2a_oe	Input	sap_sc2a output enable
dtimer_ie2	Input	dtimer_fiod2 input enable
dtimer_ie1	Input	dtimer_fiod1 input enable

Table 23-28. GPIO Module Pin List

Pin Name	Direction	Description
dtimer_ie0	Input	dtimer_fiod0 input enable
tcm_16bit_mode_en	Input	TCM memtest enable
tcm_anatest_en	Input	TCM anatest mode enable
tcm_bus_master_en	Input	TCM bus master mode enable
tcm_dsp_master_mode	Input	TCM dsp master mode enable
tcm_nsertst	Input	TCM dsp serial test mode enable
ipt_test	Input	TCM test input
cko_od	Input	CCM CKOD output enable
ckoh_od	input	CCM CKOH output enable
wdog_oe	input	watchdog output enable
ipt_scan_mode	Input	gclobal scan mode signal
ipt_clk_se	Input	scan_enable signal from the TCM
spmbif_scan_tristate_enable	Input	spmbif tristate enable signal
dvocod_tst_oe	Input	Voice codec output enable signal
dvocod_tst_ie	Input	voice codec input enable signal
gpio_bgref_en	Output	bandgap reference enable
gpio_dsp_irq_b	Output	logical “or” of DSP interrupts from the gpio ports A, B, C, D, E
gpio_mcu_irq_b[4:0]	Output	MCU interrupts from gpio ports A, B, C, D, E respectively
gpio_porta_dataout[15:0]	Output	GPIO PortA output data to the pads
gpio_portb_dataout[15:0]	Output	GPIO PortB output data to the pads
gpio_portc_dataout[15:0]	Output	GPIO PortC output data to the pads
gpio_portd_dataout[15:0]	Output	GPIO PortD output data to the pads
gpio_porte_dataout[15:0]	Output	GPIO PortE output data to the pads
gpio_porta_dir[15:0]	Output	GPIO Port A directions to the pads
gpio_portb_dir[15:0]	Output	GPIO Port B directions to the pads
gpio_portc_dir[15:0]	Output	GPIO Port Cdirections to the pads
gpio_portd_dir[15:0]	Output	GPIO Port Ddirections to the pads
gpio_porte_dir[15:0]	Output	GPIO Port E directions to the pads
gpio_pintout[4:0]	Output	GPIO external interrupts

Table 23-28. GPIO Module Pin List

Pin Name	Direction	Description
gpio_porta0_altin[3:0]	Output	GPIO Porta0 alternate inputs to the peripherals
gpio_porta1_altin[3:0]	Output	GPIO Porta1 alternate inputs to the peripherals
gpio_porta2_altin[3:0]	Output	GPIO Porta2 alternate inputs to the peripherals
gpio_porta3_altin[3:0]	Output	GPIO Porta3 alternate inputs to the peripherals
gpio_porta4_altin[3:0]	Output	GPIO Porta4 alternate inputs to the peripherals
gpio_porta5_altin[3:0]	Output	GPIO Porta5 alternate inputs to the peripherals
gpio_porta6_altin[3:0]	Output	GPIO Porta6 alternate inputs to the peripherals
gpio_porta7_altin[3:0]	Output	GPIO Porta7 alternate inputs to the peripherals
gpio_porta8_altin[3:0]	Output	GPIO Porta8 alternate inputs to the peripherals
gpio_porta9_altin[3:0]	Output	GPIO Porta9 alternate inputs to the peripherals
gpio_porta10_altin[3:0]	Output	GPIO Porta10 alternate inputs to the peripherals
gpio_porta11_altin[3:0]	Output	GPIO Porta11 alternate inputs to the peripherals
gpio_porta12_altin[3:0]	Output	GPIO Porta12 alternate inputs to the peripherals
gpio_porta13_altin[3:0]	Output	GPIO Porta13 alternate inputs to the peripherals
gpio_porta14_altin[3:0]	Output	GPIO Porta14 alternate inputs to the peripherals
gpio_porta15_altin[3:0]	Output	GPIO Porta15 alternate inputs to the peripherals
gpio_portb0_altin[3:0]	Output	GPIO Portb0 alternate inputs to the peripherals
gpio_portb1_altin[3:0]	Output	GPIO Portb1 alternate inputs to the peripherals
gpio_portb2_altin[3:0]	Output	GPIO Portb2 alternate inputs to the peripherals
gpio_portb3_altin[3:0]	Output	GPIO Portb3 alternate inputs to the peripherals
gpio_portb4_altin[3:0]	Output	GPIO Portb4 alternate inputs to the peripherals
gpio_portb5_altin[3:0]	Output	GPIO Portb5 alternate inputs to the peripherals
gpio_portb6_altin[3:0]	Output	GPIO Portb6 alternate inputs to the peripherals
gpio_portb7_altin[3:0]	Output	GPIO Portb7 alternate inputs to the peripherals
gpio_portb8_altin[3:0]	Output	GPIO Portb8alternate inputs to the peripherals
gpio_portb9_altin[3:0]	Output	GPIO Portb9 alternate inputs to the peripherals
gpio_portb10_altin[3:0]	Output	GPIO Portb10 alternate inputs to the peripherals
gpio_portb11_altin[3:0]	Output	GPIO Portb11 alternate inputs to the peripherals
gpio_portb12_altin[3:0]	Output	GPIO Portb12 alternate inputs to the peripherals

Table 23-28. GPIO Module Pin List

Pin Name	Direction	Description
gpio_portb13_altin[3:0]	Output	GPIO Portb13 alternate inputs to the peripherals
gpio_portb14_altin[3:0]	Output	GPIO Portb14 alternate inputs to the peripherals
gpio_portb15_altin[3:0]	Output	GPIO Portb15 alternate inputs to the peripherals
gpio_portc0_altin[3:0]	Output	GPIO Portc0 alternate inputs to the peripherals
gpio_portc1_altin[3:0]	Output	GPIO Portc1 alternate inputs to the peripherals
gpio_portc2_altin[3:0]	Output	GPIO Portc2 alternate inputs to the peripherals
gpio_portc3_altin[3:0]	Output	GPIO Portc3 alternate inputs to the peripherals
gpio_portc4_altin[3:0]	Output	GPIO Portc4 alternate inputs to the peripherals
gpio_portc5_altin[3:0]	Output	GPIO Portc5 alternate inputs to the peripherals
gpio_portc6_altin[3:0]	Output	GPIO Portc6 alternate inputs to the peripherals
gpio_portc7_altin[3:0]	Output	GPIO Portc7 alternate inputs to the peripherals
gpio_portc8_altin[3:0]	Output	GPIO Portc8 alternate inputs to the peripherals
gpio_portc9_altin[3:0]	Output	GPIO Portc9 alternate inputs to the peripherals
gpio_portc10_altin[3:0]	Output	GPIO Portc10 alternate inputs to the peripherals
gpio_portc11_altin[3:0]	Output	GPIO Portc11 alternate inputs to the peripherals
gpio_portc12_altin[3:0]	Output	GPIO Portc12 alternate inputs to the peripherals
gpio_portc13_altin[3:0]	Output	GPIO Portc13 alternate inputs to the peripherals
gpio_portc14_altin[3:0]	Output	GPIO Portc14 alternate inputs to the peripherals
gpio_portc15_altin[3:0]	Output	GPIO Portc15 alternate inputs to the peripherals
gpio_portd0_altin[3:0]	Output	GPIO Portd0 alternate inputs to the peripherals
gpio_portd1_altin[3:0]	Output	GPIO Portd1 alternate inputs to the peripherals
gpio_portd2_altin[3:0]	Output	GPIO Portd2 alternate inputs to the peripherals
gpio_portd3_altin[3:0]	Output	GPIO Portd3 alternate inputs to the peripherals
gpio_portd4_altin[3:0]	Output	GPIO Portd4 alternate inputs to the peripherals
gpio_portd5_altin[3:0]	Output	GPIO Portd5 alternate inputs to the peripherals
gpio_portd6_altin[3:0]	Output	GPIO Portd6 alternate inputs to the peripherals
gpio_portd7_altin[3:0]	Output	GPIO Portd7 alternate inputs to the peripherals
gpio_portd8_altin[3:0]	Output	GPIO Portd8 alternate inputs to the peripherals
gpio_portd9_altin[3:0]	Output	GPIO Portd9 alternate inputs to the peripherals

Table 23-28. GPIO Module Pin List

Pin Name	Direction	Description
gpio_portd10_altin[3:0]	Output	GPIO Portd10 alternate inputs to the peripherals
gpio_portd11_altin[3:0]	Output	GPIO Portd11 alternate inputs to the peripherals
gpio_portd12_altin[3:0]	Output	GPIO Portd12 alternate inputs to the peripherals
gpio_portd13_altin[3:0]	Output	GPIO Portd13 alternate inputs to the peripherals
gpio_portd14_altin[3:0]	Output	GPIO Portd14 alternate inputs to the peripherals
gpio_portd15_altin[3:0]	Output	GPIO Portd15 alternate inputs to the peripherals
gpio_porte0_altin[3:0]	Output	GPIO Porte0 alternate inputs to the peripherals
gpio_porte1_altin[3:0]	Output	GPIO Porte1 alternate inputs to the peripherals
gpio_porte2_altin[3:0]	Output	GPIO Porte2 alternate inputs to the peripherals
gpio_porte3_altin[3:0]	Output	GPIO Porte3 alternate inputs to the peripherals
gpio_porte4_altin[3:0]	Output	GPIO Porte4 alternate inputs to the peripherals
gpio_porte5_altin[3:0]	Output	GPIO Porte5 alternate inputs to the peripherals
gpio_porte6_altin[3:0]	Output	GPIO Porte6 alternate inputs to the peripherals
gpio_porte7_altin[3:0]	Output	GPIO Porte7 alternate inputs to the peripherals
gpio_porte8_altin[3:0]	Output	GPIO Porte8 alternate inputs to the peripherals
gpio_porte9_altin[3:0]	Output	GPIO Porte9 alternate inputs to the peripherals
gpio_porte10_altin[3:0]	Output	GPIO Porte10 alternate inputs to the peripherals
gpio_porte11_altin[3:0]	Output	GPIO Porte11 alternate inputs to the peripherals
gpio_porte12_altin[3:0]	Output	GPIO Porte12 alternate inputs to the peripherals
gpio_porte13_altin[3:0]	Output	GPIO Porte13 alternate inputs to the peripherals
gpio_porte14_altin[3:0]	Output	GPIO Porte14 alternate inputs to the peripherals
gpio_porte15_altin[3:0]	Output	GPIO Porte15 alternate inputs to the peripherals
owire_oe	input	OWIRE output enable
gpio_porta2_ode	Output	Open drain enable for the port to pad
gpio_portb10_ode	Output	Open drain enable for the port to pad
gpio_portc4_ode	Output	Open drain enable for the port to pad
gpio_portc3_ode	Output	Open drain enable for the port to pad
tcm_mbist_prod_mode	Input	MCU production mode enable
tcm_mbist_bitmap_mode	Input	MCU bit mapping mode enable

Table 23-28. GPIO Module Pin List

Pin Name	Direction	Description
sjc_sio_acs	Input	DSP bist bypass mode enable
sjc_gpio_prod_cnfg	Input	DSP production mode enable
sjc_gpio_bit_map_cnfg	Input	DSP bit mapping mde enable
sab_bmap_compr[23:0]	Input	DSP Bist inputs to GPIO
sab_bmap_addr[9:0]	Input	DSP Bist inputs to GPIO
gpio_sap_scka_pri	Output	Output from Priority logic to SAP
gpio_sap_sc2a_pri	Output	Output from Priority logic to SAP
gpio_sap_srda_pri	Output	Output from Priority logic to SAP
gpio_int_int4_pri	Output	Output from Priority logic to INT
gpio_int_int5_pri	Output	Output from Priority logic to INT
gpio_uart1_dtr1_pri	Output	Output from Priority logic to UART
gpio_uart1_rxd1_pri	Output	Output from Priority logic to UART
gpio_uart1_rts1_pri	Output	Output from Priority logic to UART
gpio_mqspi_misob_pri	Output	Output from Priority logic to MQSPI

23.20 Appendix

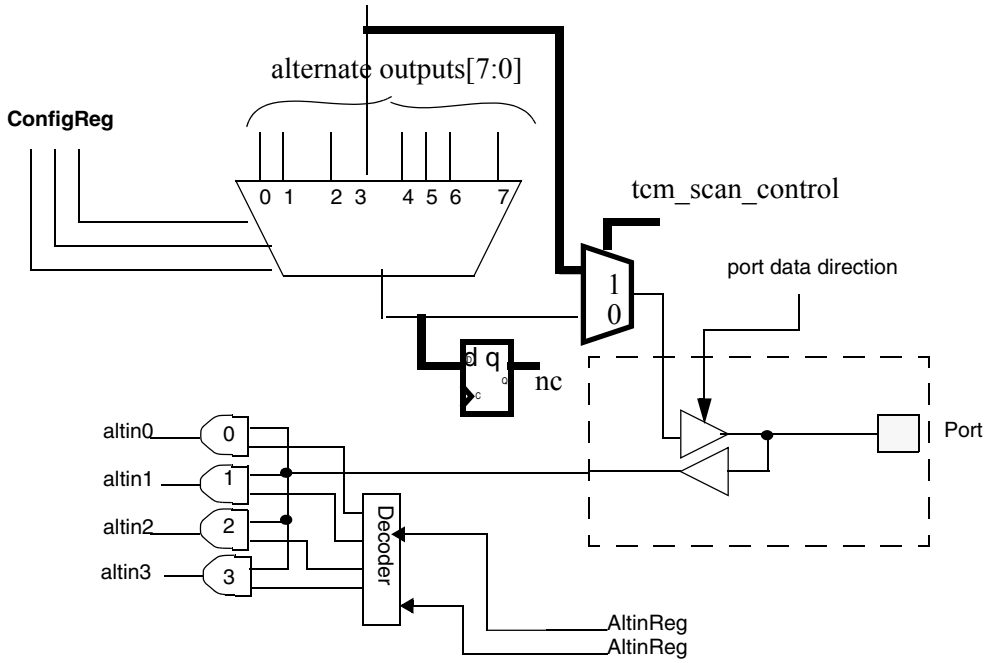


Figure 23-62. Configuring port to be used as scan_output at gpio_portx_funcout[1]

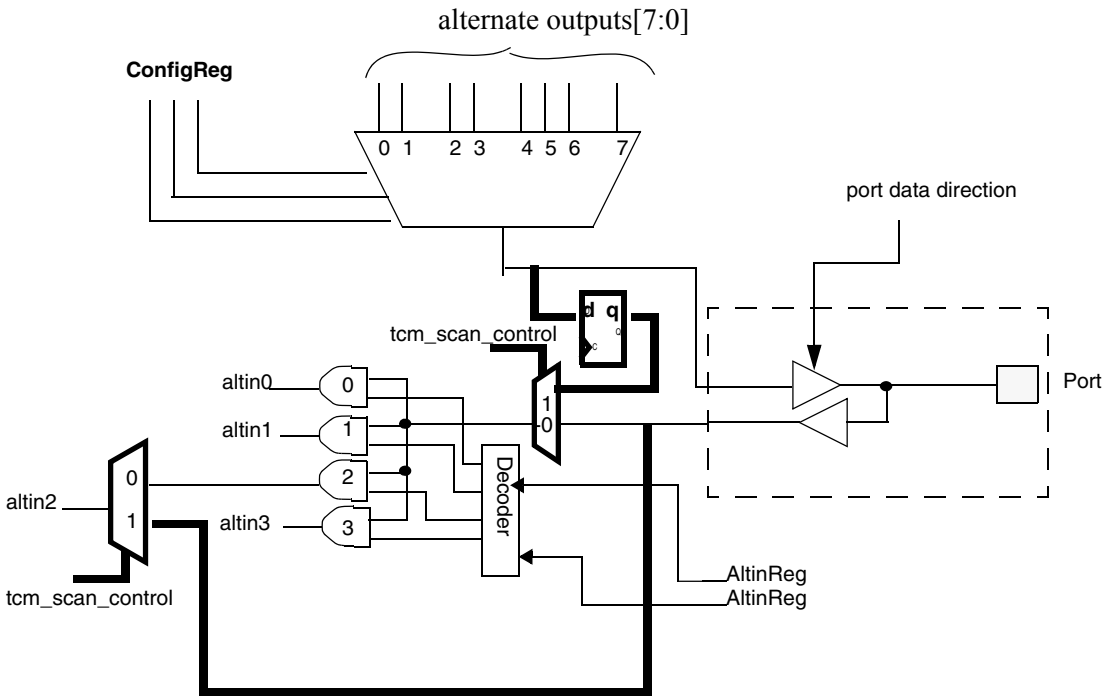


Figure 23-63. Configuring port to be used as scan input at gpio_portx_altin[2]

Chapter 24
Clock Monitor (CMON)

Revision History Table

Revision	Date	Author	Changes
0	03/25/02		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	05/07/03		Updated for LTE specification release.

24.1 System Clock Monitor (CMON, 26 MHz Typical)

24.1.1 General Description

The clock monitor module is used to detect the disappearance or a failure of the system clock for a given period of time. The monitor will assert NOCLK (high) when a “tx_26mclk_in” clock edge is not detected for 150 ns to 620 ns. When “tx_26mclk_in” is not present it is at a logic ‘0’ state as shown in the diagram in Figure 24-1. NOCLK signal is provided to the Watchdog module which will assert the Watchdog pin (WDOG_N) or the RESET_N pin. NOCLK is released within 5 uμs of valid clock edges occurring again.

The clock monitor will work for an typical input clock frequency of 26 MHz. The monitor operates on the output of the on chip crystal oscillator module, not any of the DPLL outputs.

This block will be supplied from JVDD and JVSS, which are dedicated analog supply pins, also supplied from VDD and VSS which are the digital power supply.

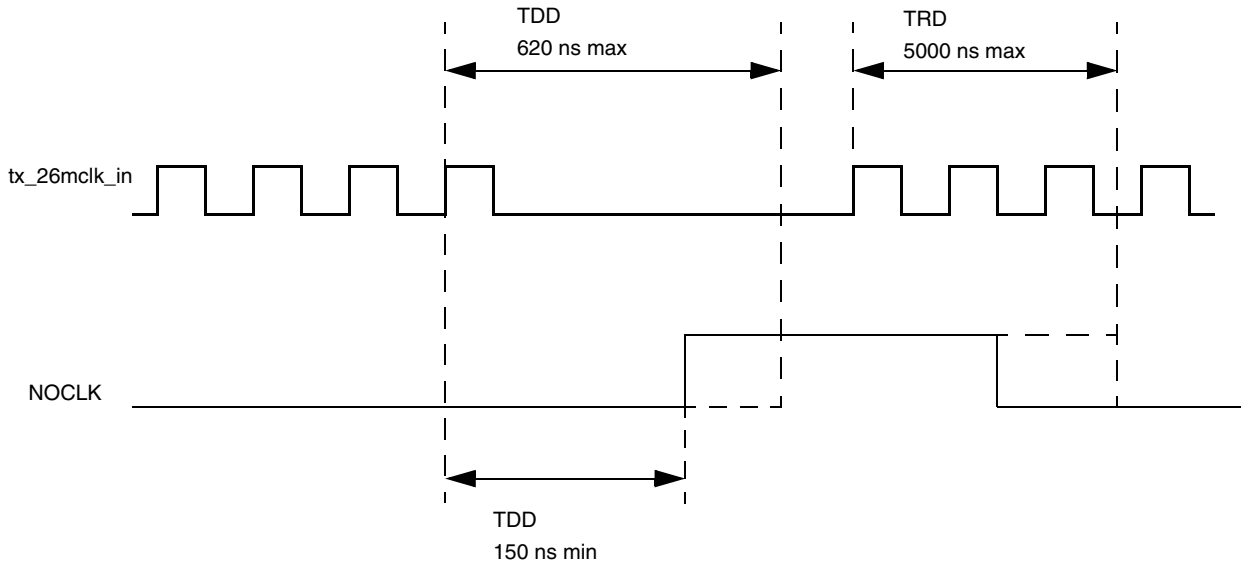


Figure 24-1. Clock Monitor Timing Diagram

Table 24-1. CMON Electrical Performance

Parameter	Min	Typ	Max	Units
Operating Voltage	2.68	2.775	2.875	V
Operating Current (@ 26 MHz operation)			100	uA
Power Down Current			3	uA
Clock Frequency	13	26	30.0	MHz
Duty Cycle	45	50	55	%
TDD: Clock disappearance detect time	0.15	.....	0.62	us
TRD: Clock appearance detect time			5	us
Startup time			50	us

24.1.2 Input and Output Description

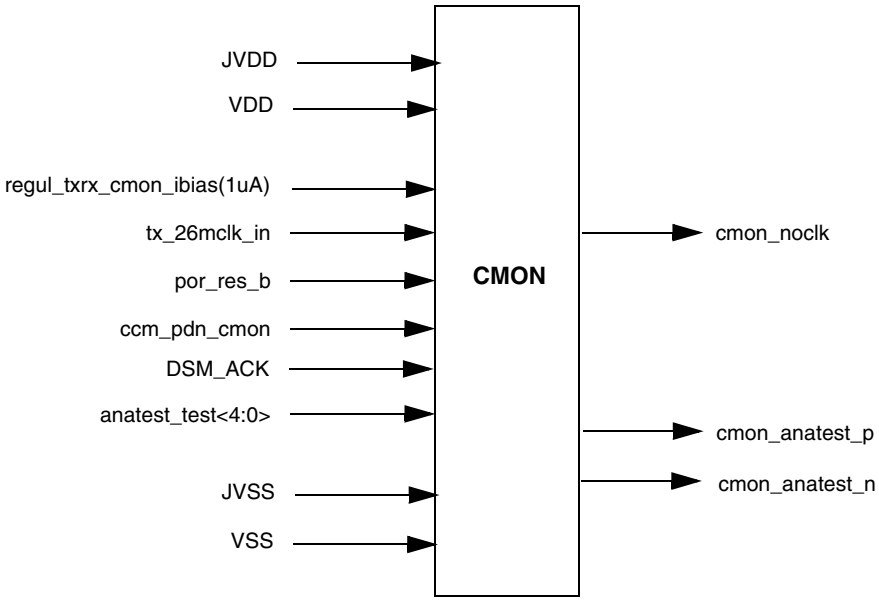


Figure 24-2. CMON Input and Output Diagram

24.2 CMON Module Pin List

Table 24-2 lists all the pins in the CMON module.

Table 24-2. CMON Module Pin List

Pin Name	Direction	Description
tx_26mclk_in	input	Input System Clock.
por_res_b	input	Power on Reset signal.
ccm_pdn_cmon	input	Power down signal. When high, powers down the clock monitor block. During power down NOCLK is kept at logic 0, indicating presence of clock. This signal is controllable by software through the Clock Control Module (CCM) and is inactive on reset.
dsm_ack	input	DeepSleepMode is used to disable the NOCLK generation when Neptune went into Deep Sleep Mode.
cmon_noclk	output	Signal to Watchdog block. Asserted high when rising clock edge has not been observed for 150 ns to 620 nµs.
cmon_anatest_p	output	Outputs for engineering debug only.
cmon_anatest_n	output	Outputs for engineering debug only.
anatest_test_x[4:0]	input	anatest control bits
regul_txrx_cmon_ibias	input	Bias current (1uA)
jvdd	input	Analog vdd (2.775 V)

Table 24-2. CMON Module Pin List

Pin Name	Direction	Description
jvss	input	Analog ground
vdd	input	Digital vdd (1.575V)
vss	input	Digital ground.

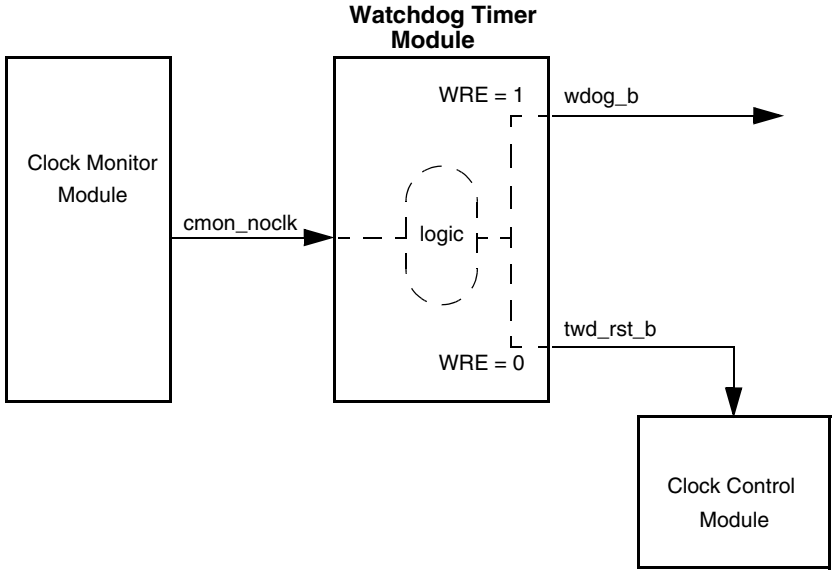


Figure 24-3. Clock Monitor and Watchdog Connection Diagram

24.2.1 Test Modes

Testing modes will be determined and defined during the design. Five(5) test bits need to be reserved from TCM register for the CMON block.

Chapter 25
Voice Codec (VOCOD)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	16/10/02	Shannon Osgood	Updated per DDTS# DSPhl15331 and DSPhl15334. Changed LTS CODEC DAC total distortion spec to 55dB min at +2dBm0 and 52dB min at 0dBm0 and CODEC DAC differential offset from 100mV to 40mV, 30mV at room temperature.
1	16/10/02	Paul McSherry	Rx_acq counter added and interpolation ratios and gain compensation has been made programmable. Also the MCU BIST production test capability has been made more robust.
1.1	31 Oct 2002	Scott King	Clarified differences between Neptune LTS and Neptune LTE/ULS
1.2	12 May 2003	Paul Sherry	Updated per DSPhl16250 and initial release for LTE specification.

25.1 Introduction

Voice codec consists of voice coding and voice decoding. The voice coding function takes a human speech signal in voltage format and converts it to 8 kHz digital signal. The voice decoding function takes 8 kilo-samples per second digital signal then converts it to an audible voice signal in voltage format. Figure 25-1 shows the top level block diagram of the Codec functional partitioning. The ADC and DAC converters are implemented with Sigma-Delta ($\Sigma\Delta$) over sampled modulators.

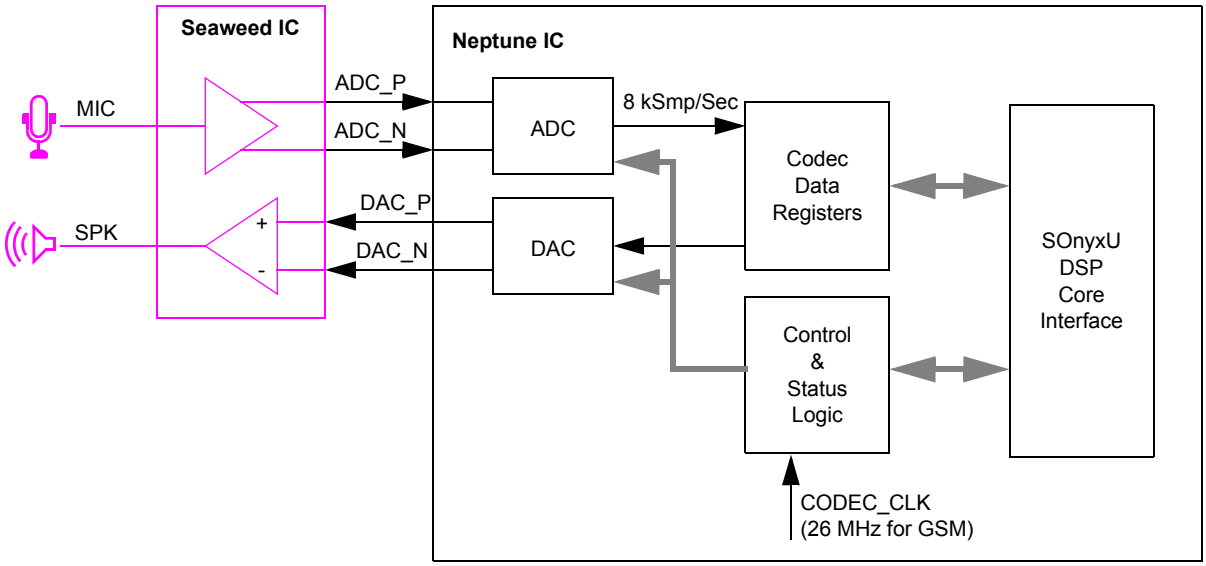


Figure 25-1. Voice Codec Block Diagram

The oversampled voice codec A/D path gets voice input from a pair of differential inputs ADC_P and ADC_N. The input signal is filtered by an analog LPF to improve the immunity to input spikes. The ΣΔ A/D converter's output is filtered to remove the quantization noise and outband signals components and decimated down to twice the Nyquist rate—that is, 16 kHz, by a comb digital filter. In the next stage the signal is gain aligned, low-pass filtered by an IIR filter and decimated by two. The sixth order, low-pass IIR filter is also compensates for passband droop introduced by the comb filter. The 8 kHz signal is optionally high-pass filtered by a second order, IIR filter and stored in the VCRX register from where it is read by the DSP software through a standard PMB interface.

The oversampling voice codec D/A path consists of an optional high-pass, second order, IIR filter, interpolation by two, sixth order IIR low-pass filter, gain alignment, interpolating digital comb filter, a ΣΔ D/A converter, an analog smoothing filter and a differential output buffer to drive the speakers. The low-pass IIR filter also compensates for passband droop introduced by the comb filter. The DSP feeds the VCTX register through the PMB interface. The comb filter interpolates the signal from twice the Nyquist rate, up to the oversampled ΣΔ frequency.

The voice codec design supports a number of input clock frequencies compatible with cellular standards. The A/D and D/A oversampling frequency is determined by the input clock (equal to a quarter of the input clock frequencies).

The codec is a peripheral of the DSP core. The codec interfaces (both data and control) to the DSP core through the SPMB interface.

In DAI test mode, software should connect the Codec to the DAI I/Os through DSP core and SAP.

25.2 Voice Codec Pin Description

The voice codec module is separated into two parts, the digital part and the analog part. The two parts are located in separate locations in the chip floor plan. The analog part contains three sub modules that are physically separate. The physical separation will enable reuse of an individual sub module in the future. The sub modules are listed in Table 25-1 :

Table 25-1. Voice Codec Sub-module list

Block Name	Sub-module Name	Description
AVOCOD	AVOCOD_REF	Generates reference power supplies for analog modules
	AVOCOD_ADC	Contains all the circuitry of the analog ADC Design
	AVOCOD_DAC	Contains all the circuitry of the analog DAC Design
DVOCOD		Contains the interface to the DSP bus (SPMB) and all the digital design including the digital filters and the sigma-delta modulators.

Figure 25-2 shows the connections between the four sub modules and the external and internal signals connected to the voice codec.

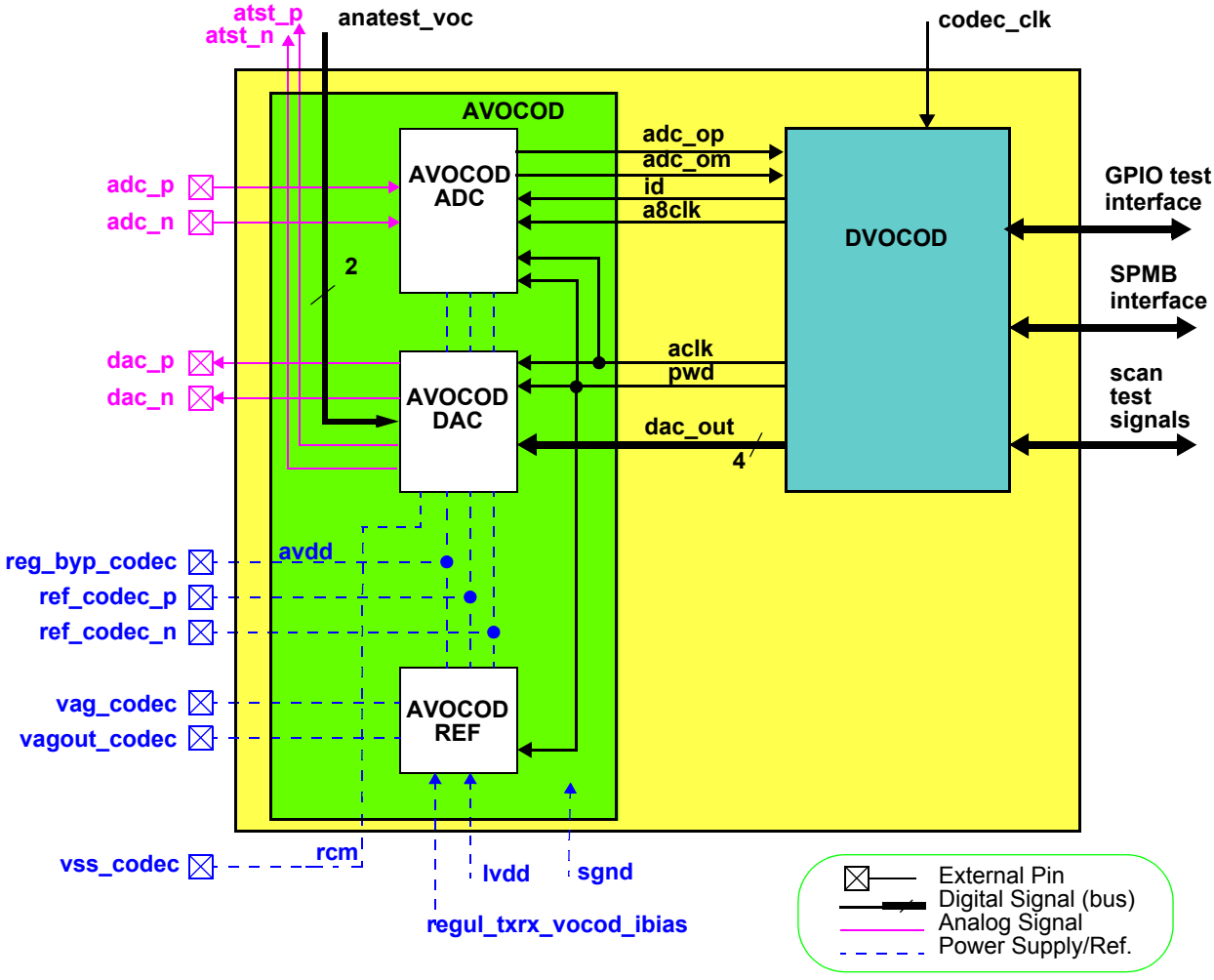


Figure 25-2. Voice Codec Module Pin and Interconnection Diagram[1]

25.2.1 Microphone Inputs (ADC_P and ADC_N)

ADC_P and ADC_N are a pair of differential analog inputs to the A/D part of the voice codec.

25.2.2 Speaker Outputs (DAC_P and DAC_N)

DAC_P and DAC_N are the differential analog outputs of the D/A part of the voice codec. When the voice codec is disabled, they are kept internally at analog GND potential.

25.2.3 Power Supply and Reference Signals

25.2.3.1 AVDD (REG_BYP_CODEC)

VOCOD analog supply voltage. Output of codec regulator for bypassing. Connected to external capacitor to the VSS_CODEC. This pin is connected to REG_BYP_CODEC pad.

25.2.3.2 VAG_CODEC

Input analog common mode reference for analog DAC part of voice codec from Seaweed.

25.2.3.3 VAG_OUT_CODEC

Buffered version of vag_codec. Connected to 0.1 µF bypass capacitor on the board to the VSS_CODEC.

25.2.3.4 REF_CODEC_P and REF_CODEC_N

Differential reference voltage signals. Each one connected to 0.1 µF bypass capacitor on the board to the VSS_CODEC.

Note: It is important to connect this capacitors to the same point on VSS_CODEC trace.

25.2.3.5 RCM (VSS_CODEC)

DAC analog ground reference. Should be connected at a single point to VSS_CODEC pad.

25.2.3.6 LVDD

Voltage reference signal connected to LVDD pad. The LVDD signal is common for number of other analog modules such as RX. LVDD is set to 1.575 Volt by Seaweed.

25.2.3.7 REGUL_TXRX_VOCOD_IBIAS

Current reference produced by txrxp_regul.

1. Note that in the RTL design all output signals names are prefixed with “dvocod_” or “avocod_” according to their source.

25.2.3.8 SGND

GND reference signal for semi-analog devices (to support pwell isolation). Should be connected to low noise point on global GND.

25.2.4 Analog Test signals

25.2.4.1 ATST_P and ATST_N

There is one differential output for the analog testing, generated by the DAC module. The signals are connect to the to the analog test module ANATEST. The ATST_P and ATST_N are connected to the pre amplified output of the DAC.

25.2.4.2 ANATEST_VOC[1:0]

There are two digital inputs ANATEST_VOC[1:0] from the ANATEST module connected to the DAC. The two inputs control the operation in the analog tests mode. The analog test is enabled when ANATEST_VOC[0] is set to high. For normal operation both ANATEST_VOC[1:0] should be set to low. (Note that in the current design ANATEST_VOC[1] do not have any function and doesn't influence the ANATEST output.)

25.2.5 Clock signal - CODEC_CLK

Clock signal that is used as a basic clock for all module operations. The voice codec supports three clock frequencies 26 MHz, 19.44 MHz and 16.8 MHz.

25.2.6 DSP Interface

The voice codec interfaces to the DSP via SPMB interface.

25.2.7 DVOCOD Module Pin List

Table 25-2 summarizes the pinout of the DVOCOD block, including all signals.

Table 25-2. DVOCOD Module Pin List

Pin Name	Direction	Description	Signal name	Envelop Pin
General Signals				
codec_clk	I	Voice Codec clock (26 MHz) from CCM module		
AVOCOD Interface Signals				
avocod_adc_op	I	Input from ADC analog Sigma Delta 1		
avocod_adc_om	I	Input from ADC analog Sigma Delta 2		
dvocod_dac_out[3:0]	O	Output to analog DAC Current Source Array		
dvocod_pwd	O	power down signal for analog parts		

Table 25-2. DVOCOD Module Pin List (Continued)

Pin Name	Direction	Description	Signal name	Envelop Pin
dvocod_id	O	Output dither signal to analog ADC sigma delta		
dvocod_aclk	O	clock for the Analog part (ADC/DAC)		
dvocod_a8clk	O	Clock the Analog Part (1/8 frequency)		
SPMBIF Signals				
mcab_wr[6:0]	I	core write sector address bits		
mcab_rd[6:0]	I	core read sector address bits		
mcsel_wr	I	X or Y I/O write select		
mcsel_rd	I	X or Y I/O read select		
gdb_wr[15:0]	I	write data bus		
gdb_rd [15:0]	O	Core Global Read Data Bus		
mcdis0	I	T0 clock disable		
mcdis1	I	T1 clock disable		
pivsse	I	Interrupt vector source select. Received from DSIH (DSP Special Interrupt Handler) dsih_pivsse_vocod pin.		
gclkw	I	raw clock for new custom module		
bcstop	I	Core executing the STOP instruction indicator		
mirq_b	O	Peripheral module interrupt request. Active low. Connected to DSIH (DSP Special Interrupt Handler) dvocod_mirq_b pin.		
vab_rd[7:1]	O	Interrupt Vector Bus 3 Interrupts of the VOCOD: VCI @VBA:$34, VROE @VBA:$36 VTUE @VBA:$38		
pires	I	Peripheral Module's software reset		
res	I	Peripheral Module's hardware reset		
bank_base_ref[6:3]	I	Peripheral sector location for reference - VOCOD register base address x:$FFE0, corresponds to base address $B. Connect to via {vcc, vcc, gnd, gnd}.		
vab_base_add[7:4]	I	Interrupt base vectors address VOCOD interrupts addr (VBA:$34, VBA:$36, VBA:$38) corresponds to base address $3. Connect to via {gnd, vcc, vcc}.		
Scan Chain Test Signals				
ipt_scan_mode	I	scan mode		
ipt_scan_enable	I	global scan enable signal		
ipt_dvocod_si[2:0]	I	scan data inputs		
ipt_clk_se	I	gated clock scan enable		

Table 25-2. DVOCOD Module Pin List (Continued)

Pin Name	Direction	Description	Signal name	Envelop Pin
spmbif_scan_tristate_enable	I	control signal for tri-state buses in scan mode		
ipt_dvocod_so[2:0]	O	scan data outputs		
Rx_Acq Counter Signals				
a2di_sc1b	I	serial receive framing signal		
a2di_rx_acq	I	rx_acq signal from rxcproc module		
Bist Test Mode Signals				
tcm_dvocod_tst_en	I	dvocod mcu bist parallel mode test enable from tcm		
gpio_dvocod_tst_dir	I	controls the direction of the test data bus from pad		
gpio_codec_bist_clk	I	codec clock in bist mode from pad		
gpio_dvocod_tst_data [15:0]	IO	dvocod bist mode test data bus.		
dvocod_tst_oe	O	test bus output enable		
dvocod_tst_ie	O	test bus input enable		
dvocod_tst_strb	O	test bus strobe signal - indicates that input data is sampled by the DAC and the output data from the ADC is ready		

25.2.8 AVOCOD Module Pin List

Table 25-3 summarizes the pinout of the AVOCOD block, including all signals.

Table 25-3. AVOCOD Module Pin List

Pin Name	Direction	Description	Signal name	Envelop Pin
General Signals				
dvocod_pwd	I	Power-Down		
dvocod_aclk	I	CODEC_CLK/4 analog clock, 50% duty cycle		
AVOCOD_ADC Signals				
dvocod_id	I	Digital input dither signal to analog ADC sigma delta		
dvocod_a8clk	I	CODEC_CLK/32 analog clock, 50% duty cycle		
avocod_adc_p	Anlg I	codec ADC analog differential input (external pad) - positive		
avocod_adc_n	Anlg I	codec ADC analog differential input (external pad) - negative		

Table 25-3. AVOCOD Module Pin List

Pin Name	Direction	Description	Signal name	Envelop Pin
avocod_adc_op	O	ADC sigma delta 1 output		
avocod_adc_om	O	ADC sigma delta 2 output		
AVOCOD_DAC Signals				
dvocod_dac_out[3:0]	I	Input to Analog DAC Current Source Array		
avocod_dac_p	Anlg O	DAC differential output (external pads)		
avocod_dac_n	Anlg O			
avocod_atst_p	Anlg O	Differential Output to Analog Test module		
avocod_atst_n	Anlg O			
anatest_voc[1:0]	I	Digital Inputs from ANATEST module (from ANATEST data reg).		
Analog Power Supply and Voltage Reference Signals				
avdd	I	VOCOD analog supply voltage. Output of codec regulator for bypassing. This pin is connected to REG_BYP_CODEC pad.		
avocod_rcm	I	DAC analog ground reference. should be connected at one point to vss_codec pad.		
avocod_ref_codec_p	O	Differential reference voltage signals. Each one connected to 0.1 μF bypass capacitor. (external pads)		
avocod_ref_codec_n	O			
avocod_lvdd	I	Voltage reference signal connected to LVDD pad. The LVDD signal is common for number of other analog modules such as RX. LVDD is set to 1.575 Volt by Seaweed.		
regul_txrx_vocod_ibias	I	current reference from txrxp_regul		
vdd	I			
agnd	I			
avocod_vag_codec	I	Input analog reference for analog part of voice codec from Seaweed		
avocod_vagout_codec	O	Buffered version of vag_codec. Connected to 0.1 μF bypass capacitor on the board to the VSS_CODEC		
sgnd	I	GND reference signal for semi-analog devices (to support pwell isolation). Should be connected to low noise point on global GND.		

25.3 Voice Codec Programming Model

The voice codec is designed as a peripheral of the DSP core. The DSP has full control of all the voice codec functions, status indicators, and the receive and transmit data registers.

25.3.1 Register Summary

The voice codec module has four registers. The registers are: a control register (VCCR), a status register (VCSR), a receive data register (VCRX), and a transmit data register (VCTX).

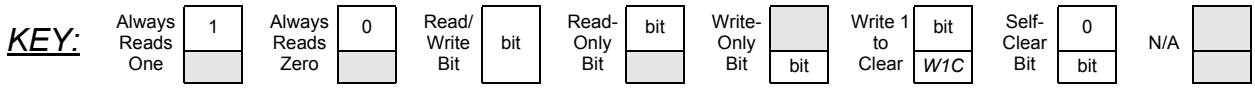


Table 25-4. Voice Codec DSP Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VCCR (X:$FFE0)	R	VCEN	VCIE	VCI XE	VCO XE	0	0	BYP DIG	VC TST	VCO HPF	VCI HPF	VC DITH	0	VC MIDI	VCLK[2:0]		
	W												VC RST				
VCSR (X:$FFE1)	R	VCRF	VCTE	VROE	VTUE	0	0	0	0	0	0	0	0	0	0	0	0
	W																
VCRX (X:$FFE2)	R	VCRX[12:0]													0	0	0
	W																
VCTX (X:$FFE3)	R	VCTX[15:0]															
	W																
VDIR_MIDI (X:$FFE4)	R	0					VDIR_MIDI[10:0]										
	W						VDIR_MIDI[10:0]										
GR_WORD_MIDI (X:$FFE5)	R	0								GR_WORD_MIDI[7:0]							
	W									GR_WORD_MIDI[7:0]							
RXACQ_COUNT (X:$FFE6)	R	RXACQ_COUNT[15:0]															
	W																

25.3.2 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various voice codec registers. The following definitions serve as a key for these figures:

- **Grey bit**: Unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: Read only. Writing this bit has no effect.
  - **w**: Write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0.
  - **1**: Will reset to a logic 1.
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset.

The voice codec DSP Control Register determines all operation modes of the module. It controls enabling/disabling of the module, enabling module interrupts, determines the operation clock frequency, and activation of various test modes.

VCCR **Voice Codec DSP Control Register** **Addr X:$FFE0**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	VCEN	VCIE	VCI XE	VCO XE			BYP DIG	VC TST	VCO HPF	VCI HPF	VC DITH	VC RST	VC MIDI	VCLK[2:0]		
TYPE:	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	slfclr	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-5. VCCR Description

Name	Description	Settings
VCEN Bit 15	**Voice Codec Enable**—A logic high enables the Codec, logic low puts the codec in battery save mode. DSP hardware reset, software reset (DSP RESET instruction) or Powerup clear VCEN bit.	VCEN = 0 → Voice codec disabled and put in power save mode. VCEN = 1 → Voice codec enabled.
VCIE Bit 14	**Voice Codec Interrupt Enable**—VCIE bit enables the voice codec interrupt (VBA:$34). Voice codec has a single interrupt for input and output paths. It indicates that the VCTX register is empty and that a new sample can be output to the DAC. It also indicates that the VCRX register contains a new input sample from the ADC. The bit can be modified during the module operation. VCIE is cleared by hardware and software reset.	VCIE = 0 → Voice codec interrupt is disabled. VCIE = 1 → Voice codec interrupt is enabled.
VCIXE Bit 13	**Voice Codec Input Exception interrupt enable**—VCIXE bit enables the voice codec input exception interrupt (VBA:$36) that occurs in an overrun situation. The interrupt indicates that a new sample is set into the VCRX register before the previous one is read out. The bit can be modified during the module operation. VCIXE is cleared by hardware and software reset.	VCIXE = 0 → Voice codec Input Exception interrupt is disabled. VCIXE = 1 → Voice codec Input Exception interrupt is enabled.
VCOXE Bit 12	**Voice Codec Output Exception interrupt enable**— VCOXE bit enables the voice codec output exception interrupt (VBA:$38) that occurs in an underrun situation. The interrupt indicates that a new sample required by the DAC but the VCTX register has not been written yet with a new data. The bit can be modified during the module operation. VCOXE is cleared by hardware and software reset.	VCOXE = 0 → Voice codec Output Exception interrupt is disabled. VCOXE = 1 → Voice codec Output Exception interrupt is enabled.
Bits 11-10	**Reserved Bits—** These bits are reserved for future expansion. They are read as zero and should be written with zero for upward compatibility.	N/A

Table 25-5. VCCR Description (Continued)

Name	Description	Settings
BYPDIG Bit 9	**Bypass Digital Filters**— Dedicated test bit. For normal operation should be written with zero. BYPDIG is cleared by hardware and software reset.	BYPDIG = 0 → Normal Operation. BYPDIG = 1→ bypass all digital filters on DAC path.
VCTST Bit 8	**Voice Codec Test** — Dedicated test bit. For normal operation should be written with zero. VCTST is cleared by hardware and software reset.	VCTST = 0 → Normal Operation. VCTST = 1→ bypass IIR filters.
VCOHPF Bit 7	**Voice Codec Output High-Pass Filter Enable**— A logic high enables the high-pass filtering at the DAC path. The bit can be changed “on the fly” even during the voice codec operation. Note that if the VCOHPF bit is changed in the presence of audio, a “pop” noise may be produced by the Codec. Power up default is 0 with HPF disabled.	VCOHPF = 0 → Voice codec HPF at DAC path disabled and put in power save mode. The signal from the VCTX register is directly bypassed to the VLPF filter. VCOHPF = 1 → Voice codec HPF at DAC path enabled.
VCIHPF Bit 6	**Voice Codec Input High-Pass Filter Enable**— A logic high enables the high-pass filtering at the ADC path. The bit can be changed “on the fly” even during the voice codec operation. Note that if the VCIHPF bit is changed in the presence of audio, a “pop” noise may be produced by the Codec. Power up default is 0 with HPF disabled.	VCIHPF = 0 → Voice codec HPF at ADC path disabled and put in power save mode. The signal from the LPF is directly bypassed to the VCRX register. VCIHPF= 1 → Voice codec HPF at ADC path enabled.
VCDITH Bit 5	**Voice Codec Output Dither Disable**— Logic low enables dithering. Dithering decorrelates the periodic modulator quantization noise of the output converter. If VCDITH is set to a logic high, dithering is disabled. Disabling dithering may be desired for data communications. Power up default is 0.	VCDITH = 0 → Voice codec output dithering is enabled. VCDITH s= 1 → Voice codec output dithering is disabled.
VCRST Bit 4	**Voice Codec Reset**— Writing ‘1’ to this bit will reset the voice codec logic. The bit is self cleared and reads as zero.	
VCMIDI Bit 3	**Voice Codec Midi Mode Control—** This bit allows the sampling rate to be made programmable. The interpolation rate and gain compensation are programmed through VDIR_MIDI and GR_WORD_MIDI. In this mode the ADC is disabled.	VCMIDI = 0 → Voice mode, interpolation rate and gain compensation determined from dclk[2:0] VCMIDI s= 1 → Midi mode, interpolation rate and gain compensation programmed through VDIR_MIDI and GR_WORD_MIDI.

Table 25-5. VCCR Description (Continued)

Name	Description	Settings
VCLK[2:0] Bits 2-0	**Voice Codec Clock Select—**Selects the CODEC clock input and output frequencies. Changing the VCLK bits when the codec is enabled (VCEN = 1) will cause a reset of the codec logic. Power up default is '000'B.	

VCLK [2:0]	CODEC_ CLK	Software Interface Rate
000	26MHz	8kHz
001	26MHz	8kHz
010	16.8MHz	8kHz
011	19.44MHz	8kHz
100	19.44MHz	8.1kHz
101	16.8MHz	8kHz
110	16.8MHz	8kHz
111	TBD	TBD

The voice codec DSP Status Register contains indicators of the voice codec data registers status. It includes indicators of transmit data empty; receive data full; and underrun/overrun situations.

VCSR **Voice Codec DSP Status Register** **Addr X:$FFE1**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	VCRF	VCTE	VROE	VTUE												
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-6. VCSR Description

Name	Description	Settings
VCRF Bit 15	**Voice Codec Receive Data Full—** read-only bit indicates that the receive data register VCRX contains data from the codec A/D section. VCRF is set when data is transferred from the A/D last filter stage output to the VCRX register. VCRF is cleared when the VCRX register is read by the DSP Core. VCRF is also cleared when the voice codec is disabled (clear VCEN bit in VCCR); writing one to VCRST bit; and by hardware reset and RESET and STOP instructions. VCRF assertion will generate a voice codec interrupt VCI, if VCIE bit in VCCR is set by the user.	VCRF = 0 → VCRX register is empty (contains old data). VCRF = 1 → receive data register VCRX contains data from the codec A/D section.

Table 25-6. VCSR Description (Continued)

Name	Description	Settings
VCTE Bit 14	**Voice Codec Transmit Data Empty—** read-only bit indicates that the transmit data register VCTX is empty and can be written by the DSP Core. VCTE is set when the data is transferred from the VCTX register to the D/A filter input. VCTE is cleared when the VCTX register is written by the Core or by writing one to VCRST bit. VCTE is set when the voice codec is disabled (clear VCEN bit in VCCR); or by hardware reset and RESET and STOP instructions. VCTE assertion will generate a voice codec interrupt VCI, if VCIE bit in VCCR is set by the user and the voice codec is enabled.	VCTE = 0 → Voice codec VCTX register is not empty yet. VCTE = 1 → Voice codec VCTX register is empty and a new data sample can be sent out.
VROE Bit 13	**Voice Codec Receive Overrun Error —** read-only flag is set when a new sample is received from the codec section while the previous received sample in VCRX register has not been read yet by the DSP Core (overrun error). In this case, the previous received sample is over-written in the VCRX register. Hardware or software reset and the STOP instruction clear VROE. VROE is also cleared by reading the VCSR (with VROE set) followed by reading VCRX or by writing one to VCRST bit. Clearing VCEN bit in the VCCR register does not affect VROE. VROE assertion will generate a voice codec receive with overrun error interrupt if VCIXE bit in VCCR is set by the user. Note: The overrun interrupt is only an exception indicator. When there is an overrun situation the user cannot count on the data in VCRX register to be correct.	VROE = 0 → No overrun errors in voice codec. VROE = 1 → Voice codec Receive Overrun Error.
VTUE Bit 12	**Voice Codec Transmit Underrun Error—** read-only flag bit is set when a sample has to be transmitted to the codec section while the DSP Core has not yet written to VCTX register (underrun error). In this case, the previous received sample is re-transmitted to the D/A section. Hardware or software reset and the STOP instruction clear VTUE. VTUE is also cleared by reading the VCSR with VTUE set followed by writing VCTX or by writing one to VCRST bit. Clearing VCE bit in the VCCA register does not affect VTUE. VTUE assertion will generate a voice codec transmit with underrun error interrupt if VCOXE bit in VCCR is set by the user. Note: The underrun interrupt is only an exception indicator. When there is an underrun situation the user cannot count on the data in VCTX register.	VTUE = 0 → No underrun errors in voice codec. VTUE = 1 → Voice codec Transmit Underrun Error.
Bits 11-0	**Reserved Bits—** These bits are reserved for future expansion. They are read as zero and should be written with zero for upward compatibility.	N/A

The voice codec Receive Data Register contains the last sample converted by the ADC.

VCRX **Voice Codec Receive Data Register** **Addr X:$FFE2**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	VCRX[12:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-7. VCRX Description

Name	Description	Settings
VCRX [12:0] Bits 15-3	**Voice Codec Receive Data Register—** The VCRX register is a 13-bit read only register that contains the last sample converted by the voice codec ADC. The VCRX register is loaded with 13-bit data from the last filter (LPF or HPF) at a rate of 8kHz (or 8.1kHz). This transfer operation sets the VCRF bit in the VCSR Status Register, and if VCIE bit in VCCR is set, a VCI interrupt will be generated. Reading VCRX clears VCRF. Writing 'one' to VCRST bit clears VCRX.	
Bits 2-0	**Reserved Bits—** These bits are reserved for future expansion. They are read as zero and should be written with zero for upward compatibility.	N/A

The voice codec Transmit Data Register is used for output of voice data samples to the codec DAC.

VCTX **Voice Codec Transmit Data Register** **Addr X:$FFE3**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	VCTX[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-8. VCTX Description

Name	Description	Settings
VCTX [15:0] Bits 15-0	**Voice Codec Transmit Data Register—** The VCTX register is a 16-bit read/write register that used by the DSP Core to write samples to the codec DAC. The new data should be written after receive of VCI interrupt (or when VCTE status bit high). Writing to the VCTX clears the VCTE bit in VCSR register. VCTX is cleared by DSP hardware and software reset, STOP operation and by writing ‘one’ to VCRST bit.	

The voice codec interpolation ratio for midi mode is stored in VDIR_MIDI.

.

VDIR_MIDI **Voice Codec Interpolation Rate Register** **Addr X:$FFE4**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
						VDIR_MIDI										
TYPE						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:						0	1	0	0	1	0	0	1	1	1	0

Table 25-9. VCRX Description

Name	Description	Settings
Bits 15-11	**Reserved Bits—** These bits are reserved for future expansion. They are read as zero and should be written with zero for upward compatibility.	N/A
VDIR_MIDI [10:0] Bits 10-0	**Voice CodecMidi Interpolation ratio register —** The VDIR_MIDI register is an 11-bit read/write register that contains the interpolation ratio for midi dac mode of operation. This interpolation ratio is selected when VCMIDI is set. VDIR_MIDI is reset by DSP hardware and software reset.	N/A

The voice codec gain compensation value for midi mode is stored in GR_WORD_MIDI.

.

GR_WORD_MIDI **Voice Codec Gain Compensation Register** **Addr X:$FFE5**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									GR_WORD_MIDI							
TYPE									rw	rw	rw	rw	rw	rw	rw	rw
RESET:									0	1	1	1	0	0	0	0

Table 25-10. VCRX Description

Name	Description	Settings
Bits 15-8	**Reserved Bits—** These bits are reserved for future expansion. They are read as zero and should be written with zero for upward compatibility.	N/A
GR_WORD_MIDI [10:0] Bits 7-0	**Voice CodecMidi Gain Compensation register —** The GR_WORD_MIDI register is an 8-bit read/write register that contains the gain compensation value for midi dac mode of operation. This gain compensation is selected when VCMIDI is set. GR_WORD_MIDI is reset by DSP hardware and software reset.	N/A

The number of gclkw cycles between the rising edge of rx_acq and the rising edge of sdfs is stored in the RXACQ_COUNT register.

.

RXACQ_COUNT **EOTD RXACQ_COUNT Register** **Addr X:$FFE6**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	RXACQ_COUNT[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-11. VCRX Description

Name	Description	Settings
RXACQ_COUNT[10:0] Bits 15-0	**RXACQ_COUNT Register—** The RXACQ_COUNT register is an 16-bit read only register that is used to count the number of gclkw cylces between the rising edge of RX_ACQ and the rising edge of SDFS. The counter starts after detecting a rising edge on RX_ACQ and stops after the first rising edge occurs on SDFS. The counter is cleared by by DSP hardware and software reset and when the Counter is read by the DSP.	N/A

25.3.3 Voice Codec Interrupts

The voice codec has three interrupt vectors.

VCI - Voice Codec Interrupt (VBA:$34). Voice codec has a single interrupt for input and output paths. It indicates that the VCTX register is empty and that a new sample can be output to the DAC. It also indicates that the VCRX register contains a new input sample from the ADC. The VCI interrupt is enabled by setting high the VCIE bit in VCCR register. VCRF and VCTE status bits are set simultaneously with the activation of the VCI interrupt. The interrupt request is cleared when the VCRX register is read and VCTX register is written (in any sequence). In midi mode the interrupt is cleared by writing to VCTX, no need to read VCRX as ADC is off.

VROE - Voice Codec Receive Overrun Error Interrupt (VBA:$36). The interrupt is activated when a new sample is received from the codec section while the previous received sample in VCRX register has not been read yet by the DSP Core. In this case, the previous received sample is over-written in the VCRX register. The interrupt is enabled when VCIXE bit in VCCR is set by the user. VROE status bit is set simultaneously with the activation of the VROE interrupt. VROE interrupt is cleared by reading the VCSR with VROE set followed by reading VCRX or by writing one to VCRST bit. Clearing VCEN bit in the VCCR register does not affect VROE. This interrupt will never be activated in midi mode.

NOTE:

The overrun interrupt is only an exception indicator. When there is an overrun situation the user cannot count on the data in VCRX register to be correct.

VTUE - Voice Codec Transmit Underrun Error Interrupt (VBA:$38). The interrupt is set when a sample has to be transmitted to the codec section while the DSP Core has not yet written to VCTX register. In this case, the previous received sample is re-transmitted to the D/A section. The interrupt is enabled when VCOXE bit in VCCR is set by the user. VTUE is cleared by reading the VCSR with VTUE set followed by writing VCTX or by writing one to VCRST bit. Clearing VCE bit in the VCCA register does not affect VTUE. VTUE status bit is set simultaneously with the activation of the VTUE interrupt.

NOTE:

The underrun interrupt is only an exception indicator. When there is an underrun situation the user cannot count on the data in VCTX register.

25.4 Voice Codec Design Description

The following sections describe in detail the processing of the signal along the ADC and DAC paths. The ADC and DAC converters are implemented with Sigma-Delta (ΣΔ) over sampled modulators. The sampling frequency of the ΣΔ converters, the Interpolation/Decimation ratios and the final voice sampling rate are determined by the codec input clock (CODEC_CLK) and VCLK[2:0] control bits according to Table 25-12.

Table 25-12. Voice Codec Input Clock Settings

VCLK[2:0]	CODEC Input clock CODEC_CLK	Sigma Delta Clock $F_{\Sigma\Delta}$	First Decimation Ratio	Intermediate frequency	Software Interface Frequency
VCLK[2:0]=0,1	26MHz	6.5MHz	406.25	16kHz	8kHz
VCLK[2:0]=2,5,6	16.8MHz	4.2MHz	262.5	16kHz	8kHz
VCLK[2:0]=3	19.44MHz	4.86MHz	303.75	16kHz	8kHz
VCLK[2:0]=4	19.44MHz	4.86MHz	300	16.2kHz	8.1kHz
VCLK[2:0]=7	TBD	TBD	TBD	TBD	TBD

25.4.1 Voice Codec ADC Operation

Figure 25-3 shows the signal path block diagram for the analog-to-digital voice converter.

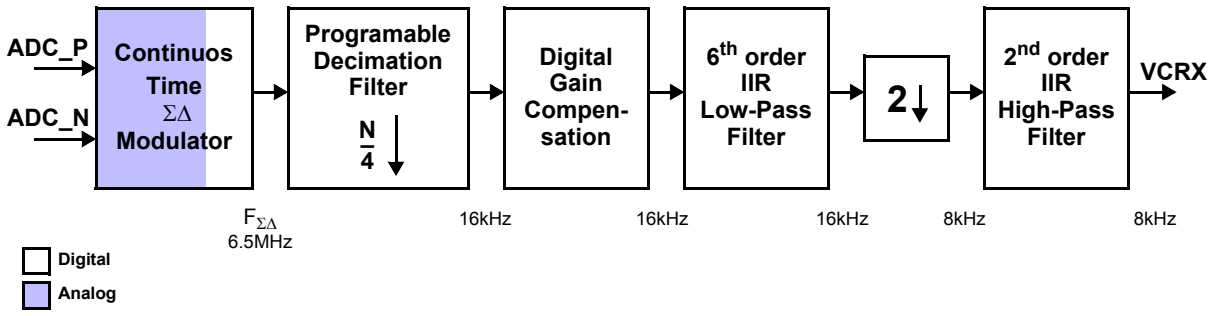


Figure 25-3. Voice Codec ADC Signal Path Block Diagram[1]

1. The clock frequencies are for VCLK=0 (GSM)

The input signal is first low pass filtered with a 200-300kHz input filter, just to prevent high-frequency overload of the ADC. In continuous-time ADC there is no significant aliasing of high-frequency input signals and therefor an anti-aliasing filtering is not needed. The analog-to-digital conversion is performed with a first order continuos time Sigma-Delta (ΣΔ) converter. The Sigma-Delta modulator performs quantization to a multibit word length with the quantization noise shaping of first order. The Sigma-Delta modulator operates at a high frequency clock ($F_{\Sigma\Delta}$). The $F_{\Sigma\Delta}$ frequency depends on the input system clock frequency, and varies between 4.2 MHz and 6.5 MHz (for GSM setup VCLK = 0, the input system clock is 26 MHz and the Sigma-Delta operates at 6.5 MHz). Next, the signal is low-pass filtered and down sampled to about 16k samples per second with a programmable filter. The filter is capable of performing decimation by a fractional ratio with a resolution of one-fourth (by a factor N/4 when N is an integer). For the GSM setup, the decimation ratio is 406.25. The decimation filter architecture is a variation of a comb filter and has a finite impulse response and a liner phase. After the decimation, the signal gain is corrected to compensate for the non-unity gain of the components along the signal path and to correlate the input signal level to the digital representation in the VCRX register. Next, the signal is low-pass filtered with a 6^{th} order, IIR filter. The filter's purpose, in addition to anti-aliasing of the following decimation by 2 stage, is to compensate for the decimation filter droop in the passband. The signal is then additionally decimated down by two, to a final sampling rate of about 8 kHz. Finally the signal is optionally high-pass filtered and is output to the VCRX register. The HPF filter operation is enabled by the VCIHPF control bit.

25.4.1.1 ADC Sigma-Delta Modulator

The analog part of the converter is a first order, 1-bit ΣΔ modulator. The ΣΔ operates with $F_{\Sigma\Delta}$ clock ($F_{\Sigma\Delta}$ frequency is 6.5 MHz for GSM setup). Table 25-12 specifies the $F_{\Sigma\Delta}$ frequency for various setups. The analog ΣΔ modulator uses fully differential circuitry to increase the robustness to noise and spikes at the comparator input.

Low-order sigma-delta modulators are known to generate tones which have especially high level for DC inputs and low-level AC inputs. The tones are sinusoidal interferences. Frequencies and magnitudes of the tones are depended on an input analog signal. Dithering methods are used to reduce the tones level by spreading the tones energy in a wide band. The aim of the digital part of the ADC is to increase the signal-to-noise ratio (SNR) of the converter by means of dithering.

The digital part of the ADC receives two, one-bit outputs of the analog part and returns a dither feedback signal to the analog part.

The functional diagram of the ADC is shown in Figure 25-4. The analog part of the converter consists of a differential first order, 1-bit ΣΔ modulator. The modulator output is fed to the digital ΣΔ modulator as a differential signal consisted of two, one bit outputs. The order of the digital modulator is two. The first integrator of the digital modulator operates at the ADC sampling rate. The output of the first integrator is decimated by eight. And the second integrator operates only at one-eighth of the ADC sampling frequency. The decimation is required because the relatively long transition time of a feedback signal in the analog part.

The negative feedback signal is generated by a digital one-bit comparator. The comparator input signal is a sum of the second integrator output and a multilevel pseudorandom sequence.

On the ADC output, compensation of the dither noise is performed. The digital feedback signal is added back to the analog modulators outputs with the same level as in the analog modulators inputs.

The designed ADC has the following advantages. When an input analog signal has a low magnitude, it is suppressed in the digital modulator by the dither noise. In this case, the effect of the negative feedback is not significant and the ADC operates like a conventional sigma-delta modulator with a spectral shaped dither being fed to the ADC input.

When a high level input signal is applied, the feedback returns a part of the signal. As a result of addition of the analog part and digital part outputs, the maximal signal which can be converted without significant

distortions is close to the sum of magnitudes of the analog and digital feedback signals. Thus, dithering in this ADC doesn't decrease the maximal admissible magnitude of the signal in contrast to other ADCs with dithering.

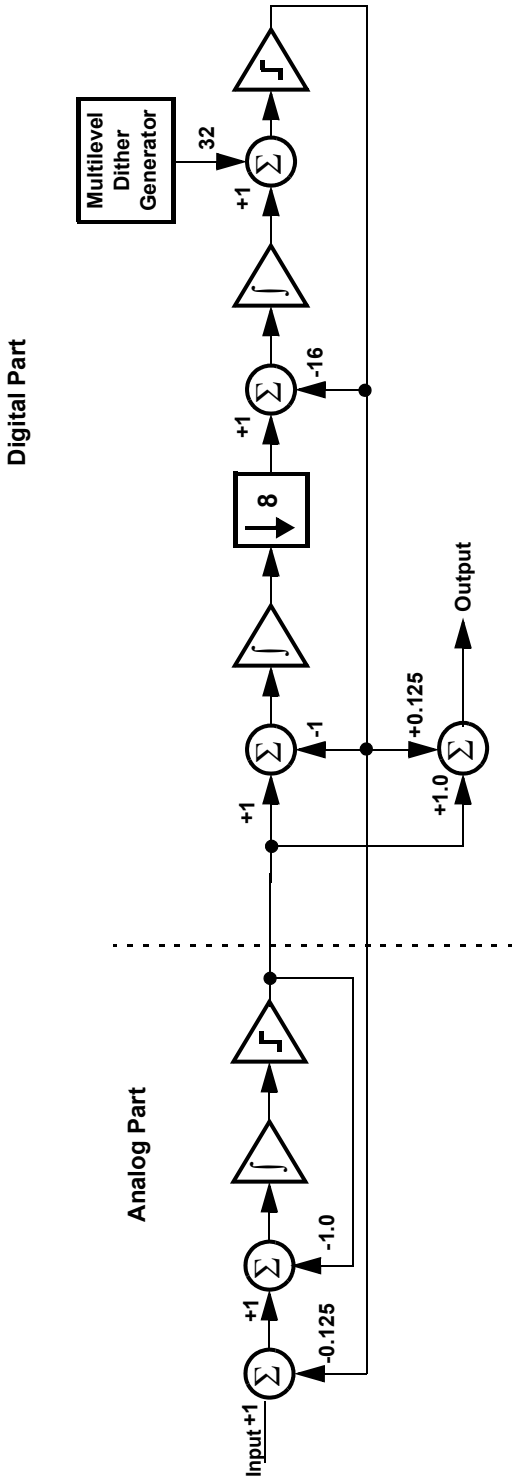


Figure 25-4. Functional Diagram of the A/D Sigma-Delta Converter

25.4.1.2 Decimation Filter

The filter is designed as a general programable decimation filter with a fractional decimation factor. The filter is capable of performing decimation by a fractional ratio with a resolution of one-fourth (by a factor N/4 when N is an integer). The minimal available decimation ratio is 256 and the maximal is 511.75. For the GSM setup the decimation ratio is 406.25. The specific decimation ratios that can be used in the Neptune IC are given in Table 25-12. The decimation filter architecture is a variation of a comb filter and it has a finite impulse response and a liner phase.

The filter DC gain at the A/D path is given in Equation 25-1.

$$H_T(0) = 16 \cdot K \cdot (K^2 - 36) \cdot 2^{-30} \qquad \textbf{Eqn. 25-1}$$

when D is the decimation factor, K is the integer part of D,

$$K = \lfloor D \rfloor \qquad \textbf{Eqn. 25-2}$$

The 2^{-30} factor is due to an internally incorporated scaling by shift.

The filter transfer function is:

$$H_{ADF}(z) = \frac{H_{FIR}(z)}{(1 - z^{-1}) \cdot (1 - z^{-4})^2} \qquad \textbf{Eqn. 25-3}$$

where z conforms to the sampling rate of 4× $F_{\Sigma\Delta}$, and the transfer function of the FIR filter for even values of K = int(D), is expressed in Equation 25-4.

$$H_{FIR}(z)\big|_{K \text{ even}} = \frac{A_0}{4}(1 - z^{-(4 \cdot K - 24)}) \cdot (1 - z^{-4 \cdot K}) \qquad \textbf{Eqn. 25-4}$$
$$\cdot (1 - z^{-(4 \cdot K + 24)}) \cdot (1 + z^{-(2 \cdot K - 6)}) \cdot (1 + z^{-(2 \cdot K + 6)})$$

where A_0 ia a scaling factor (equal to 4).

Transfer function of the FIR filter provides third multiplicity of a zero at the null frequency, fifth multiplicity of a zero at the frequency $F_{\Sigma\Delta}$ and third multiplicity of a zero at the frequency $2 \times F_{\Sigma\Delta}$ (the FIR filter operate at the rate $4 \times F_{\Sigma\Delta}$). The transfer function of the ADF in the frequency domain is given in Equation 25-5 on page 25-23, for even values of K.

$$H_{ADF}(f) = \frac{A_0}{\sin\left(\frac{\pi \cdot f}{4 \cdot F_S}\right) \cdot \sin^2\left(\frac{\pi \cdot f}{F_S}\right)} \cdot \sin\left(\frac{\pi \cdot (K-6) \cdot f}{F_S}\right) \cdot \sin\left(\frac{\pi \cdot K \cdot f}{F_S}\right) \cdot \sin\left(\frac{\pi \cdot (K+6) \cdot f}{F_S}\right) \cdot \cos\left(\frac{\pi \cdot (K-3) \cdot f}{2 \cdot F_S}\right) \cdot \cos\left(\frac{\pi \cdot (K+3) \cdot f}{2 \cdot F_S}\right)$$ **Eqn. 25-5**

where A0 is a scaling factor. Figure 25-6 through Figure 25-9 on page 25-25, show the frequency response of the filter for even K.

For an odd K the $H_{FIR}(z)$ is given by:

$$\begin{aligned} H_{\text{FIR}}(z)\big|_{K \text{ odd}} = \frac{A_0}{4}(&-1 - z^{-(2K-4)} - z^{-(2K+8)} + z^{-(4K-24)} \\ &+ z^{-(4K+24)} + z^{-(6K-28)} + z^{-(6K-16)} + z^{-(6K-4)} \\ &+ z^{-(6K+8)} + z^{-(6K+20)} + z^{-(6K+32)} - z^{-(10K-28)} \\ &- z^{-(10K-16)} - z^{-(10K-4)} - z^{-(10K+8)} - z^{-(10K+20)} \\ &- z^{-(10K+32)} - z^{-(12K-24)} - z^{-(12K+24)} + z^{-(14K-4)} \\ &+ z^{-(14K+8)} + z^{-16K}) \end{aligned}$$ **Eqn. 25-6**

Figure 25-5 shows the decimation filter block scheme.

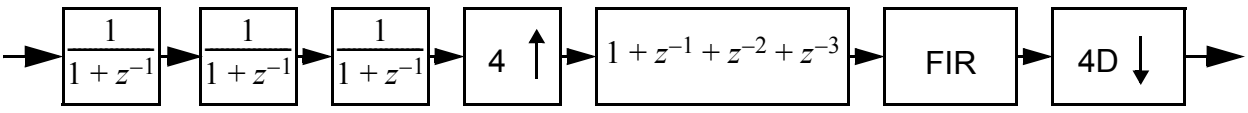


Figure 25-5. Decimation Filter Block Scheme

In the hardware implementation the filter receives 4 bit input, represented as 1.3 (one signed and 3 fractional bits). The output is 21 bits, represented as 3.18 (sign bit plus 2 integer bits and 18 fractional bits). The filter incorporates an internal scaling by 2^{-30}.

a.

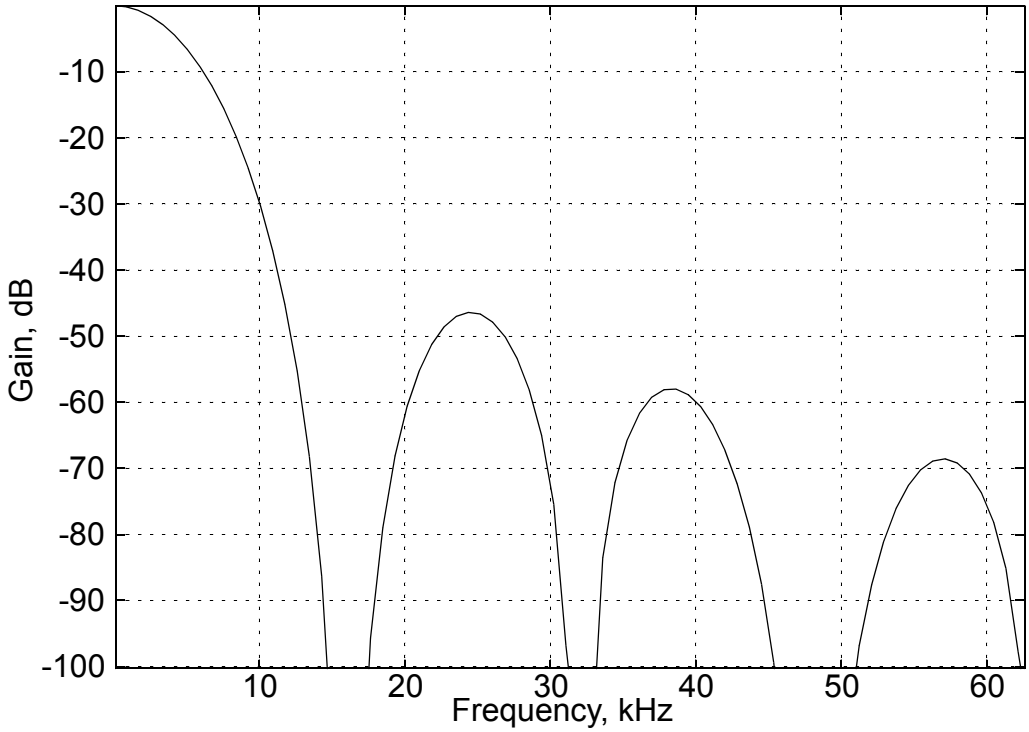


Figure 25-6. Frequency Response of the Filter for Even K (a) normalized to 0 dB at DC.

b.

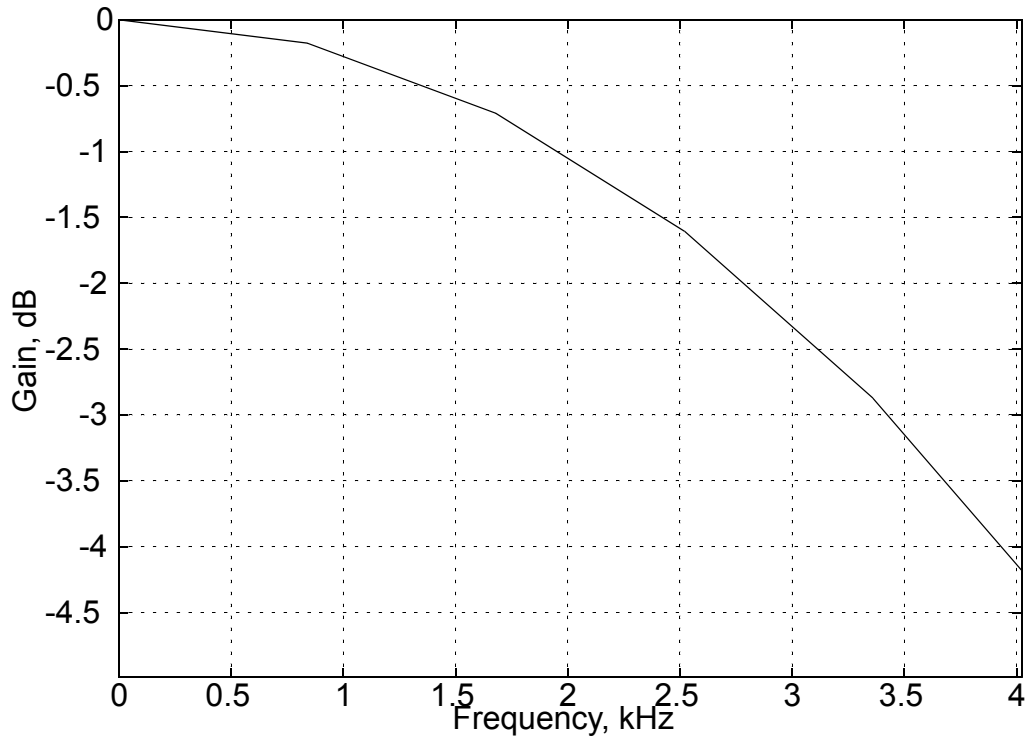


Figure 25-7. Frequency Response of the Filter for Even K (b)

c.

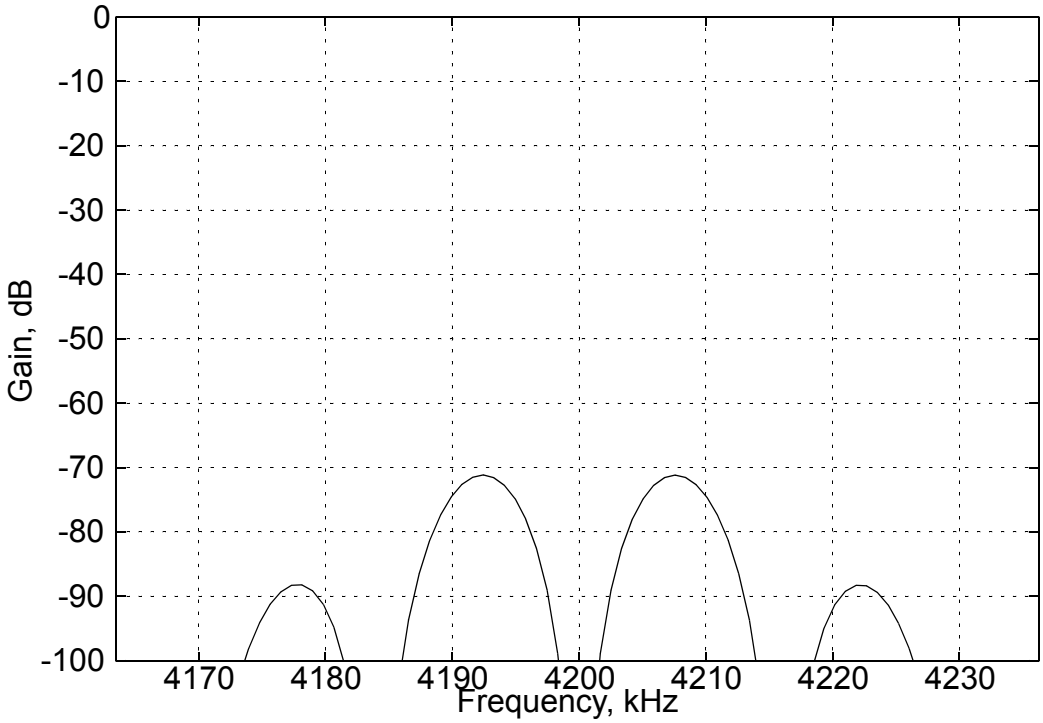


Figure 25-8. Frequency Response of the Filter for Even K (c)

d.

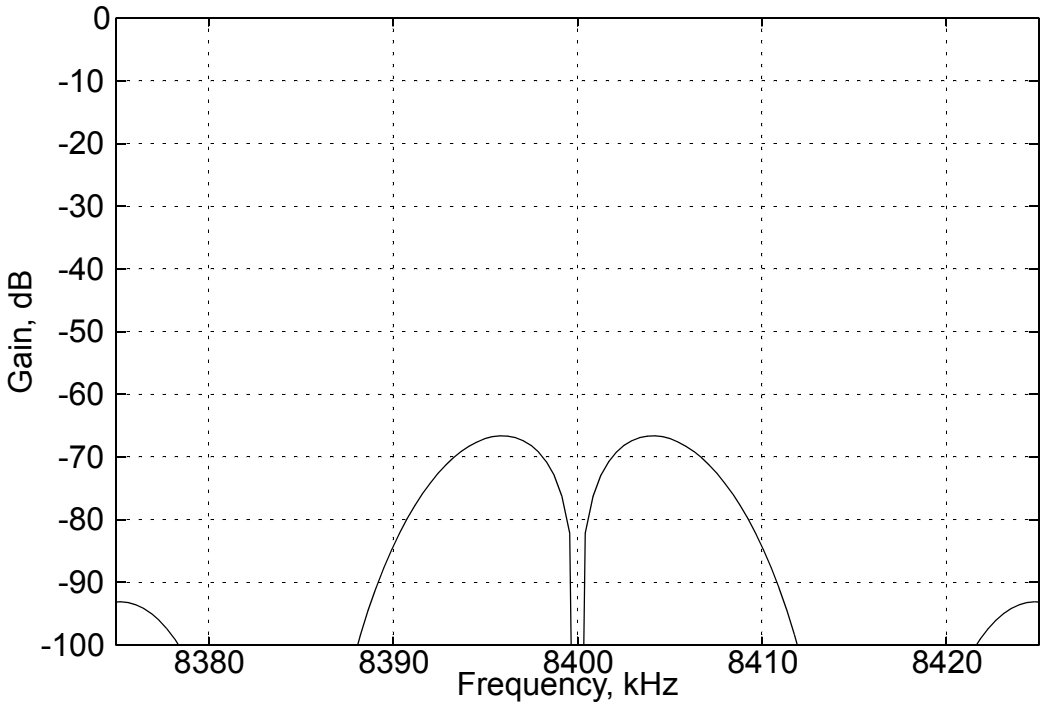


Figure 25-9. Frequency Response of the Filter for Even K (d)

25.4.1.3 Digital Gain Compensator

The function of this module is to adjust the gain along the ADC signal path to meet the spec requirements for relation between the analog voltage at the ADC input and the digital representation of the sample in the VCRX register. The ADC requirements state that the analog voltage level should translate in to digital samples values according the following rule: $340mV_{rms}$ on A/D differential input (ADC_P - ADC_M) is equal to 0dBm0 level for 1.02kHz tone signal. 0dBm0 corresponds to -3dB of full scale signal. This results in the following: (ADC_P - ADC_M) = 680 mVolt translates to VCRX = '0,1111,1111,1111'B.

The gain of some modules along the signal path is not constant and depends on the control configuration. The following list summarizes the gain contribution of each signal path stage:

- **Analog portion -** The analog portion should translate (ADC_P - ADC_M) = 680 mVolt to analog Sigma Delta output full scale for 1.02kHz signal.
- **Sigma-Delta modulator -** The Sigma-Delta modulator gain is 1 (0 dB).
- **Decimation filter -** Decimation filter gain depends on the decimation ratio according to the following results (Table 25-13 and Table 25-14):

Table 25-13. Decimation filter gain (a)

VCLK[2:0]	CODEC Input clock CODEC_CLK	Sigma Delta Clock $F_{\Sigma\Delta}$	Intermediate frequency	Software Interface Frequency	Decima-tion Ratio
VCLK[2:0]=0, 1	26MHz	6.5MHz	16kHz	8kHz	406.25
VCLK[2:0]=2, 5, 6	16.8MHz	4.2MHz	16kHz	8kHz	262.5
VCLK[2:0]=3	19.44MHz	4.86MHz	16kHz	8kHz	303.75
VCLK[2:0]=4	19.44MHz	4.86MHz	16.2kHz	8.1kHz	300
VCLK[2:0]=7	TBD	TBD	TBD	TBD	TBD

Table 25-14. Decimation filter gain (b)

VCLK[2:0]	Filter DC gain	Filter gain at 1.02 kHz	Decimation filter internal scaling (with shift)	Total signal gain at 1.02 kHz
VCLK[2:0]=0, 1	180.592 dB	180.331 dB	2^{-30}	0.967462 (-0.287 dB)
VCLK[2:0]=2, 5, 6	169.176 dB	168.915 dB	2^{-30}	0.259931 (-11.703 dB)
VCLK[2:0]=3	172.966 dB	172.705 dB	2^{-30}	0.40212 (-7.913 dB)
VCLK[2:0]=4	172.706 dB	172.451 dB	2^{-30}	0.390522 (-8.167 dB)
VCLK[2:0]=7	TBD	TBD	TBD	TBD

- **Gain Compensator -** The function of this stage is to compensate for the various gain changes along the signal path in all the operation modes in order to meet the required analog to digital translation. Table 25-15 summarizes the various gain values.
- **LPF filter -** The LPF filter is constructed from three biquad stages. The total LPF filter gain (at 1.02 kHz) is 27.432 (28.8 dB).

- **Decimation by 2 -** Decimation by 2 is performed by dropping every second sample. This operation doesn't change the signal gain.
- **HPF filter -** The HPF filter gain (at 1.02 kHz) is 1.10102 (0.836 dB). Note that the HPF filter can be bypassed by s/w control, in this case the equivalent gain is 1 (0dB).

The gain line-up along the signal path is summarized in Table 25-15 . Note that the total gain corresponds to input data in VCTX register represented as a fractional number. Fractional representation means that the MSB bit is a sign bit and the decimal point located between bit #12 and #11 (for example negative full scale -1 is represented as 1000h in VCRX).

Table 25-15. ADC signal path gain lineup (for 1.02 kHz signal)

mode	VCLK [2:0]	Analog portion	Deci-mation filter	Gain Compen-sator	LPF fil-ter	Deci-mation by 2	HPF fil-ter	Total ADC path gain
Without IIR filters (test mode)	0, 1	1/0.68 V^{-1}	0.96746	0.5	1	1	1	0.7114 V^{-1}
	2, 5, 6		0.25993	0.5				0.1911 V^{-1}
	3		0.40212	0.5				0.2957 V^{-1}
	4		0.39052	0.5				0.2871 V^{-1}
Without HPF	0, 1		0.96746	2^{-3} x 0.30078125	27.432		1	1.4674 V^{-1}
	2, 5, 6		0.25993	2^{-3} x 1.12109375				1.4695 V^{-1}
	3		0.40212	2^{-3} x 0.7265625				1.4732 V^{-1}
	4		0.39052	2^{-3} x 0.74609375				1.4691 V^{-1}
With HPF	0, 1		0.96746	2^{-3} x 0.2734375			1.10102	1.4687 V^{-1}
	2, 5, 6		0.25993	2^{-3} x 1.01953125				1.4713 V^{-1}
	3		0.40212	2^{-3} x 0.66015625				1.4738 V^{-1}
	4		0.39052	2^{-3} x 0.6796875				1.4737 V^{-1}

25.4.1.4 IIR Low-Pass Filter

The LPF filter is a 6^{th} order IIR filter constructed as three biquad "Direct Form II" cascaded sections of the form given in Eq. 25-7.

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}} \qquad \textit{Eqn. 25-7}$$

Due to the hardware implementation b_0 and a_0 are set to 1 and the filter stages have a non-unit gain. The filer coefficients are summarized in Table 25-16. The total filter gain at 1.02 kHz is 27.432 (28.8 dB).

Table 25-16. LPF Filter Coefficients

Filter	b_0	b_1	b_2	a_0	a_1	a_2
LPF filter stage 1	1	1.84375	1	1	-0.453125	0.140625
LPF filter stage 2	1	1.0625	1	1	-0.390625	0.46875
LPF filter stage 3	1	0.53125	1	1	-0.296875	0.8125

LPF filter frequency response is given in Figure 25-10. Note that the purpose of the rise in the pass-band is to compensate the decimation filter drop in this region.

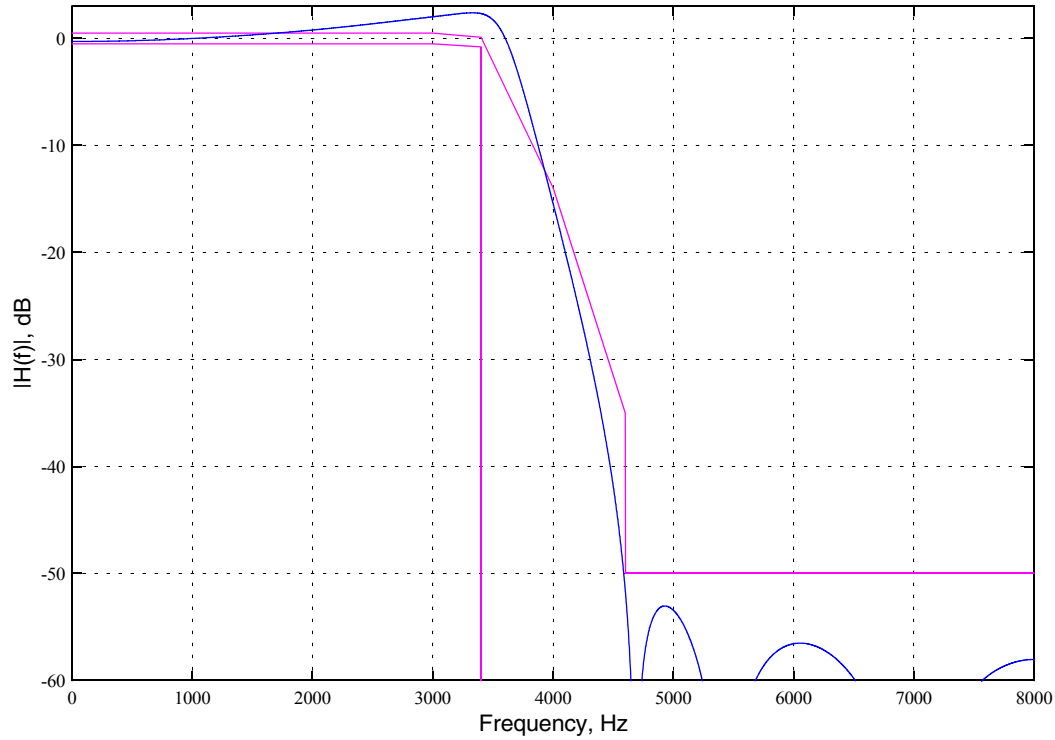


Figure 25-10. LPF filter frequency response

Figure 25-11, Figure 25-12, and Figure 25-13 show the combined frequency response of the LPF filter with the decimation filter.

The maximal ripple of the combined frequency response in the pass band (0:3000 Hz) is **0.046 dB**. The attenuation in the stop band (above 4600 Hz) is **57.06 dB**, about 10 dB more than required.

LPF filter absolute group delay at 1.6 kHz is **152.9 μSec.** The decimation filter group delay is **124.75 μSec**. The total absolute group delay at 1.6 kHz without HPF is **277.7 μSec**.

NOTE:

The frequency response deviation in the 3400:4000 Hz range is acceptable by spec requirements.

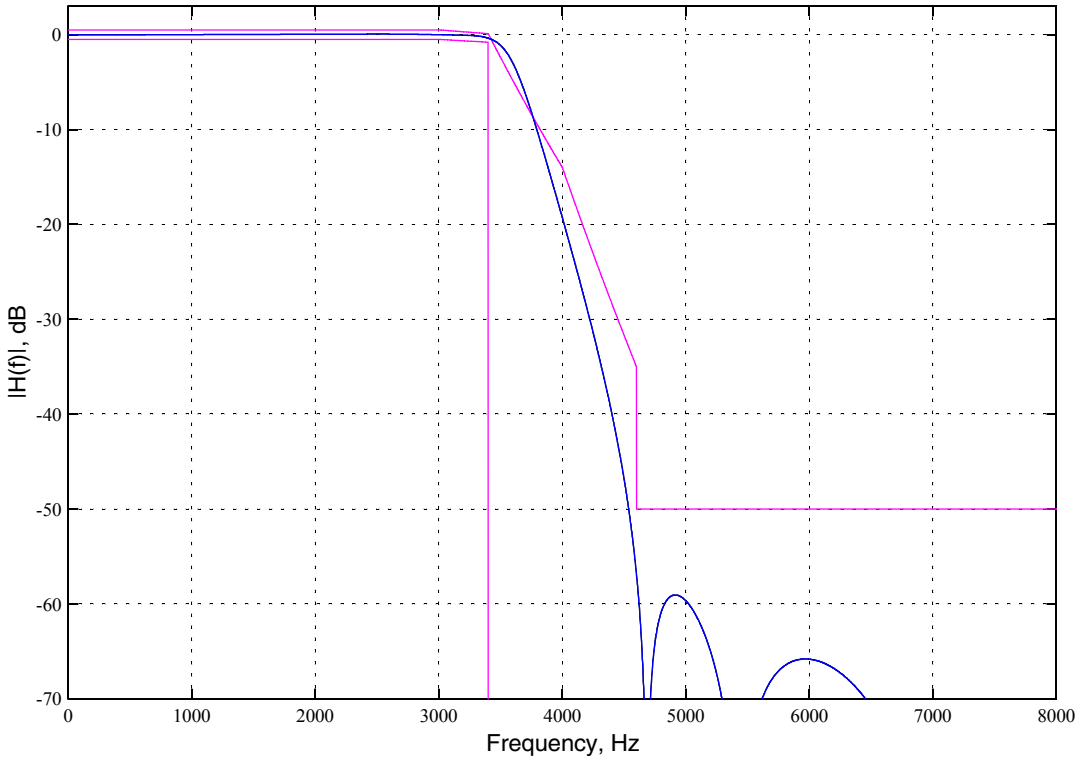


Figure 25-11. Total Frequency response without HPF (decimation filter + LPF) (a)[1]

1. The deviation of the frequency response out of limits in the 3.4 kHz - 4 kHz range is acceptable by spec requirements.

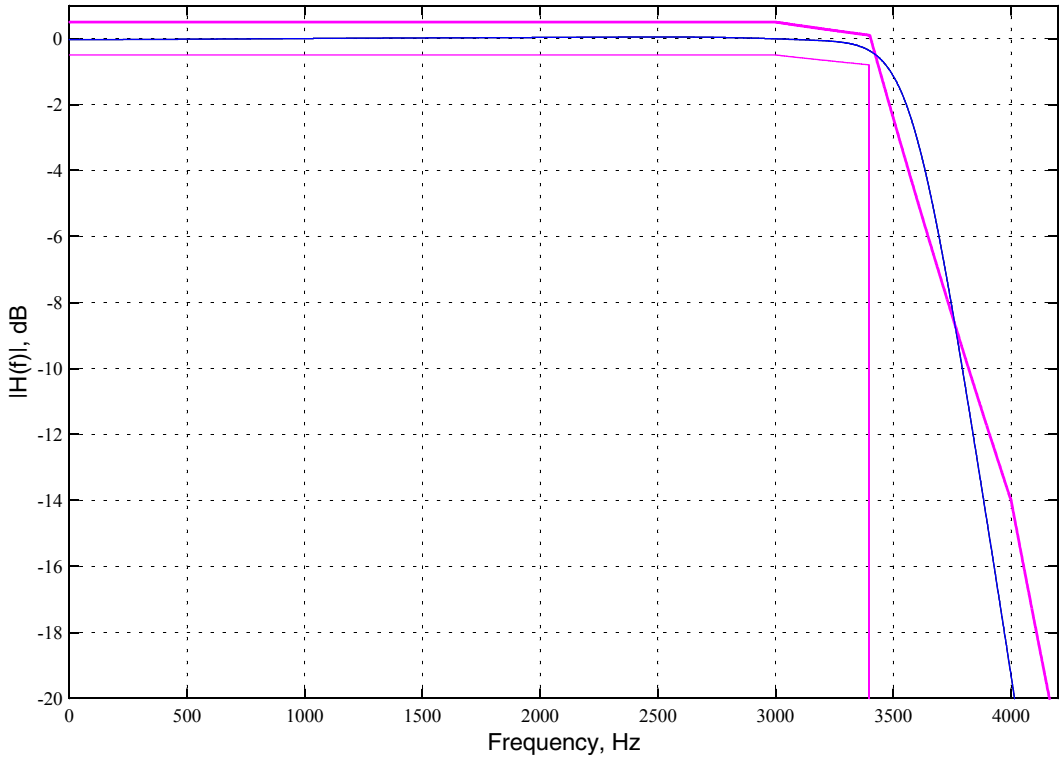


Figure 25-12. Total Frequency response without HPF (b)

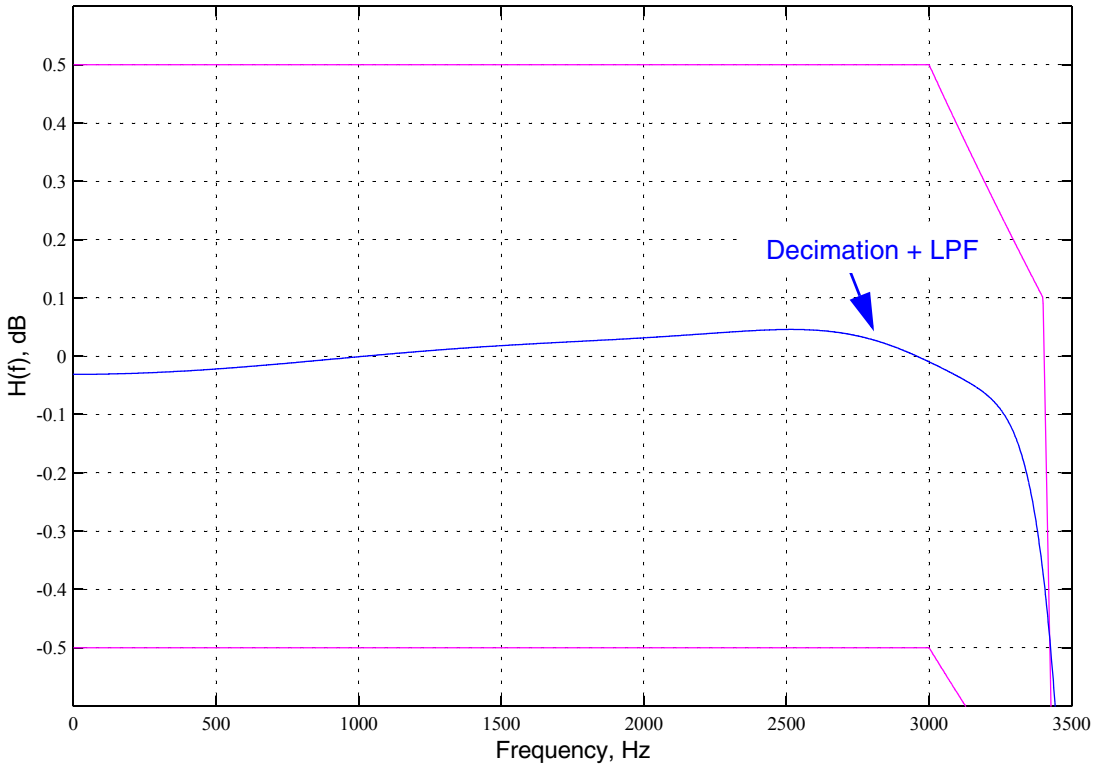


Figure 25-13. Total Frequency response without HPF (c)

Figure 25-14 shows the total relative group delay without the HPF. The exact values of the group delay at the requirements points are given in Table 25-17.

Table 25-17. LPF filter relative group delay

Frequency	ADC requirement	DAC requirement	Designed Filter	Note
0 Hz	-40 μSec	-40 μSec	-36.3 μSec	Minimal requirement
2600 Hz	100 μSec	100 μSec	98.5 μSec	Maximal requirement
2800 Hz	150 μSec	160 μSec	139.5 μSec	Maximal requirement
3000 Hz	200 μSec	200 μSec	197.8 μSec	Maximal requirement

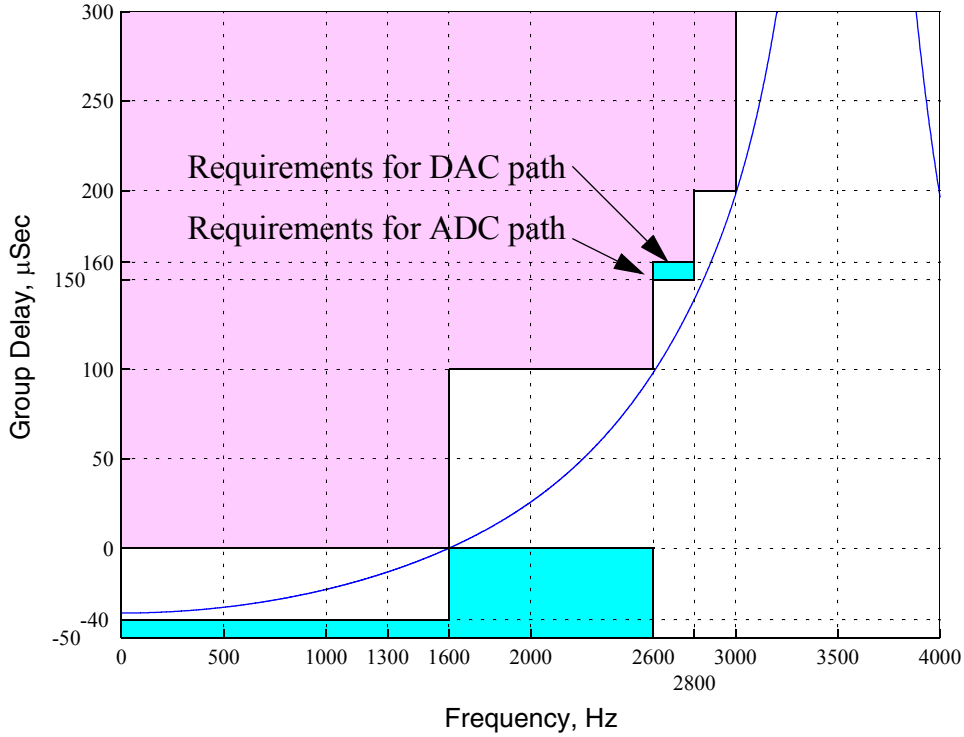


Figure 25-14. Total Group Delay without HPF

25.4.1.5 IIR High-Pass Filter

The HPF filter operates at 8 kHz sampling rate (8.1 kHz for VCLK[2:0] = 4). It is implemented as a single biquad "Direct Form II" section (second order) with the discrete time shown in Equation 25-8.

$$y_{hpf}[n] = x[n] + (-2 + 2^{-10})x[n-1] + x[n-2] - \{(-2 + 2^{-3} + 2^{-4} - 2^{-8})y_{hpf}[n-1] + (1 - 2^{-3} - 2^{-5} - 2^{-7})y_{hpf}[n-2]\}$$ ***Eqn. 25-8***

The transfer function is of the form given in Eq. 25-9

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}} \qquad \textit{Eqn. 25-9}$$

with coefficients given in Table 25-18 .

Table 25-18. HPF Filter Coefficients

Filter	b_0	b_1	b_2	a_0	a_1	a_2
HPF filter stage	1	-1.9990234375	1	1	-1.81640625	0.8359375

Figure 25-15 through Figure 25-16, show frequency response of the HPF filter compared with the requirements limits.

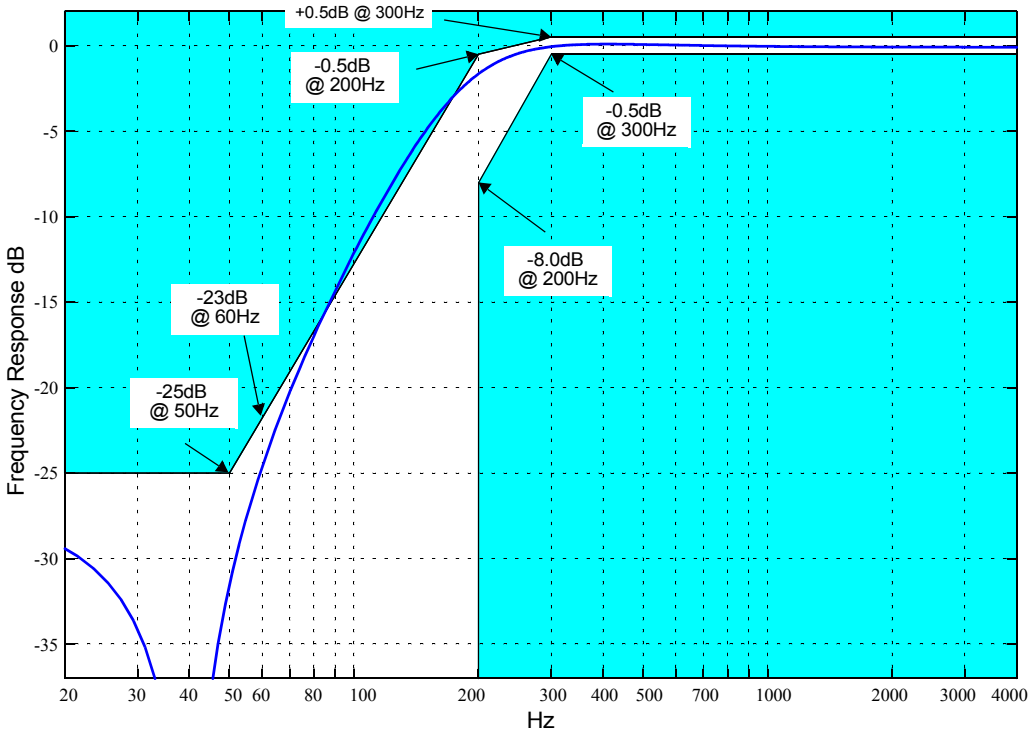


Figure 25-15. HPF Filter Frequency Response (a)

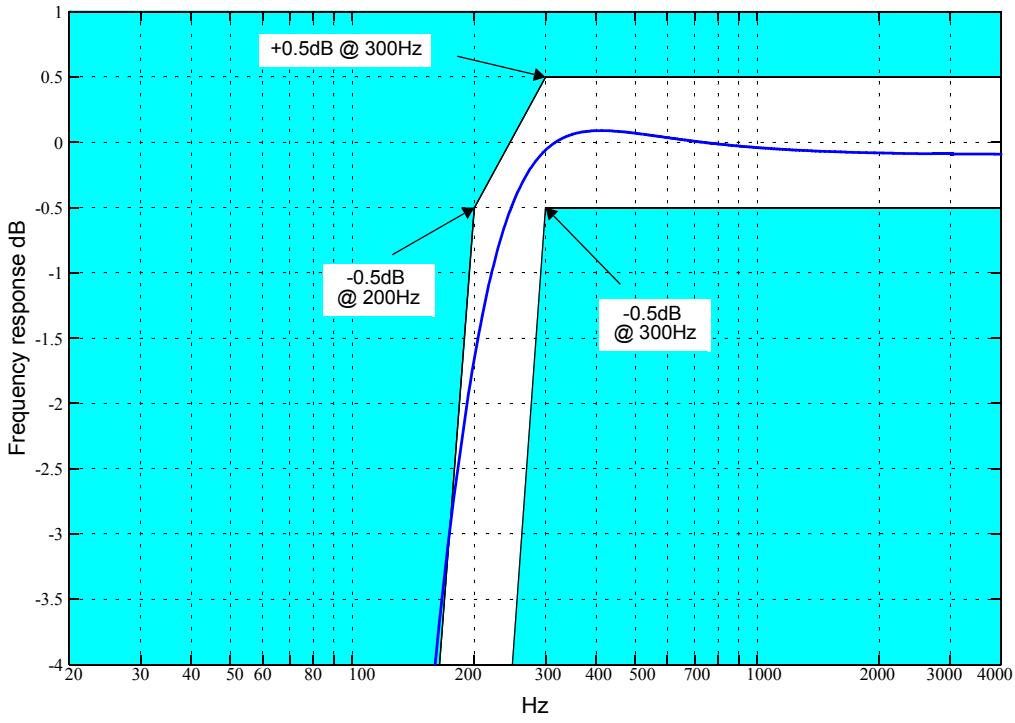


Figure 25-16. HPF Filter Frequency Response (b)

Figure 25-17, Figure 25-18, and Figure 25-19 show the combined frequency response of the whole path including the decimation filter, the LPF filter and the HPF filter.

The attenuation in the first stop band (0:50 Hz) is **26.89 dB**. The maximal ripple of the combined frequency response in the pass band (300:3000 Hz) is **0.107 dB**. The attenuation in the stop band (above 4600 Hz) is **57.11 dB**, about 10 dB more than required.

NOTE:

The frequency response deviation in the 70:180 Hz and 3400:4000 Hz ranges is acceptable by spec requirements.

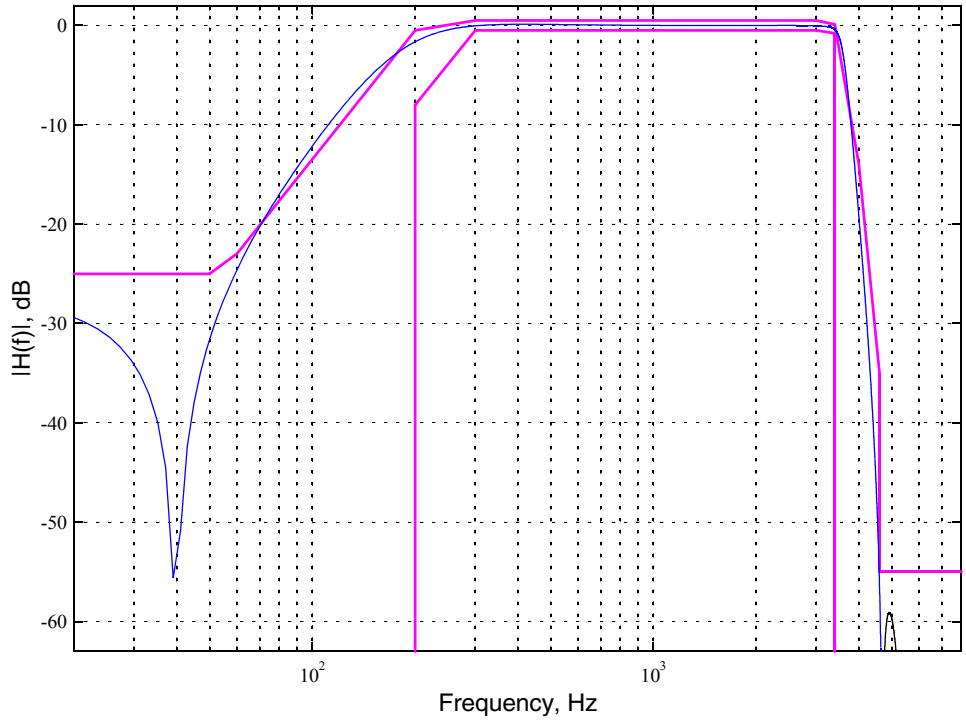


Figure 25-17. Total Frequency response with HPF (a)[1]

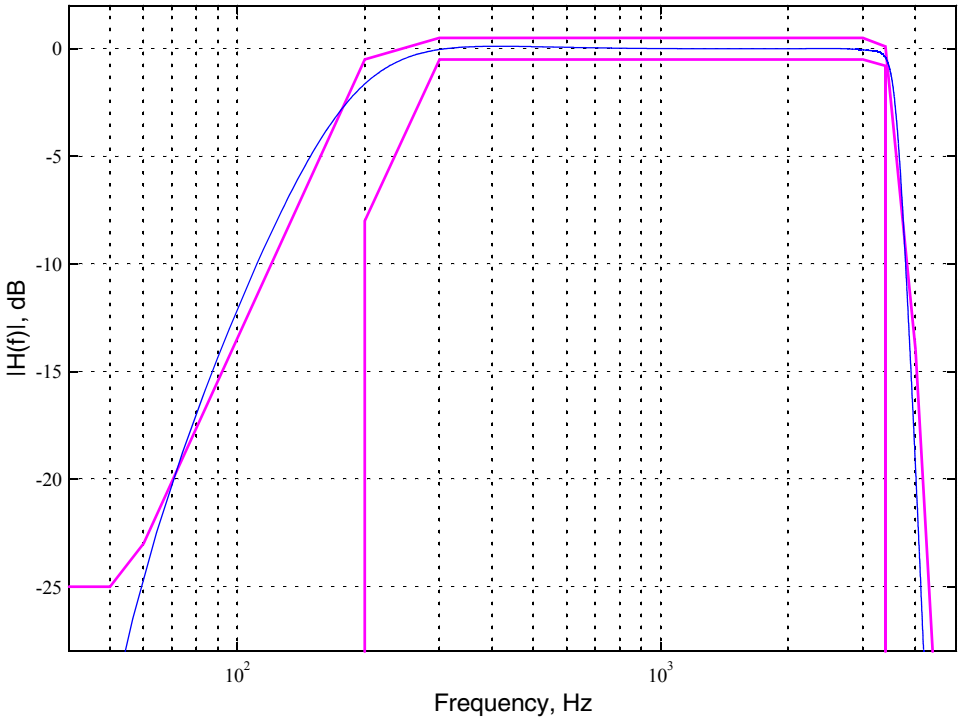


Figure 25-18. Total Frequency response with HPF (b)

1. The deviation of the frequency response out of limits in the 70 - 180Hz range is acceptable by spec requirements.

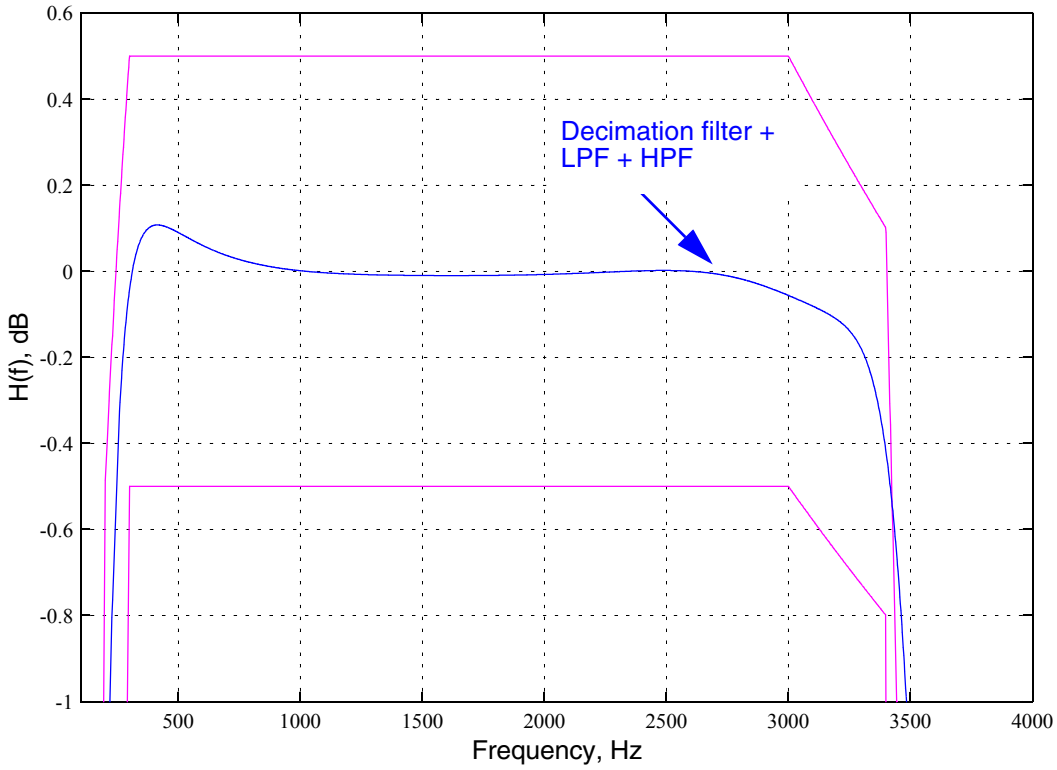


Figure 25-19. Total Frequency response with HPF (c)

Figure 25-20 shows the total relative group delay with the HPF. The exact values of the group delay at the requirements points are given in Table 25-19. HPF filter absolute group delay at 1.6 kHz is **16.5** μ**Sec**. Total absolute group delay at 1.6 kHz (including decimation filter, LPF, and HPF) is **294.2** μ**Sec**.

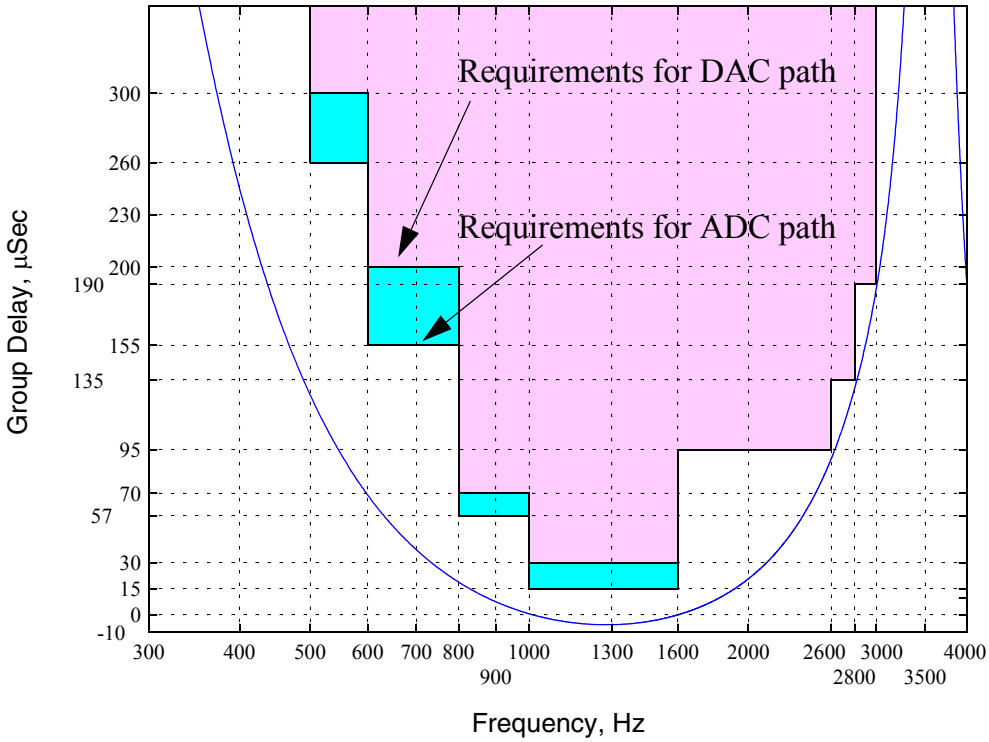


Figure 25-20. Total Group Delay with HPF

Table 25-19. LPF filter relative group delay

Frequency	ADC requirement	DAC requirement	Designed Filter
500 Hz	**260 µSec**	**300 µSec**	**126.4 µSec**
600 Hz	**155 µSec**	**200 µSec**	**68.8 µSec**
800 Hz	**57 µSec**	**70 µSec**	**18.8 µSec**
1000 Hz	**15 µSec**	**30 µSec**	**0.6 µSec**
2600 Hz	**95 µSec**	**95 µSec**	**89.7 µSec**
2800 Hz	135 µSec	135 µSec	130.0 µSec
3000 Hz	190 µSec	190 µSec	187.9 µSec

25.4.2 Voice Codec DAC Operation

Figure 25-21 shows the signal path block diagram for the voice Digital to Analog conversion.

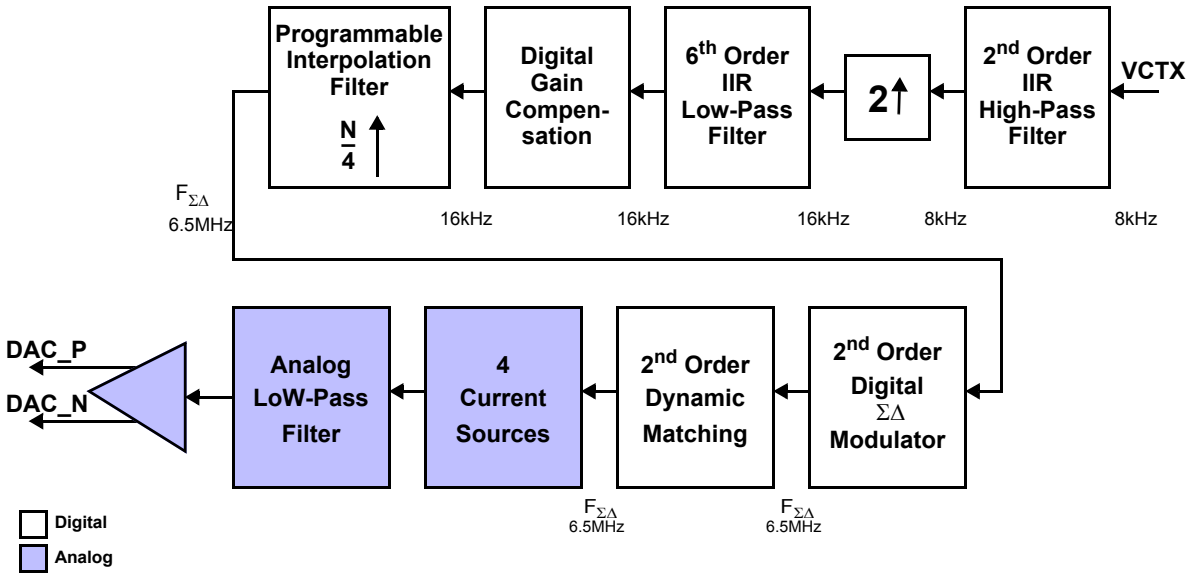


Figure 25-21. Voice Codec DAC Signal Path Block Diagram[1]

The digital-to-analog signal path contains the following components. The data from the VCTX register is read at about an 8 kHz rate and transferred through a high-pass filter. The filter is a second order IIR. The high-pass filter is identical to the HPF at the ADC path and it is optionally enabled by the VCOHPF control bit. Next, the signal is interpolated by two, passes through a 6th order, IIR LPF and gain corrected. The LPF filter is identical to the LPF filter at the ADC path. The resulting signal at about 16 kHz sampling rate is interpolated to a high frequency and low-pass filtered by the Programmable Interpolation filter. The filter is capable of performing interpolation by a fractional ratio with a resolution of one-fourth (by a factor N/4 when N is an integer). For the GSM setup the interpolation ratio is 406.25. The interpolation filter architecture is a variation of a comb filter and it has a finite impulse response and a linear phase. The transfer function of the interpolation filter is the same as the decimation filter at the ADC path. Next, the signal is quantized into five levels with a second order, digital Sigma-Delta modulator. The analog-to-digital conversion is performed with four identical current sources that are enabled or disabled according to the sigma-delta output. Because of a possible gain mismatch errors of up to 15% between the current sources, the specific current source selection is performed by a Dynamic Matching module. The Dynamic Matching module action is to shape the noise resulting from current sources mismatch into a second order shape. Finally the signal is filtered with 50 kHz second order low-pass analog filter and output as a differential signal to DAC_P and DAC_N pins.

25.4.2.1 IIR High-Pass Filter

The HPF filter is identical to HPF filter on the ADC path.

25.4.2.2 IIR Low-Pass Filter

The LPF filter is identical to the LPF filter on the ADC path. Figure 25-22 and Figure 25-23 show the total frequency response at the pass band with and without the HPF filter. The figures also show the influence of the analog low pass filter in this band.

1. The clock frequencies are for VCLK=0 (GSM)

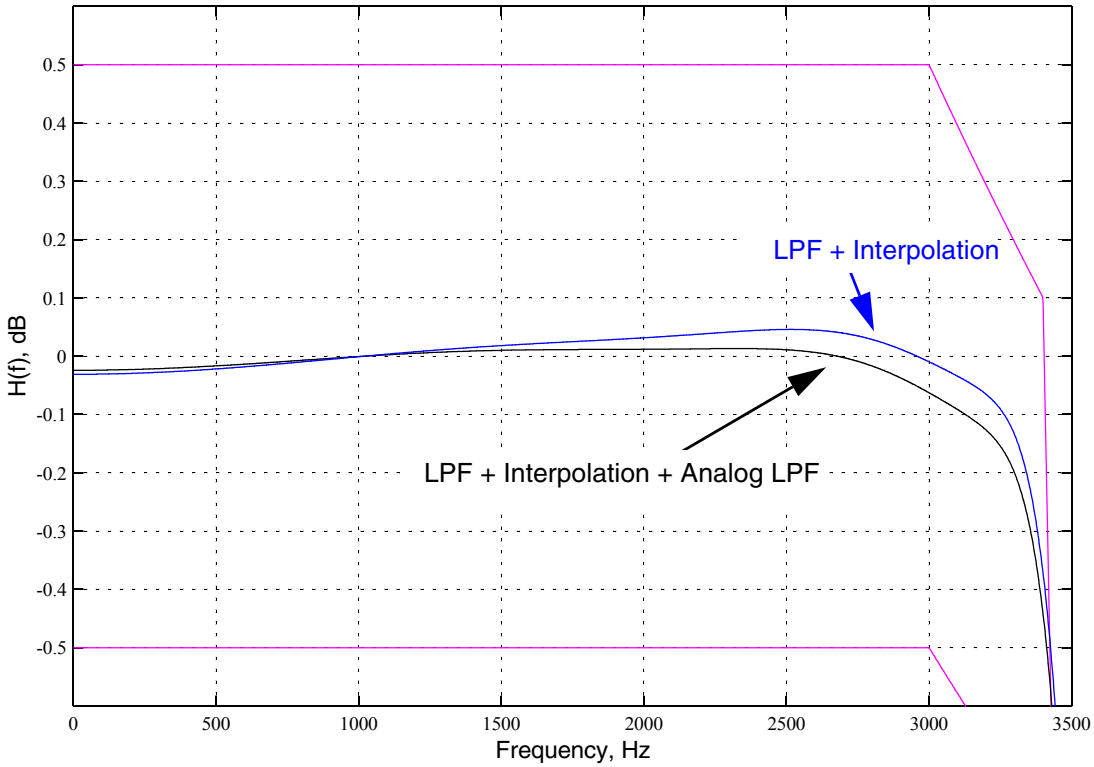


Figure 25-22. Total DAC Frequency response without HPF

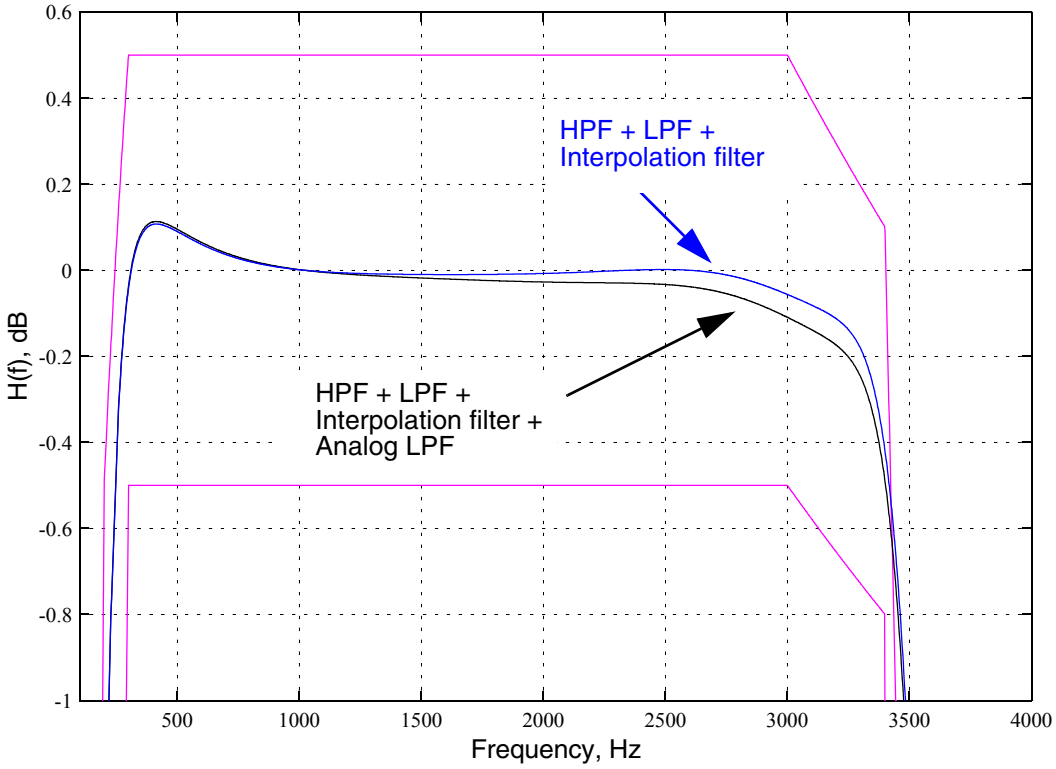


Figure 25-23. Total DAC Frequency response with HPF

25.4.2.3 Digital Gain Compensator

The function of this module is to adjust the gain along the DAC signal path to meet the spec requirements for relation between the digital representation of the sample in the VCTX register and the analog voltage at the DAC output. The DAC requirements state that the digital samples values should translate in to analog voltage level according the following rule: 0dBm0 = 500 mVrms on the D/A differential output (DAC_P - DAC_M) for 1.02kHz tone signal. 0dBm0 corresponds to -3dB of full scale signal. This results in the following: VCTX = '0111,1111,1111,1111'B translates to DAC output (DAC_P - DAC_M) = 1 volt.

The gain of some modules along the signal path is not constant and depends on the control configuration. The following list summarizes the gain contribution of each signal path stage:

- **HPF filter -** The HPF filter gain (at 1.02 kHz) is 1.10102 (0.836 dB). Note that the HPF filter can be bypassed by s/w control, in this case the equivalent gain is 1 (0dB).
- **Interpolation by 2 -** Interpolation by 2 is performed by zero insertion. This results in gain attenuation by 2.
- **LPF filter -** The LPF filter is constructed from three biquad stages. The total LPF filter gain (at 1.02 kHz) is 27.432 (28.8 dB).
- **Gain Compensator -** The function of this stage is to compensate for the various gain changes along the signal path in all the operation modes in order to meet the required digital to analog translation. Table 25-22 summarizes the various gain values.
- **Interpolation filter -** Interpolation filter gain depends on the interpolation ratio according to the following results:

Table 25-20. Interpolation filter gain (a)

VCLK[2:0]	CODEC Input clock CODEC_CLK	Software Interface Frequency	Intermediate frequency	Sigma Delta Clock $F_{\Sigma\Delta}$	Interpola-tion Ratio
VCLK[2:0]=0, 1	26MHz	8kHz	16kHz	6.5MHz	406.25
VCLK[2:0]=2, 5, 6	16.8MHz	8kHz	16kHz	4.2MHz	262.5
VCLK[2:0]=3	19.44MHz	8kHz	16kHz	4.86MHz	303.75
VCLK[2:0]=4	19.44MHz	8.1kHz	16.2kHz	4.86MHz	300
VCLK[2:0]=7	TBD	TBD	TBD	TBD	TBD

Table 25-21. Interpolation filter gain (b)

VCLK[2:0]	Filter gain at 1.02 kHz	Interpola-tion gain	Signal gain (including inter-polation and filter) at 1.02 kHz	Interpolation filter internal scaling (with shift)	Total signal gain (including interpo-lation and filter) at 1.02 kHz
VCLK[2:0]=0, 1	180.331 dB	1/1625	116.114 dB	2^{-18}	2.4386 (7.743 dB)
VCLK[2:0]=2, 5, 6	168.915 dB	1/1050	108.491 dB	2^{-17}	2.02796 (6.141 dB)
VCLK[2:0]=3	172.705 dB	1/1215	111.014 dB	2^{-17}	2.71125 (8.663 dB)
VCLK[2:0]=4	172.451 dB	1/1200	110.867 dB	2^{-17}	2.66596 (8.517 dB)
VCLK[2:0]=7	TBD	TBD	TBD	TBD	TBD

- **Sigma-Delta modulator -** The Sigma-Delta modulator gain is 1 (0 dB). Because of the limiting and resolution considerations the full scale signal level should be limited to 95% of the Sigma Delta possible full scale.
- **Analog portion -** The analog portion should translate 95% of Sigma Delta output full scale to (DAC_P - DAC_M) = 1 volt for 1.02kHz signal.

The gain line-up along the signal path is summarized in Table 25-22 . Note that the total gain corresponds to input data in VCTX register represented as a fractional number. Fractional representation means that the MSB bit is a sign bit and the decimal point located between bit #15 and #14 (for example negative full scale -1 is represented as 8000h in VCTX).

Table 25-22. DAC signal path gain lineup (for 1.02 kHz signal)

mode	VCLK [2:0]	HPF filter	Interpolation by 2	LPF filter	Gain Compensator	Interpolation filter	Analog portion	Total DAC path gain
Without IIR filters (test mode)	0, 1	1	1	1	1	2.4386	1/0.95 Volt	2.5669 V
	2, 5, 6				1	2.02796		2.1347 V
	3				1	2.71125		2.8539 V
	4				1	2.66596		2.8063 V
Without HPF	0, 1	1	1/2	27.432	2^{-5} x 0.90625	2.4386		0.9971 V
	2, 5, 6				2^{-5} x 1.09375	2.02796		1.0008 V
	3				2^{-5} x 0.8203125	2.71125		1.0035 V
	4				2^{-5} x 0.828125	2.66596		0.9961 V
With HPF	0, 1	1.10102			2^{-5} x 0.828125	2.4386		1.0032 V
	2, 5, 6				2^{-5} x 0.9921875	2.02796		0.9995 V
	3				2^{-5} x 0.7421875	2.71125		0.9996 V
	4				2^{-5} x 0.7578125	2.66596		1.0036 V

In midi mode the gain needs to be changed according to the interpolatin ratio. The following table illustrates the gain settings required for midi quality sound with a 26MHz codec clock.

Table 25-23. DAC signal path gain and interpolation ratio for midi mode(for 1.02 kHz signal)

sampling rate (kHz)	interpolation ratio/2[10:0]	VDIR_MIDI register value	gain_word[7:0]	GR_WORD_MIDI register value
8.000	406.25	$0659	0.90625	$0074
11.025	294.75	$049B	0.86719	$006F
12.000	270.75	$043B	1.03125	$0084
16.000	203.25	$032D	0.91406	$0075

Table 25-23. DAC signal path gain and interpolation ratio for midi mode(for 1.02 kHz signal)

sampling rate (kHz)	interpolation ratio/2[10:0]	VDIR_MIDI register value	gain_word[7:0]	GR_WORD_MIDI register value
22.050	147.50	$024E	0.87500	$0070
24.000	135.50	$021E	1.09837	$0085
32.000	101.50	$0196	0.92969	$0077
44.100	073.75	$0127	0.89844	$0073
48.000	067.75	$010F	1.06250	$0088

25.4.2.4 Interpolation Filter

The filter is designed as a general programmable interpolation filter with a fractional interpolation factor. The filter is capable of performing interpolation by a fractional ratio with a resolution of one-fourth (by a factor N/4 when N is an integer). The minimal available interpolation ratio is 256 and the maximal is 511.75. For the GSM setup the interpolation ratio is 406.25. The specific interpolation ratios that can be used in the Neptune IC are given in Table 25-12. The interpolation filter architecture is a variation of a comb filter and it has a finite impulse response and a liner phase. The filter has the same transfer function as the decimation filter at the ADC path. The interpolation filter operation is synchronized with the decimation filter at the ADC path and they share the same control logic.

The filter gain of the D/A path is in Equation 25-10,

$$H_T(0) = B_0 \cdot \left(\frac{K}{I}\right) \cdot (K^2 - 36) \cdot 2^{-17} \qquad \textit{Eqn. 25-10}$$

when I is the interpolation ratio (equal to A/D path decimation ratio D), K is the integer part of I,

$$K = \lfloor I \rfloor \qquad \textit{Eqn. 25-11}$$

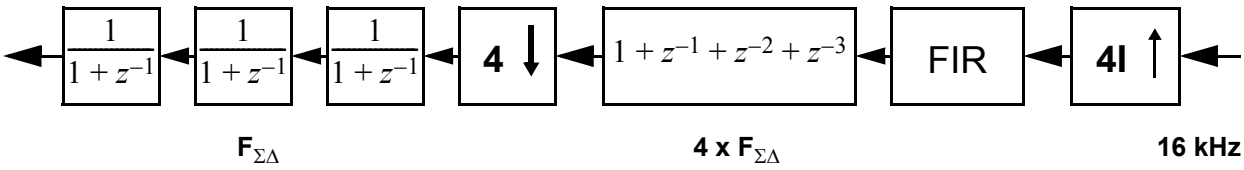
and the output scaling factor B_0 is defined as follows:

$$B_0 = \begin{cases} 2, D = [256,361.75] \\ 1, D = [362,511.75] \end{cases} \qquad \textit{Eqn. 25-12}$$

Note that for fractional Decimation/Interpolation ratio, the gain of the ADC path depends only on the integer part of the decimation ratio, while the gain of the DAC path depends also on the fraction part of the interpolation ratio.

Figure 25-5 on page 25-23 shows the Interpolation filter block scheme.

Figure 25-24. Interpolation Filter Block Scheme



The frequency response of the Interpolation filter is identical to the Decimation filter and is given in Equation 25-3 through Equation 25-6 on page 25-23 and in Figure 25-6 on page 25-24.

In the hardware implementation the filter receives 16 bit input, represented as 1.15 (one signed and 15 fractional bits). The output is also 16 bit wide, represented as 1.15. The filter incorporates an internal scaling by 2^{-18}.

For midi modes the interpolation ratio is programmable through VDIR_MIDI[10:0]. The following table shows the interpolation rates required for commonly used sampling frequencies for midi quality sound with a 26MHz codec clock.

Table 25-24. Midi sampling rate and frequency deviation

Desired Sampling Rate (Hz)	Interpolation Ratio	Actual Sampling Rate (Hz)	Intermediate Frequency (Hz)	Frequency Error (%)	LPF Cutoff Frequency (Hz)
8000	406.25	8000	16000	0.00%	3400
11025	294.75	11026	22053	0.01%	4686
12000	270.75	12004	24007	0.03%	5100
16000	203.25	15990	31980	-0.06%	6800
22050	147.50	22034	44068	-0.07%	9371
24000	135.50	23985	47970	-0.06%	10200
32000	101.50	32020	64039	0.06%	13600
44100	73.75	44068	88136	-0.07%	18743
48000	67.75	47970	95941	-0.06%	20400

25.4.2.5 DAC Sigma-Delta Modulator

The ΣΔ modulator is structured as a second order with five quantization levels. A random dither is inserted before the quantization to eliminate tones. For a reference on the theory of the ΣΔ converters refer to [2, 3].

Low-order sigma-delta modulators are known to generate tones which have especially high level for DC inputs and low-level AC inputs. The tones are sinusoidal interferences. Frequencies and magnitudes of the tones are depended on an input signal. Dithering methods are used to reduce the tones levels by spreading the tones energy in a wide band. The addition of a dither signal is controlled by the VCDITH control bit in the VCCR register.

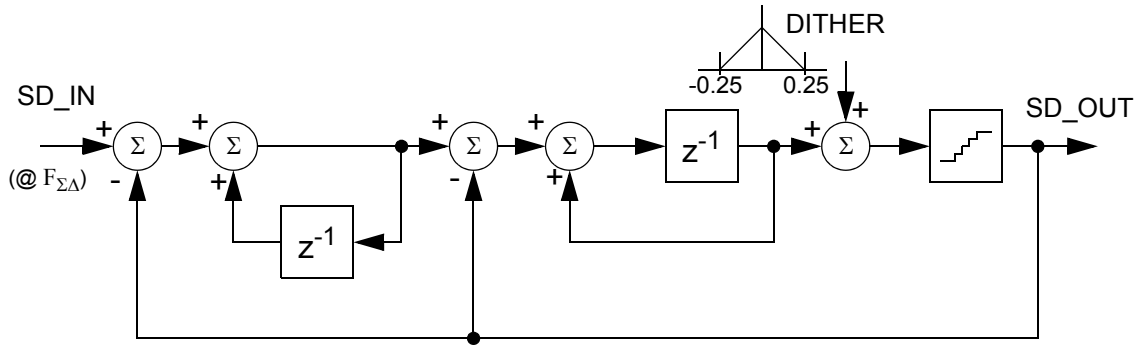


Figure 25-25. ΣΔ Modulator Scheme

Table 25-25 lists the five quantization level values and the corresponding number of current sources that are turned on.

Table 25-25. ΣΔ Modulator Quantization Levels

quantizer input	quantization level	current sources ON
$0.75 \le x$	1.0	4
$0.25 \le x < 0.75$	0.5	3
$-0.25 \le x < 0.25$	0.0	2
$-0.75 \le x < -0.25$	-0.5	1
$x < -0.75$	-1.0	0

Figure 25-26 shows the hardware implementation scheme of the ΣΔ modulator. The (x.y) notation represents the location of the decimal point. x - number of integer bits including sign, y - number of fractional bits.

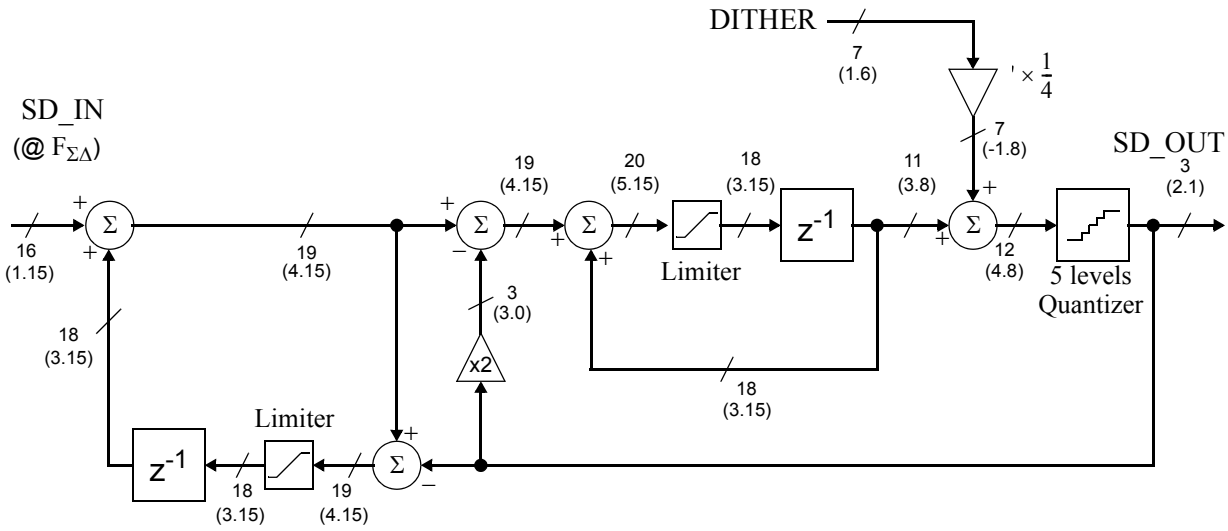


Figure 25-26. ΣΔ Modulator Hardware Implementation Scheme

Figure 25-27 shows the spectrum of the DAC Sigma-Delta output with a full scale - 6 dB 1 kHz sine wave input signal. The figure shows the signal tone with the second order shaped quantization error noise.

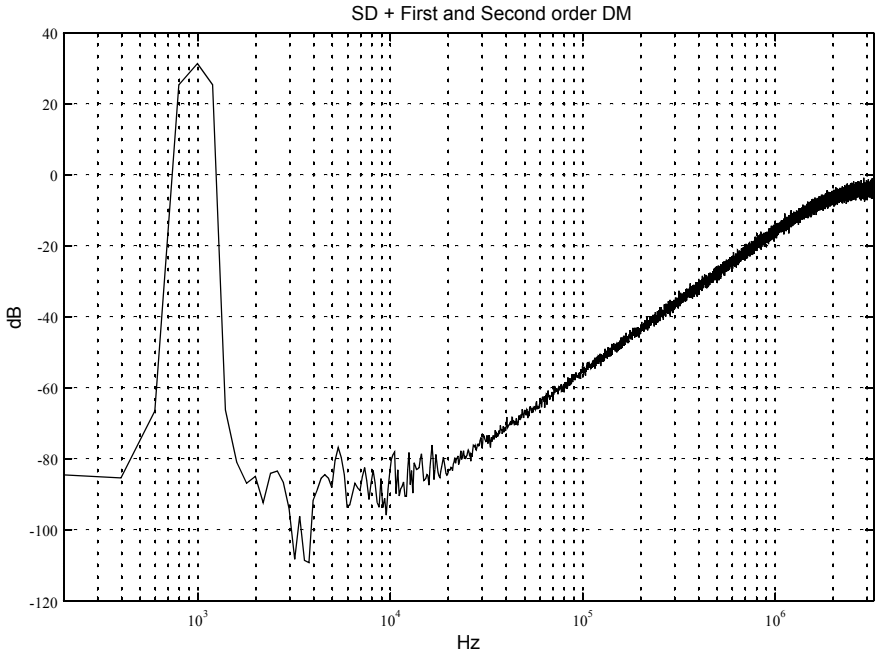


Figure 25-27. Spectrum at DAC Sigma-Delta Output (with half full scale 1 kHz sine wave signal)

25.4.2.6 Dynamic Matching

Figure 25-28 on page 25-46 shows the block diagram of the voice codec DAC path Dynamic Matching module. The digital-to-analog conversion is performed with four identical current sources that are enabled or disabled according to the sigma-delta output. Because of possible gain mismatch errors of up to 15% between the current sources, the specific current source selection is performed by a Dynamic Matching

module. The Dynamic Matching module action is to shape the noise resulting from current sources mismatch into a second order shape. Figure 25-29 shows the current sources mismatch noise after second order dynamic matching for an extreme condition of 15% mismatch.

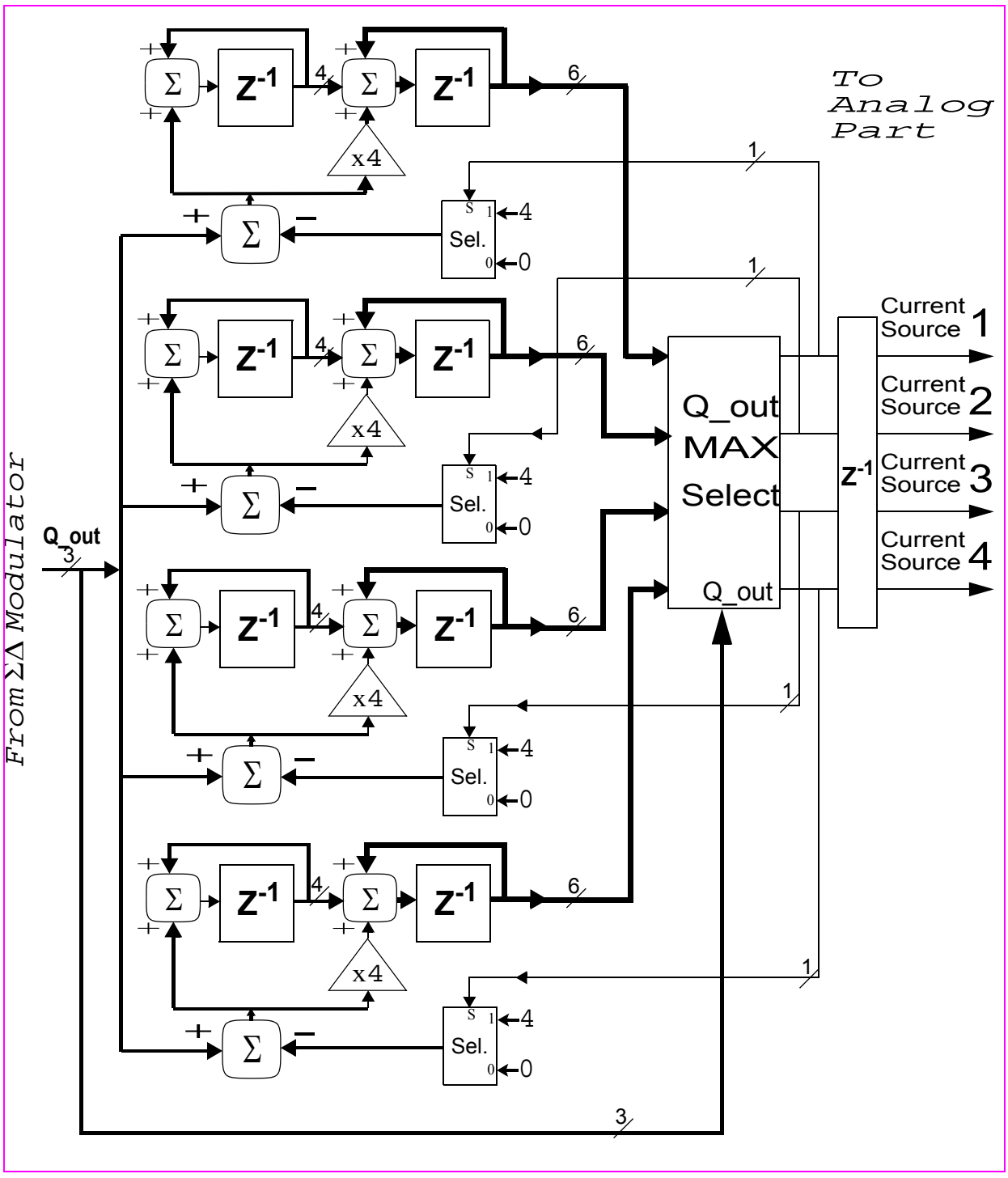


Figure 25-28. Voice Codec DAC Dynamic Matching Module

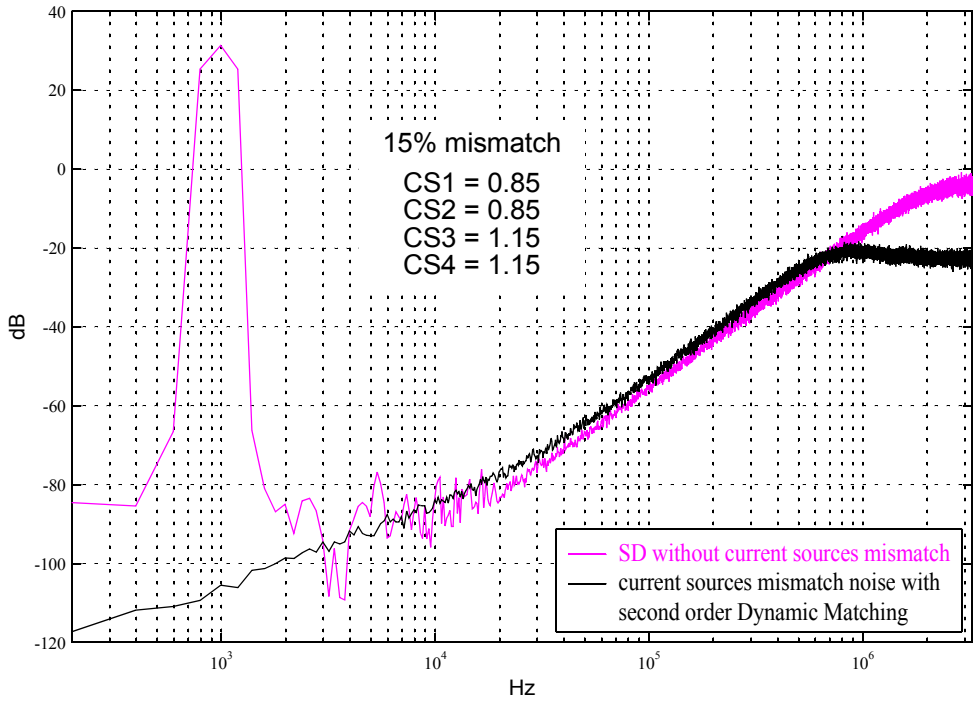


Figure 25-29. Current Sources Mismatch Noise After Second Order Dynamic Matching (Compared to Sigma - Delta Output)

25.4.2.7 DAC Analog Part

D/A contains a fourth order analog low-pass filter with a cut-off frequency 50KHz typical. A differential output buffer sets the necessary output voltage.

25.5 Voice Codec Electrical Specifications

The voice codec function is analog-to-digital and digital to analog conversion of the voice signal. The following section contains detailed electrical specifications for the analog and digital parts performance. The voice codec is powered down when not enabled for power consumption saving.

NOTE:

Most of the requirements are identical to the voice codec requirements in the GCAP3 chip.

Table 25-26 shows the voice codec general specifications.

Table 25-26. Voice Codec General Specifications

Parameter	Condition	Min	Typ	Max	Units
CODEC Input clock CODEC_CLK	VCLK[2:0]=0,1 VCLK[2:0]=2,5,6 VCLK[2:0]=3,4 VCLK[2:0]=7		26 16.8 19.44 TBD		MHz MHz MHz MHz
VAG input Voltage	No Load, AVDD (REG_BYP_CODEC) = 2.5V	1.225	1.325	1.425	V
Ref_Codec_p		1.49625	1.575	1.65375	V
Ref_Codec_n		0.836	0.88	0.924	V
VAG External Cap		0.01	0.1		µF
CD_CAP External Cap		0.02	0.1		µF
CODEC Analog Supply Current (includes Rx and Tx paths)	AVDD (REG_BYP_CODEC) = 2.5V, operational		5	6	mA
	power-down mode			5	µA
CODEC Digital Supply Current[1]	operational mode			1	mA
Response to input ON/OFF (settling time at turn on)				1	mS

1.More accurate estimation will be given after some progress in design.

25.5.1 Voice Codec ADC Specifications

Voice coding function includes a 50 kHz second order, low-pass anti aliasing filter, an analog-to-digital converter, digital filters for decimation, band-passing, frequency ripple compensation and DSP interface logic. The audio input A/D converter converts the incoming signal to 13-bit 2s compliment linear PCM words at an 8 or 8.1 kHz rate. Following the A/D converter, the signal is digitally filtered, low-pass and selectable high-pass. Table 25-27 shows the voice coding specifications.

Table 25-27. Voice Codec ADC Specifications[1]

Parameter	Condition	Min	Typ	Max	Units
VAG Noise Rejection	20Hz to 100kHz, with 100 mV_{pp} noise applied to VAG input	50	60		dB
Power Supply Rejection Ratio with respect to AVDD (REG_BYP_CODEC)[2]	20Hz to 100kHz, with 100 mV_{pp} noise applied to AVDD, with an external VAG cap	50	60		dB
Peak Input	(+3dBm0)[3] on an individual differential pin (ADC_P or ADC_M)	VAG-0.34		VAG+0.34	V
Tx AC Input Impedance	f=1.02kHz	100			kΩ
Absolute Gain	0dBm0@1.02kHz	-1.0		1.0	dB
Gain vs. Signal	Relative to -10dBm0 @1.02kHz +3 to -40dBm0 -40 to -50dBm0 -50 to -55dBm0	 -0.25 -1.2 -1.3		 0.25 1.2 1.3	 dB dB dB

Table 25-27. Voice Codec ADC Specifications[1] (Continued)

Parameter	Condition	Min	Typ	Max	Units
Total Distortion (noise and harmonic) (300 Hz - 20 kHz Noise BW in 300 Hz - 4kHz measured BW out)	1.02kHz tone (linear) +2dBm0[4] 0dBm0 -6dBm0 -10dBm0 -20dBm0 -30dBm0 -40dBm0 -45dBm0 -55dBm0	 57 60 60 55 45 35 25 20 15	 60 64 70 65 55 45 35 30 20		 dB dB dB dB dB dB dB dB dB
Idle Channel Noise[5]	Psophometric Weighting at the output			-72	dBm0p
Digital Offset[6]				5	%Full Scale
Frequency Response VCIHPF = logic high	Relative to 0dBm0@1.02kHz 50Hz 60Hz[7] 200Hz 300 to 3000Hz 3400Hz[8] 4000Hz 4600Hz	 -8 -0.5 -1.0	 	 -25 -23 -0.5 +0.5 +0.1 -14 -35	 dB dB dB dB dB dB dB
Frequency Response VCIHPF=logic low	Relative to 0dBm0@1.02kHz 50Hz 200Hz 300 to 3000Hz 3400Hz[9] 4000Hz 4600Hz	 -0.5 -0.5 -0.5 -1.0	 	 +0.5 +0.5 +0.5 +0.1 -14 -35	 dB dB dB dB dB dB
Inband Spurious	1.02kHz @ 0dBm0, 300 to 3kHz			-48	dB
Crosstalk D/A to A/D	D/A = 0 dBm0 @1.02kHz Measured while stimulated w/ 2667Hz @-50dBm0			-75	dB
Intermodulation Distortion	Two frequencies of amplitudes -4 to -21 dBm0 from the range 300 to 3400Hz			-41	dB
Filter Group Delay VCIHPF=logic high CODEC_CLK=26MHz (Relative to 1.6kHz)	500Hz<f<600Hz 600Hz<f<800Hz 800Hz<f<1kHz 1kHz<f<1.6kHz 1.6kHz<f<2.6kHz 2.6kHz<f<2.8kHz 2.8kHz<f<3.0kHz			260 155 57 15 95 135 190	µS µS µS µS µS µS µS
Filter Group Delay VCIHPF=logic low CODEC_CLK=26MHz (Relative to 1.6kHz)	f<1.6kHz 1.6kHz<f<2.6kHz 2.6kHz<f<2.8kHz 2.8kHz<f<3.0kHz	-40 0		0 100 150 200	µS µS µS µS

Table 25-27. Voice Codec ADC Specifications[1] (Continued)

Parameter	Condition	Min	Typ	Max	Units
Filter Absolute Group Delay VCIHPF=logic high	f=1.6kHz			300	µS
Filter Absolute Group Delay VCIHPF=logic low	f=1.6kHz			235	µS
Out of Band input fold-in spurious	with 0dBm0 input signal from 4.6 kHz to 8.4 kHz			-50	dB

1.All analog signals are referenced to VAG unless otherwise noted.

2.Power Supply Rejection Ratio is for Neptune IC only. Total PSRR from battery to output is obtained by summing the PSRR from Neptune to the one from the Regulator in Seaweed. It is assumed that the regulators in Seaweed will have a minimum PSRR of 45 dB.

3.For A/D differential input (ADC_P - ADC_M) 0dBm0 = 340mV$_{rms}$.

The CODEC output will not “foldback” or oscillate if overdriven, but clip.

4.The digital word corresponding to +3dBm0 is ‘0111111111111’B. Therefore if the audio level is set to +3dBm0, any variation in gain could cause large distortion if the digital number exceeds ‘0111111111111’B. For this reason the maximum recommended signal for low distortion is +3dBm0- (Absolute Gain Error) = +2dBm0.

5.GSM Spec = -64 0dB.

6.This value is a preliminary target. The final number will be specified after obtaining the production statistical data.

7.Small frequency response deviation from straight line in the 60:200 Hz range is acceptable by spec requirements.

8.Small frequency response deviation from straight line in the 3400:4000 Hz range is acceptable by spec requirements.

9.Small frequency response deviation from straight line in the 3400:4000 Hz range is acceptable by spec requirements.

Figure 25-30 and Figure 25-31 show the filter frequency response for the audio signal for voice coding path. (Note that all filter frequencies increase by 8.1/8.0 if VCLK is selected to generate FSYNC=8.1kHz).

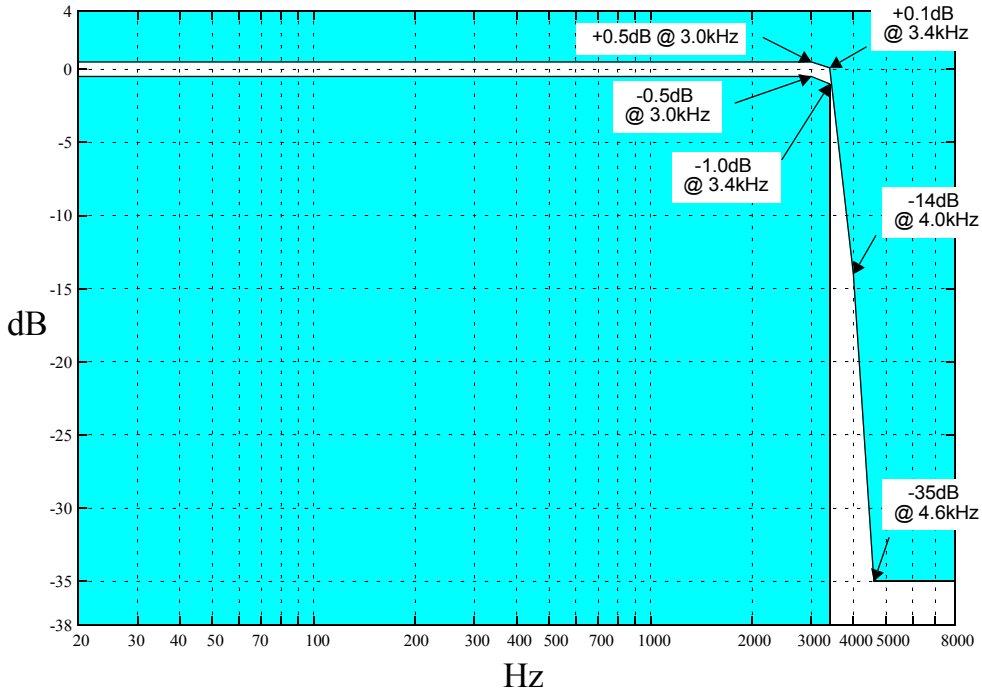


Figure 25-30. Voice Signal Frequency Response Requirements at the ADC Path (VCIHPF=0, LPF Alone Without HPF)

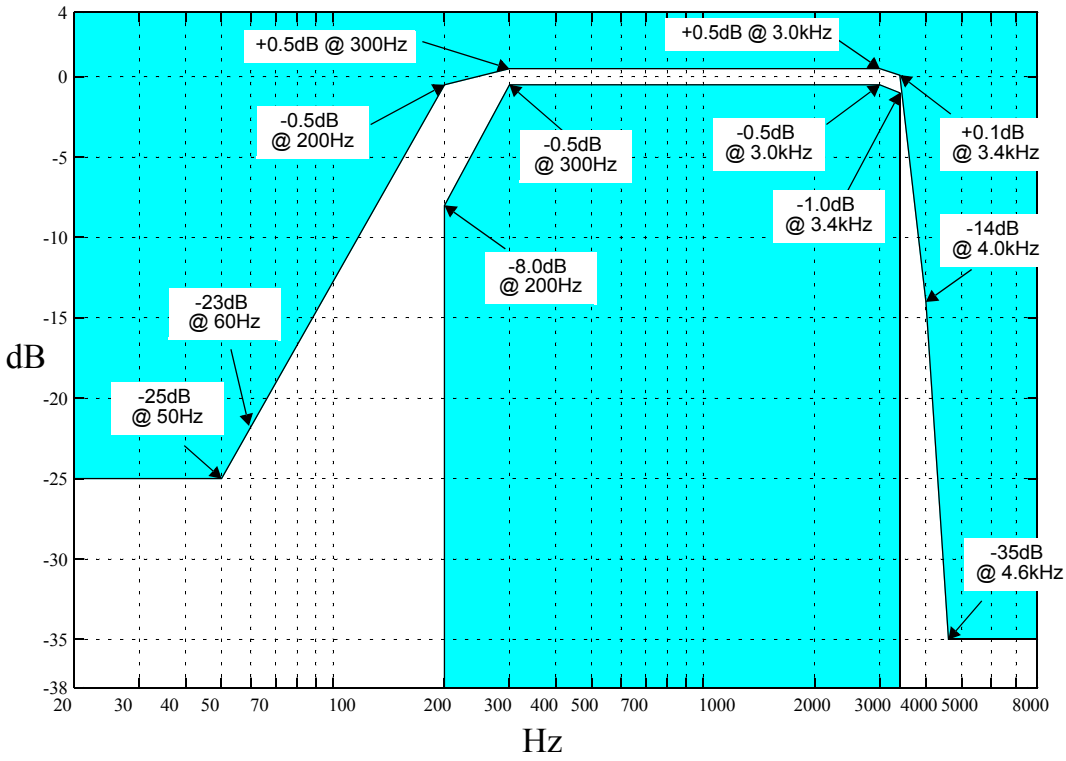


Figure 25-31. Voice Signal Frequency Response Requirements at the ADC Path (VCIHPF=1, HPF and LPF Together)

25.5.2 Voice Codec DAC Specifications

Voice decoding function includes frequency ripple compensation, interpolation, digital to analog conversion, and anti-imaging filter. The input signal for the voice decoding function is in linear 16-bit 2's compliment PCM words at an 8 kHz or 8.1 kHz rate. Table 25-28 shows the voice decoding specifications.

Table 25-28. Voice Codec DAC Specifications[1]

Parameter	Condition	Min	Typ	Max	Units
Output Level	+3dbm0[2] (clipping level) on an individual differential output pin (DAC_P or DAC_M)	VAG-0.5		VAG+0.5	V
Output Source Impedance	10kΩ Load		100		Ω
Output Power Supply Rejection Ratio	20Hz to 100kHz with 100 mVrms, noise applied to AVDD (REG_BYP_CODEC)	50	60		dBa
Absolute Gain	0dBm0@1.02kHz	-1.0		1.0	dB
Gain vs. Signal	-10dBm0@1.02kHz +3 to -40dBm0 -40 to -50dB -50 to -55dBm0	 -0.25 -1.2 -1.3		 0.25 1.2 1.3	 dB dB dB
Total Noise and Distortion (4kHz noise BW in 300 Hz - 20 kHz measured BW out)	1.02kHz tone (linear) +2dBm0 0dBm0 -6dBm0 -10dBm0 -20dBm0 -30dBm0 -40dBm0 -45dBm0 -55dBm0	 79 77 71 67 57 47 37 32 22	 85 83 77 73 63 53 43 38 28		 dB dB dB dB dB dB dB dB dB
Idle Channel Noise[3] (At CODEC out)	A weighted to 20kHz Neptune LTE		-78	-73	dBm0
	8kHz, 30Hz BW, D/A = zero code	No spurious			
Differential offset	T_A = 70 °C T_A = 25 °C			40 30	mV
Frequency Response VCOHPF = logic high (Min. limit valid for CODEC_CLK=26MHz)	Relative to 0dBm0@1.02kHz 50Hz 60Hz[4] 200Hz 300 to 3000Hz 3400Hz[5] 4000Hz 4600Hz	 -8 -0.5 -0.8		 -25 -23 -0.5 +0.5 +0.1 -14 -35	 dB dB dB dB dB dB dB

Table 25-28. Voice Codec DAC Specifications[1] (Continued)

Parameter	Condition	Min	Typ	Max	Units
Frequency Response VCOHPF = logic low (Min. limit valid for CODEC_CLK=26MHz)	Relative to 0dBm0@1.02kHz 50Hz 200Hz 300 to 3000Hz 3400Hz[6] 4000Hz 4600Hz	 -0.5 -0.5 -0.5 -0.8		 +0.5 +0.5 +0.5 +0.1 -14 -35	 dB dB dB dB dB dB
Inband Spurious	1.02kHz @ 0dBm0, 300 to 3kHz			-48	dB
Out-of-Band Spurious (Interpolation Image Suppression)	300 to 3400Hz @ 0dBm0 input 4600 to 7600Hz 7600 to 8400Hz 8400 to 20,000Hz			 -50 -50 -50	 dB dB dB
Crosstalk A/D to D/A	A/D = 0dBm0 @1.02kHz			-75	dB
Intermodulation Distortion	Two frequencies. of amplitudes -4 to -21 dBm0 from the range 300 to 3400Hz			-41	dB
Filter Group Delay VCOHPF = logic high CODEC_CLK=26 MHz (Relative to 1.6kHz)	500Hz<f<600Hz 600Hz<f<800Hz 800Hz<f<1kHz 1kHz<f<1.6kHz 1.6kHz<f<2.6kHz 2.6kHz<f<2.8kHz 2.8kHz<f<3.0kHz			300 200 70 30 95 135 190	µS µS µS mS µS µS µS
Filter Group Delay VCOHPF = logic low CODEC_CLK=26 MHz (Relative to 1.6kHz)	f<1.6kHz 1.6kHz<f<2.6kHz 2.6kHz<f<2.8kHz 2.8kHz<f<3.0kHz	-40 0		0 100 160 200	µS µS µS µS
Filter Absolute Group Delay VCOHPF = logic high	f=1.6kHz			350	µS
Filter Absolute Group Delay VCOHPF = logic low	f=1.6kHz			320	µS

1.All analog signals are referenced to VAG unless otherwise noted. Output is 0dbm0 unless noted.

2.For D/A differential output (DAC_P - DAC_M) 0dBm0 = 500 mV_{rms}.

3.GSM Spec = -64.

4.Small frequency response deviation from straight line in the 60:200 Hz range is acceptable by spec requirements.

5.Small frequency response deviation from straight line in the 3400:4000 Hz range is acceptable by spec requirements.

6.Small frequency response deviation from straight line in the 3400:4000 Hz range is acceptable by spec requirements.

Figure 25-32 and Figure 25-33 on page 25-54 show the filter frequency response for the audio signal for both voice decoding. The requirements for the decoding path at 3.4 kHz are slightly different from the coding path. (Note that all filter frequencies increase by 8.1/8.0 if VCLK is selected to generate FSYNC = 8.1 kHz).

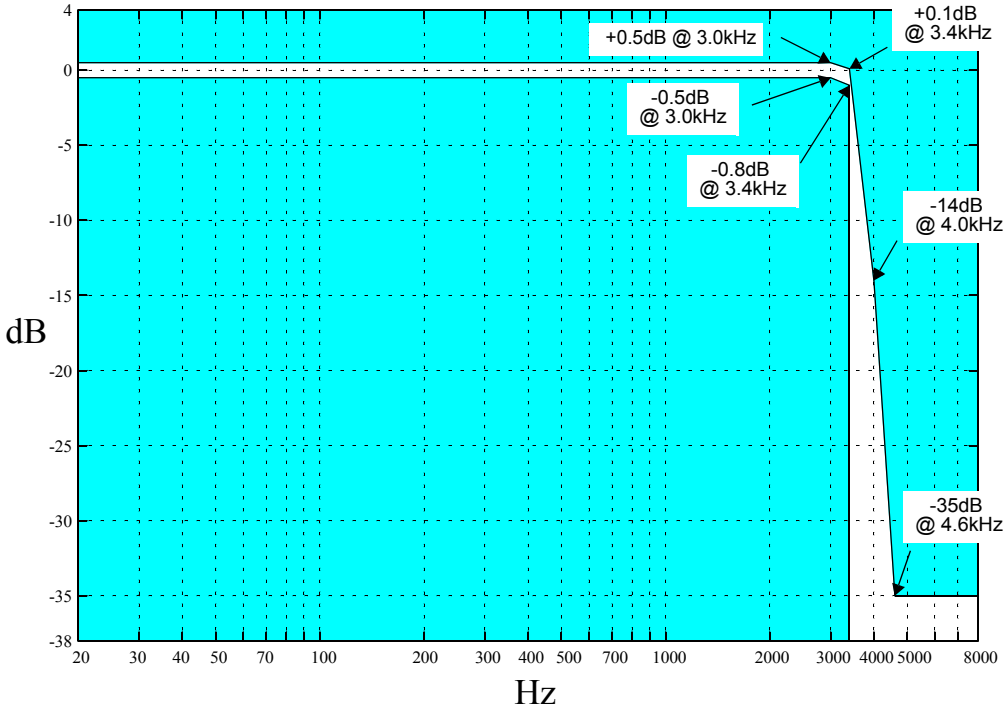


Figure 25-32. Voice Signal Frequency Response Requirements at the DAC Path (VCOHPF=0, LPF Alone Without HPF)

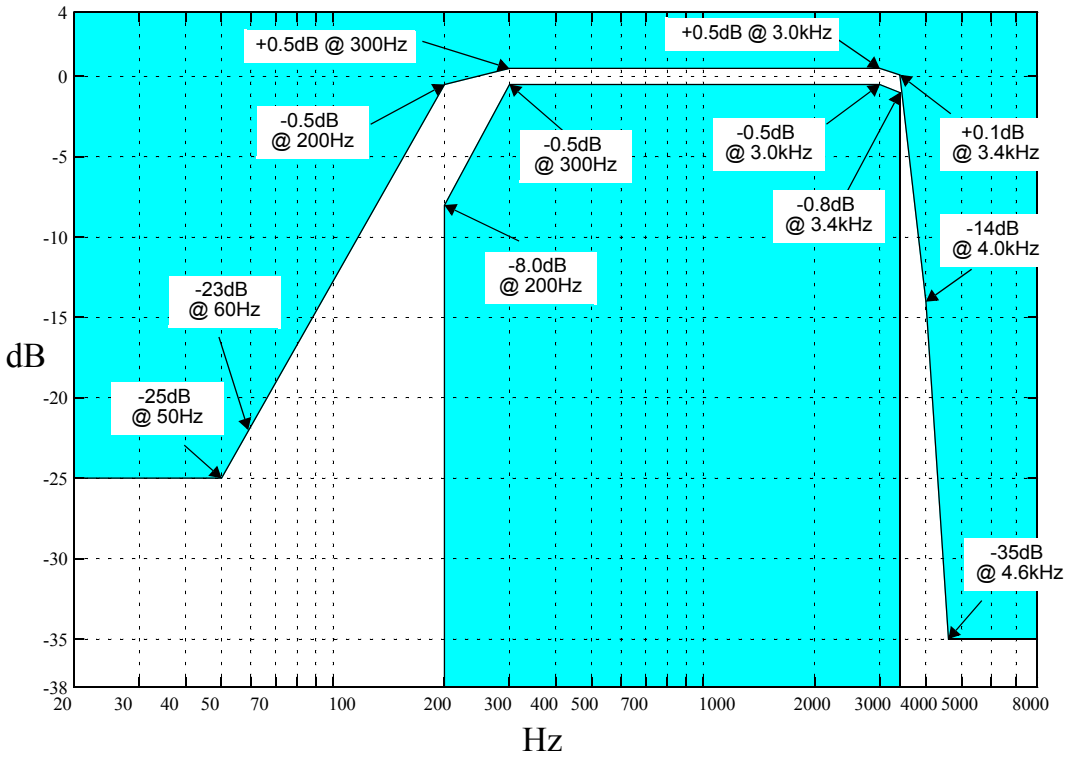


Figure 25-33. Voice Signal Frequency Response Requirements at the DAC Path (VCOHPF=1, HPF and LPF Together)

25.6 Voice Codec Testing

25.6.1 Characterization Tests

The characterization tests will be based on the Wally test bench with some additional tests for the new parameters. Details TBD.

25.6.2 Production Tests

25.6.2.1 Digital Parts Tests

The testing of the digital portion will be based on scan chain test for the major part of the logic (95%). In addition to this the DSP core interfaces will be tested with a dedicated functional tests. Also the digital portion functionality will be verified during the measurement of the analog parameters.

25.6.2.2 Analog Parts Tests

The tests of the voice codec analog parts will be based on the test bench developed for Wally with some additional tests for the new parameters. The detailed description is TBD.

25.6.2.3 Output to ANATEST module

There is one differential output for the analog testing, generated by the DAC module. The signals are connect to the analog test module ANATEST and can be viewed on the analog test output pads. Currently the ATST_P and ATST_N are connected to the pre amplified output of the DAC. The analog test is enabled when ANATEST_VOC[0] is set to high. For normal operation both ANATEST_VOC[1:0] should be set to low. (Note that in the current design ANATEST_VOC[1] do not have any function and doesn't influence the ANATEST output.) The ANATEST_VOC[1:0] signals correspond to VOC[1:0] bits in the ANATEST control register.

25.6.3 Backup plan

A backup plan in case the module is non operational is to use the SAP module for a connection to an external codec.

25.7 REFERENCES

[1] Shuni C. and C. Sidney Burrus, "Multirate Filter Designs Using Comb Filters", IEEE Trans. Circuits and Sys., vol. CAS-31, pp.913-924, November 1984.

[2] S.R. Norsworthy, R. Schreier and G.C. Temes, "Delta-Sigma Data Converters: Theory, Design, and Simulation", IEEE Press, 1997, IEEE Order No. PC3954.

[3] J.C. Candy and G.C. Temes, "Oversampling Delta-Sigma Data Converters: Theory, Design and Simulation", IEEE Press,1992, IEEE Order No. PC2741.

Chapter 26
Receiver Analog Front End (RxAFE)

Revision History Table

Revision	Date	Author	Changes
0.0	03/25/02	Chunhe Zhao	Initial Release : Copied from LT rxafe spec rev1.2; Updated per DDTS DSPhl13832
0.1	07/19/02	Chunhe Zhao Shannon Osgood	Updated per DDTS# DSPhl14224/DSPhl14450.
0.2	11/12/02	Giovanni Angello	Updated per DDTS# DSPhl15697.
0.3	04/30/03	Kiyoshi Kase, Shannon Osgood	Updated per DDTS# DSPhl16065.
0.4	05/07/03		Updated for LTE specification release.

26.1 Introduction

RxAFE converts the baseband or very low IF received signal from analog to digital. RxAFE consists of two sub-blocks: the RX Clock generator and fourth order sigma delta modulator Baseband Receive I/Q A/Ds.

NOTE:

The Neptune LTE chips has an option to provide a corrected or uncorrected clock programmable through the Group 5 SPI bit RXCLK_SEL. The analog top provides the switch to select the clock outside of the RXAFE module.

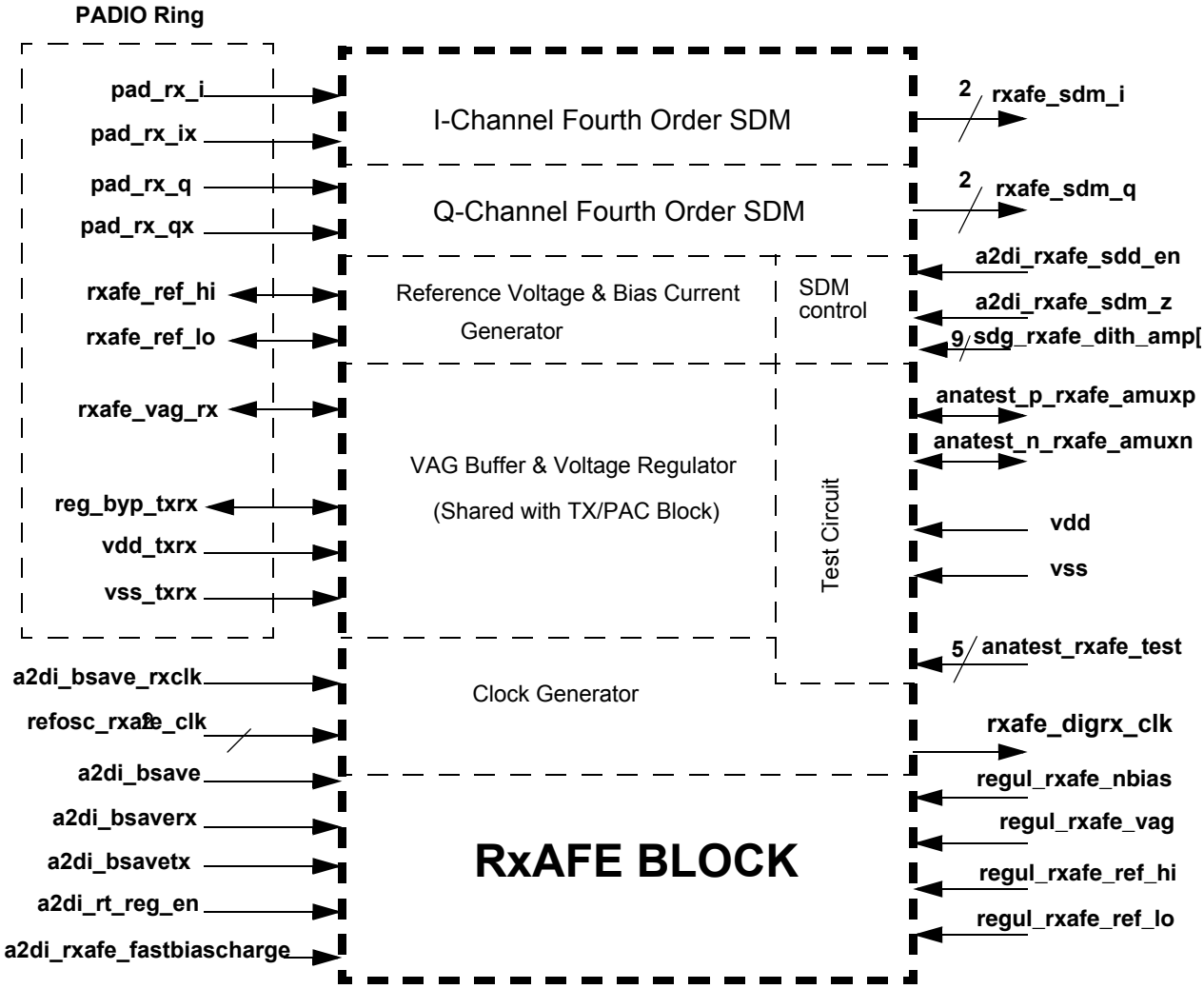


Figure 26-1. RxAFE Block Diagram

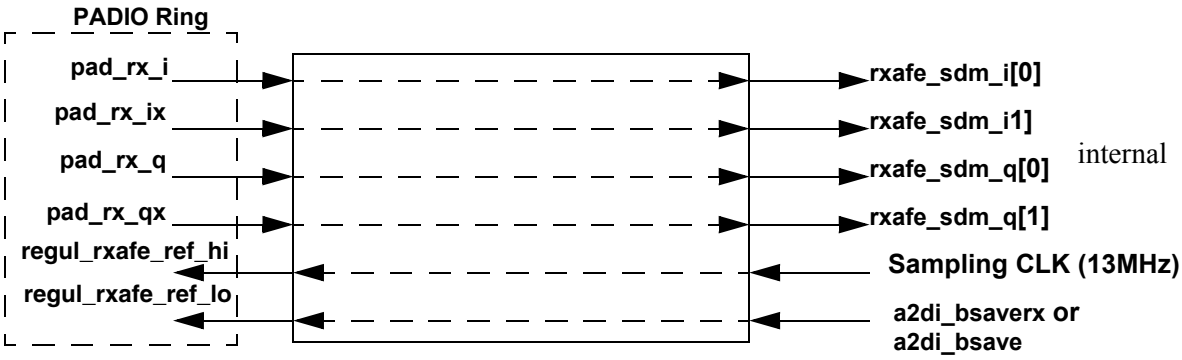


Figure 26-2. Bypass Test Mode Connection

26.2 RxAFE Module Pin List

Table 26-1. RxAFE Module Pin List

Pin Name	Direction	Description	From/To	Type
pad_rx_i	input	Receiver I-channel positive input	PADIO	analog
pad_rx_ix	input	Receiver I-channel negative input	PADIO	analog
pad_rx_q	Input	Receiver Q-channel positive input	PADIO	analog
pad_rx_qx	input	Receiver Q-channel negative input	PADIO	analog
rxafe_ref_hi	inout	Receiver ADC High level reference decoupling	PADIO	analog
rxafe_ref_lo	inout	Receiver ADC Low level reference decoupling	PADIO	analog
refosc_rxafe_clk[1]	input	Differential 26 MHz reference clock - non inverting input	REFOSC	digital
refosc_rxafe_clk[0]	input	Differential 26 MHz reference clock - inverting input	REFOSC	digital
a2di_bsave_rxclk	input	13MHz reference clock gate signal	A2DIGL	digital
rxafe_digrx_clk	output	13 MHz reference gated clock (ph2 of sdm_clk)	RXCPROC	digital
rxafe_vag	output	Receiver VAG buffered output decoupling	PADIO	analog
reg_byp_txrx	input	Receiver internal 2.5 V±5% voltage regulator decoupling	PADIO	analog
vdd_txrx	input	Receiver 2.775 V ± 3.6% analog power supply	PADIO	power supply
vss_txrx	input	Receiver 0.0 V analog ground supply	PADIO	power supply

Table 26-1. RxAFE Module Pin List

Pin Name	Direction	Description	From/To	Type
rxafe_sdm_i[1]	output	Receiver I-channel modulator output (from CMP4)	RXCPROC	digital
rxafe_sdm_i[0]	output	Receiver I-channel modulator output (from CMP2)	RXCPROC	digital
rxafe_sdm_q[1]	output	Receiver Q-channel modulator output (from CMP4)	RXCPROC	digital
rxafe_sdm_q[0]	output	Receiver Q-channel modulator output (from CMP2)	RXCPROC	digital
rxafe_anatest_p	inout	Receiver analog positive test	ANATEST	analog
rxafe_anatest_n	inout	Receiver analog negative test	ANATEST	analog
vdd	input	Core VDD 1.575 V digital power supply		power supply
vss	input	Core VSS digital ground supply		power supply
anatest_rxafe_test_x_[4:0]	input	Receiver test mode input	ANATEST	digital
a2di_bsave	input	Receiver Power Save Mode	A2DIGL	digital
a2di_bsaverx	input	Receiver SDM Power Save Mode	A2DIGL	digital
a2di_bsavetx	input	Transmitter Power Save Mode	A2DIGL	digital
a2di_rt_reg_en	input	RxTx regulator enable	A2DIGL	digital
a2di_rxafe_fastbiascharge	input	Fast Bias Charge	A2DIGL	digital
a2di_rxafe_sdm_z	input	Sigma Delta Modulator Zeroing (defaulted to one)	A2DIGL	digital
a2di_rxafe_sdd_en	input	Sigma Delta Modulator Dither Enable	A2DIGL	digital
sdg_rxafe_dith_amp_x_[8:0]	input	Receiver Dither level	RxSDG	digital
regul_rxafe_nbias	input	NMOS Reference current	REGUL	analog
regul_rxafe_vag	input	From REGUL feedback resistor	REGUL	analog
regul_rxafe_ref_hi	input	From REGUL feedback resistor	REGUL	analog
regul_rxafe_ref_lo	input	From REGUL feedback resistor	REGUL	analog

26.3 RxAFE Test Modes

The test mode signal (TEST[4:0]) can be set for testing or evaluating the RxAFE ADC block. In the normal mode of operation, these bits should be kept at low level (VSS_TXRX). a2di_bsaverx may be set or reset at any time whenever needed. If any one of the TEST[3:0] bits become set to 1, RxAFE ADC should be turned ON regardless of the status of the BSAVERX bits, except for the case where all TEST[3:0] bits are set to 1. In this case, Bypass Mode is entered, and an SDM power down occurs.

Table 26-2. RxAFE Test Mode RXAFE_AMUXP/N Output List

BSAVERX	TEST[4]	TEST[3]	TEST[2]	TEST[1]	TEST[0]	Output Description
0	x	0	0	0	0	**Normal operating mode**
x	x	1	1	1	1	**Bypass Mode (only SDM analog circuit power down—run clock generator)**
1	x	0	0	0	0	**RX BSave mode—SDM is power down & stop clock generator**
x	0	x	x	x	x	I channel select
x	1	x	x	x	x	Q Channel select
x	0/1	0	0	0	1	(Reserved)
x	0/1	0	0	1	0	I or Q-ch First integrator output
x	0/1	0	0	1	1	I or Q-ch Second integrator output
x	0/1	0	1	0	0	I or Q-ch Third integrator output
x	0/1	0	1	0	1	I or Q-ch Fourth integrator output
x	0/1	0	1	1	0	I or Q-ch Comparator output
x	0/1	0	1	1	1	I or Q-ch First/ Second integrator bias current input
x	0/1	1	0	0	0	I or Q-ch Third/ Forth integrator bias current input
x	0/1	1	0	0	1	I or Q-ch comparator bias current input
x	0	1	0	1	0	REF_HI/REF_LO gate drive input
x	1	1	0	1	0	VAG_RX gate & bias current input
x	0/1	1	0	1	1	Phi1, Phi1A/Phi2, Phi2A clock output (Use a differential stage as the input differential clock buffer)
x	0/1	1	1	0	0	Phi1, Phi2/Phi1A, Phi2A clock output
x	0/1	1	1	0	1	SDM current bias up, up (dith_amp[0], dith_amp[1])/down, down (dith_amp[2], dith_amp[3]) 15%
x	0/1	1	1	1	0	SDM current bias up, up/down, down 30%

26.4 TXRX Voltage Regulator

The voltage regulator is shared with the transmitter analog block. The specification of the voltage regulator can be found in the specification of the REGUL block.

Table 26-3. TXRX Voltage Regulator Power Down Control

RT_REG_EN	FastBiasCharge	BSAVE	BSAVETX	BSAVERX	TEST [3:0]	Mode
0	x	x	x	x	x	Power Down
1	x	x	x	x	x	Normal Bias Current

26.5 RxAFE VAG Buffer

The RxAFE VAG Buffer is shared with the transmitter analog block (TX and PAC). The control circuit opamp bias is controlled by the RxAFE Power Save mode decoder.

Table 26-4. RxAFE VAG Buffer Specifications

Specification	Min	Typ	Max	Unit
Power Supply Rejection Ratio from DC to 13 MHz C = 1.0μF	45	50		dB
Output voltage	1.15	1.25	1.35	V
Output decoupling capacitance (ESR < 0.5 Ω)	0.47	1.0	2.2	μF
Source or Sink Output Current	-300	0	300	μA
Operating Current (High bias mode)			500	μA
Operating Current (Normal bias mode)			100	μA
Standby Current (Low bias mode)			10	μA
Power Down Current			3	μA

Table 26-5. RxAFE VAG Buffer Power Save Control

RT_REG_EN	FastBiasCharge	BSAVE	BSAVETX	BSAVERX	TEST [3:0]	Mode
0	x	x	x	x	Normal	Power Down
1	0	1	x	x	Normal	Low Bias Current
1	0	0	1	1	Normal	Low Bias Current
1	0	0	0	0	Normal	Normal Bias Current
1	0	0	1	0	Normal	Normal Bias Current
1	0	0	0	1	Normal	Normal Bias Current

Table 26-5. RxAFE VAG Buffer Power Save Control

RT_REG_EN	FastBiasCharge	BSAVE	BSAVETX	BSAVERX	TEST [3:0]	Mode
1	1	x	x	x	Normal	High Bias Current
1	0	x	x	x	**Bypass**	Normal Bias Current
1	0	x	x	x	Any Test	Normal Bias Current

26.6 RxAFE Reference Voltage Generator

Table 26-6 presents RxAFE reference voltage generator details.

Table 26-6. RxAFE Reference Voltage Generator Specifications

Specification	Min	Typ	Max	Unit
Minimum Power Supply Rejection Ratio from DC to 13 MHz	45			dB
Reference voltage High refer to RX_VAG voltage (REF_HI)		0.40		V
Reference voltage Low refer to RX_VAG voltage (REF_LO)		-0.40		V
Output decoupling capacitance	0.047	0.1	0.22	µF
Source or Sink Output Current	-300	0	300	µA
Operating Current (High bias mode)			500	µA
Operating Current (Normal bias mode)			100	µA
Standby Current (Low bias mode)			10	µA
Power Down Current			3	µA

Table 26-7.

RT_REG_EN	FastBiasCharge	BSAVE	BSAVETX	BSAVERX	TEST [3:0]	Mode
0	x	x	x	x	Normal	Power Down
1	0	1	x	x	Normal	Low Bias Current
1	0	0	1	1	Normal	Low Bias Current
1	0	0	0	0	Normal	Normal Bias Current
1	0	0	1	0	Normal	Normal Bias Current
1	0	0	0	1	Normal	Normal Bias Current
1	1	0	x	x	Normal	High Bias Current

Table 26-7.

RT_REG_EN	FastBiasCharge	BSAVE	BSAVETX	BSAVERX	TEST [3:0]	Mode
1	x	x	x	x	**Bypass**	Power Down
1	0	x	x	x	Any Test	Normal Bias Current

26.7 Baseband Receive I/Q A/Ds

This section discusses specifications, timing, and dither input.

26.7.1 Specifications

Two identical analog-to-digital converters are required to convert analog baseband or very low IF analog signals to digital. The input signals come from the RF chip (Algae). They are in differential voltage signaling format. One A/D is for the I channel, and the other is for the Q channel. Table 26-8 shows the target specification for the baseband receive A/Ds analog front end, including the anti-aliasing filter.

No digital filter is included in the RxAFE. Each ADC outputs two-bit-width data from two comparators. No digital noise cancellation logic is included in the RxAFE block. These digital processes should be integrated in the RXCPROC block.

Table 26-8. Baseband Receive A/D Converters Specifications

Specification	Min	Typ	Max	Unit
Rx Standby Recovery Time			10	μS
Sampling Rate		13		MHz
Analog Supply Voltage		2.775 +/-4%		V
Analog Supply Voltage (PSRR improvement regulator output)		2.50 +/-4%		V
Standby Mode Analog Supply Current (Standby - one converter)			10	μA
Analog Supply Current (one converter)		5.6		mA
Digital Supply Voltage		1.575 +/-4%		V
Digital Supply Average Current (one converter)			1.0	mA
Digital interface signal level reference		1.575		V
Sampling Jitter (refosc_rxafe_clk jitter is not included)			250	pS
Input Impedance (single ended with respect to VAG)		15		KΩ
Differential Input Linear Range			1.0	Vpp
Differential Reference Voltage (REF_HI - REF_LO)		800 +/-4%		mVdc
I/Q Amplitude Mismatch (frequency = 50.78125 kHz)	-0.2	0	0.2	dB

Table 26-8. Baseband Receive A/D Converters Specifications (Continued)

Specification	Min	Typ	Max	Unit
I/Q Phase Mismatch (frequency = 50.78125 kHz)	-0.2	0.0	+ 0.2	Degrees
Maximum DC Offset	-5	0.0	5	mV
Gain Accuracy* (frequency = 50.78125 kHz)	-0.5	0.0	0.5	dB
S/(N+THD) (25 kHz to 205 kHz, BW = IF+/-90 kHz)	74	84		dB
Total Output Noise (25 kHz to 205 kHz, BW = IF+/-90 kHz)			70.5	μVrms
Input Referred Noise			166.3	nV/rt(Hz)
IIP2* Vin = -47.2 dBVrms (4.35 mVrms) at 3 MHz and 3.1 MHz (Differential Input) *note IIP	16			dBVrms
IIP3 Vin = -30.7 dBVrms (29 mVrms) at 800 kHz-IF and -51.9 dBVrms(2.53 mVrms) at 1600 kHz-IF (Differential Input) *note IIP	7			dBVrms
RX 1 dB Compression Point	-9.0			dBVrms
Common Mode Voltage		1.25		V
PSRR refer to Internal Regulator output DC to 200 kHz 13 MHz 26 MHz	 45 35 35	 55 45 45		dB
Common Mode Rejection Ratio (Gain-Diff out/Common Input) (Differential Output/Common Input) (Common Output/Common Input)	 34 -34 0			dB

*Note IIP: Vin levels may be raised to observe inter-modulated signals on the rxafe output.

*Note Gain Accuracy: the variation of the RxAFE gain (from the linear interpolation) across its linear input range.

26.7.2 Timing

The refosc_rxafe_clk signal is the 26 MHz differential input from REFOSC. It is internally divided by two for the sigma delta modulator, RxDigital (DCADAPT, RXCOPROC, and RxSDG). “digrx_clk” is the buffered phase 2 clock of the sigma delta modulator. refosc_rxafe_clk divider should be reset when a2di_bsave_rxclk is 1.

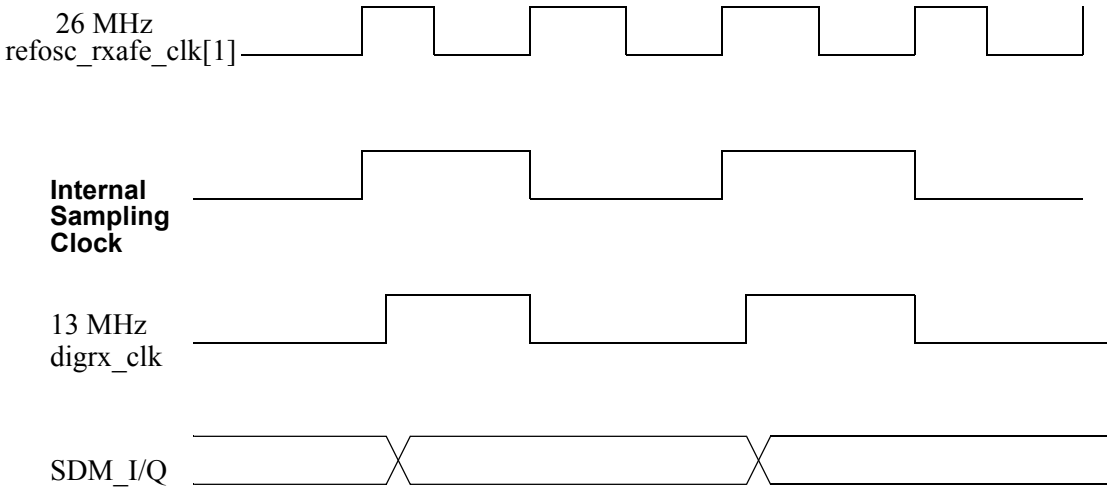


Figure 26-3. RX SDM Output Timing

26.7.3 Dither Input

DITH_AMP[8:0] controls a dither charge to the second integrator from -120 mV to +120 mV with 30 mV stepping (9 levels total, including “0”). DITH_AMP[8:0] are negative true logic. All 1 values mean no dither signal is applied to the integrator.

26.8 Baseband Receive I/Q A/Ds Block Diagram

RxAFE does not include the error correction logic consisting of the delay flip-flops and arithmetic logic to cancel the quantization noise. Figure 26-4 is a block diagram of the normalized baseband receive I/Q sigma delta modulator, which employs a fourth order 2-2 cascaded structure. The actual input level is determined by the reference voltage and input scaling factor, which is not shown in the block diagram. The typical reference voltage is 0.8 Vdc for 1.0 Vpp differential input voltage. All scaling factors in the baseband receiver I/Q A/D of the RxAFE are subject to change in order to optimize performance. However, the pass band gain and the pass band frequency response should not be changed.

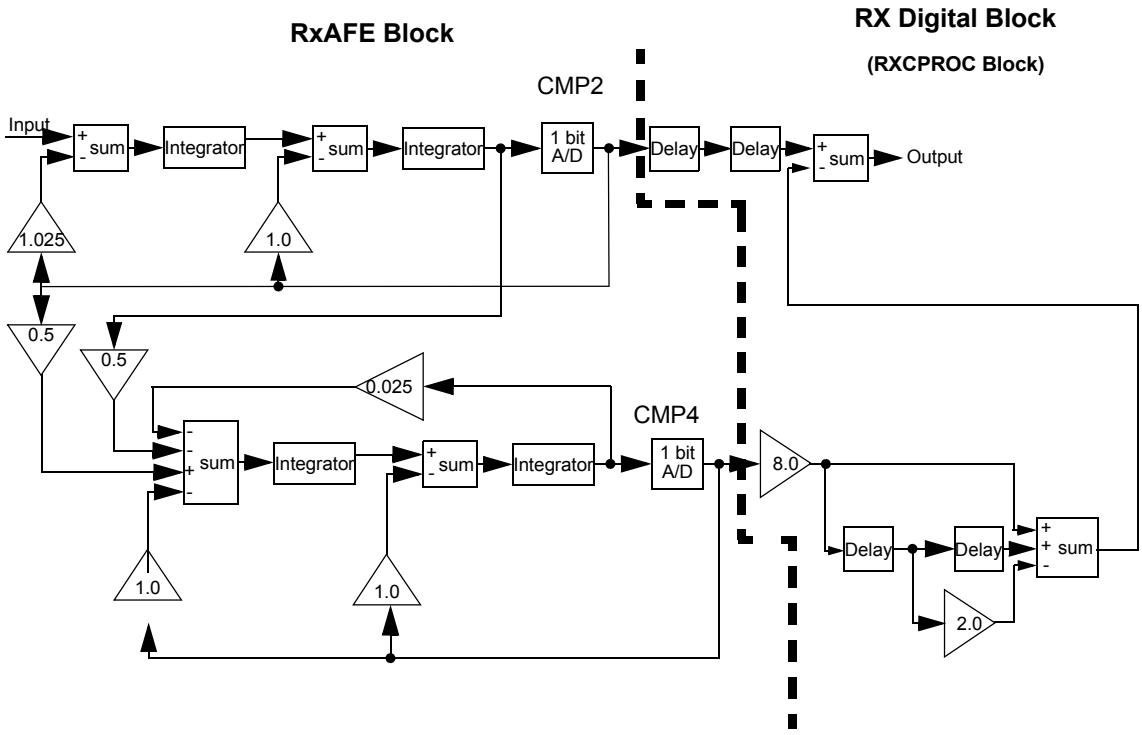


Figure 26-4. Normalized Baseband Receive I/Q Sigma Delta Modulator Block Diagram

The transfer functions of the signals are expressed in Equation 26-1 and Equation 26-2.

$$STF(z) = \frac{0.25z^{-4}}{1 - 1.5z^{-1} + 0.75625z^{-2}} \qquad \textit{Eqn. 26-1}$$

$$NTF(z) = \frac{8(1 - z^{-1})^{2}(1 - 2z^{-1} + 1.00625z^{-2})}{1 - 1.5z^{-1} + 0.75625z^{-2}} \qquad \textit{Eqn. 26-2}$$

Chapter 27
Receive Coprocessor (RxCPROC)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	03/26/02	PP	Added 2bits for phaseadb and changed GROUP3 definition to cope with these new bits.
0.2	05/02/02	PP	Changed bit definition in GROUP3 for sign and phaseadb[9:8] bits. This follows DSPhl13874.
0.3	10/18/02	PP	Aligned phaseadxb description to the A2DIGL description and corrected typos. Rework BBP interface paragraph to account for DSPhl14362/DSPhl15540.
0.4	05/07/03		Updated for LTE specification release.

27.1 Introduction

The RxCPROC includes the digital signal processing hardware required for the receive transceiver (Rx) after the initial conversion done by the sigma-delta modulator. It can be configured to be used in the direct conversion mode (DCR) or in the very low intermediate frequency mode (VLIF). The RxCPROC supports the GSM and EDGE standards.

The RxCPROC is represented by the shaded blocks called “decimation filters”, “digital IF mixer”, “digital LO” and “serial interface” in Figure 27-1. The RxCPROC decimates and filters the I and Q quadrature input signals and converts them to baseband if required. Processed signals are sent serially to the Base Band Port (BBP) to be further handled by the DSP and VIAC.

The RxCPROC is controlled via the MQSPI. The A2DIGL interface decodes the messages sent serially by the MQSPI and provides the control bits to control the behavior of the receiver.

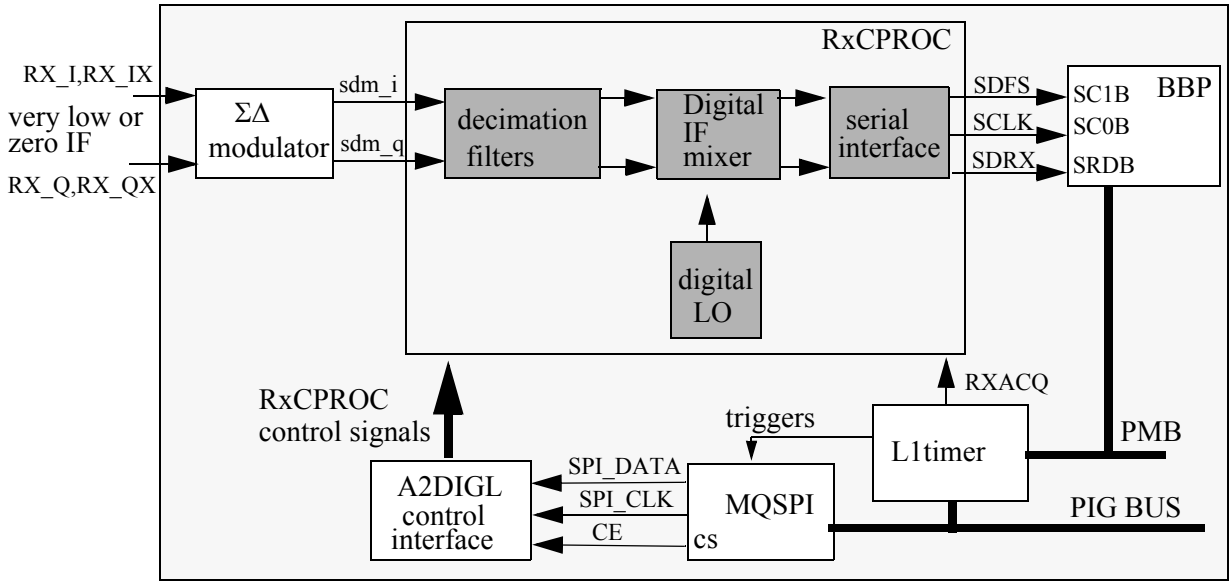


Figure 27-1. Block diagram of Rx chain in the Neptune

Feature list:

- Supports GSM and EDGE standards
- Configurable in DCR or VLIF mode
- Cancellation of 2nd order quantization noise
- Filtering of shaped quantization noise
- Compensation for in-band attenuation of analog filters
- Anti-aliasing of interferers
- Down sampling
- Quadrature IF carrier generation
- Programmable IF mixing
- Capability for calibration of image rejection
- Filtering of DC offset and part of 1/f noise
- Programmable selectivity

Chapter 27.2 presents the RxCPROC pins
Chapter 27.3 focuses on the RxCPROC architecture
Chapter 27.4 describes the SPI interface and the SPI bits used to control the RxCPROC.
Chapter 27.5 describes the serial interface to the BBP
Chapter 27.6 presents the power saving operation

For an in-depth understanding of the RxCPROC logic, the following documents are recommended:
Magicv LV IC contract book v3.4
Magic-LV Rx coprocessor v6.1
Magic-LV Rx coprocessor modification v1.0
Magic-LV Rx coprocessor Powering on and off v0.1

27.2 RxCPROC Pin List

Table 27-1 lists all the pins in the RxCPROC module.

Table 27-1. RxCPROC Pin List

Pin Name	Direction	Description
clk	Input	Main clock gated in RxAFE by BSAVErxclk
sdm_i[1:0]	Input	Channel I sigma delta output
sdm_q[1:0]	Input	Channel Q sigma delta output
rxacq	Input	From L1 timer. This signal allows SSI transmission
a2di_bsaverx	Input	Allows processor to start clocking data. Connected to a2di_bsaverxcproc in Rxdigital
a2di_ma_select[1:0]	Input	Select GSM, EDGE or DCR wide
a2di_sign[1:0]	Input	Controls four complex quadrature mixer combination
a2di_gainada[6:0]	Input	Gain adjust a for the complex quadrature mixer
a2di_gainadb[6:0]	Input	Gain adjust b for the complex quadrature mixer
a2di_dif[6:0]	Input	Digital IF local oscillator setting
a2di_phaseada[7:0]	Input	Phase adjust a for the complex quadrature mixer
a2di_phaseadb[9:0]	Input	Phase adjust b for the complex quadrature mixer
a2di_tol[1:0]	Input	Droop digital filter setting
a2di_thp1[2:0]	Input	Duration of DC adapt phase1 in DCR mode
a2di_thp2[2:0]	Input	Duration of DC adapt phase2 in DCR mode
a2di_fhp1[2:0]	Input	DC adapt setting for phase1 in DCR mode
a2di_fhp2[3:0]	Input	DC adapt setting for phase2 in DCR mode
a2di_fhp3[3:0]	Input	DC adapt setting for phase3 in DCR mode

Table 27-1. RxCPROC Pin List

Pin Name	Direction	Description
a2di_testbyp	Input	Bypass filters and mixer.
a2di_iqswap	Input	Q bits sent first if IQSWAP is high otherwise I bits are sent first
a2di_2x_sel	Input	Select the output rate
a2di_hdfo	Input	Hold DC offset value in non DCR mode
a2di_dbw[2:0]	Input	Selectivity filter setting
a2di_fbyp	Input	Selectivity filter 14th FIR bypass
a2di_bbiqlen[3:0]	Input	BBP interface setting
scan_mode	Input	Scan test mode from TCM
scan_en	Input	Scan enable
scan_rxcoproc_sdi	Input	Scan input
test_reset	Input	Reset from TCM
test_clock	Input	Scan clock from TCM
rxcp_thp1_pulse	Output	Pulse for A2DIGL module
rxcp_sdfs	Output	Serial data framing
rxcp_sclk	Output	Serial data clock
rxcp_sdrx	Output	Serial baseband I/Q data
rxcproc_sdo	Output	Scan output

Figure 27-2 shows the pinout diagram for the RxCPROC module.

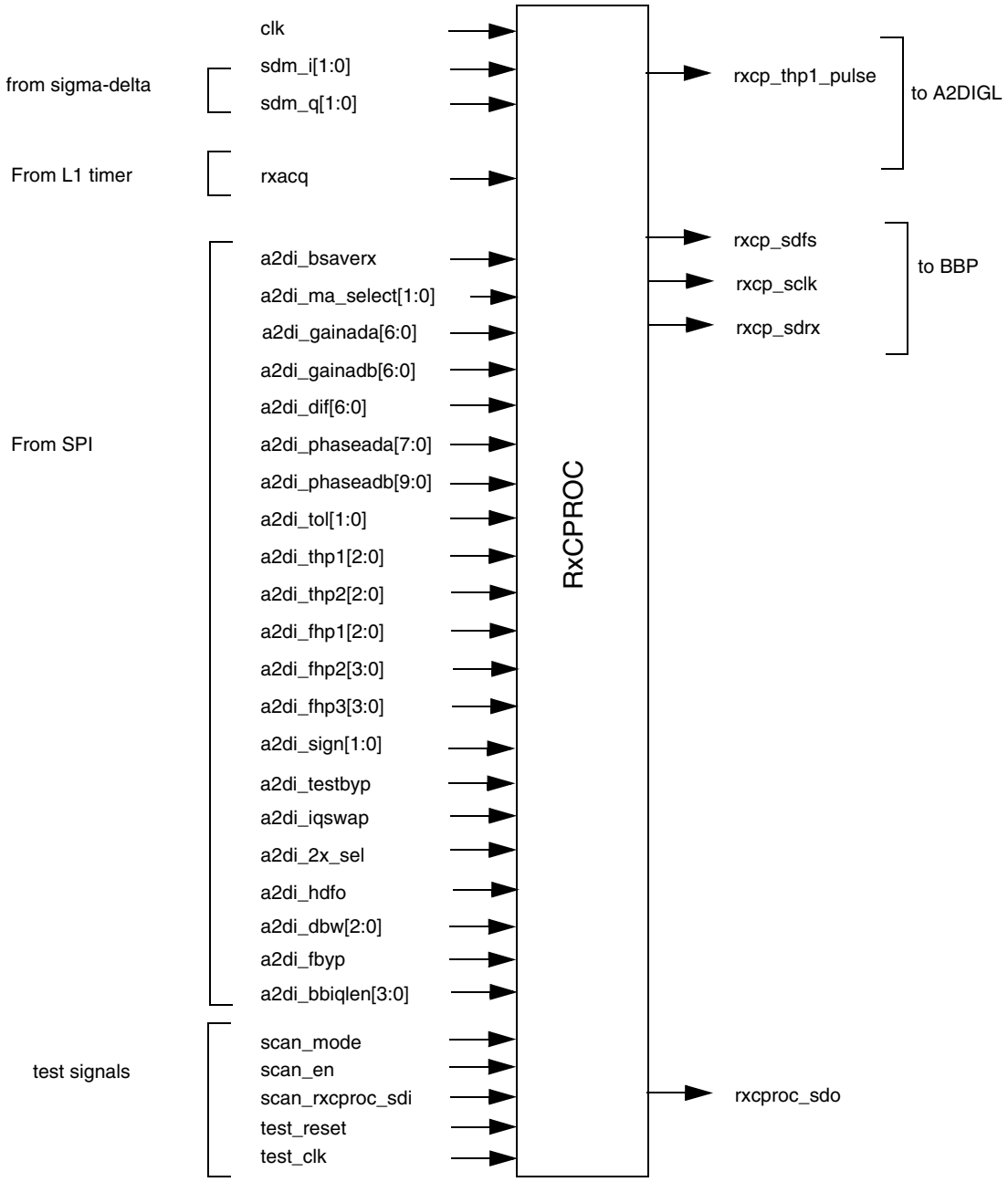


Figure 27-2. RxCPROC pinout diagram

For inputs from the SPI see Section 27.4 on page 27-16 for a complete description.

In the following, pin names will also be found in upper case without the a2di_ and rxcp_ prefixes.

27.3 RxCPROC architecture

The block diagram of the RxCPROC is given in Figure 27-3. The coprocessor supports both the GSM or EDGE standards.

Two 4th order Sigma delta are used to convert the low or Zero IF spectrum informations coming from an external chip on the RX_I/RX_IX and RX_Q/RX_QX pins of Neptune. They generate the sdm_i[1:0] and sdm_q[1:0] outputs which are supplied to the RxCPROC.
For GSM/EDGE (200kHz channel spacing) the sampling clock frequency fs is equal to 13MHz.

The output of the RxCPROC is able to deliver 16 bit I and Q baseband data output at a rate of 13MHz/48=270.833kbits/s for GSM/EDGE

27.3.1 Receiver signal processing chain

The signal processing of the received GMSK-modulated data should not introduce any corruption and then the phase response of the decimation filtering must be linear. This is achieved by using even symmetric FIR filters. The decimation process is split in 2 phases. A first low order SINC filter attenuates frequencies above f4x (fs/12 or fs/20 depending on the modes) to allow a down-sampling by 12 or 20. A second FIR filter allows a further decimation by 2, but works at a much lower rate which allows to use a reduced number of taps for its implementation. Cascading two filter-decimation stages is an optimum solution in terms of power consumption and die size.

After the SINC filter and up to the selectivity filter, data will be 17 bit wide for each input and output block. Data width is determined by signal to noise and distortion ratio (SNDR) criterion. Internal computation of each block may require some local data width extension but the LSBs at the blocks output are pruned to go down to 17 bits. As pruning generates DC offset, these truncations are allowed up to the high pass filters which remove the DC offset component. After the FIR (VLIF) or IIR (DCR) high pass filters, rounding is required instead of truncation to prevent the reintroduction of DC offset.

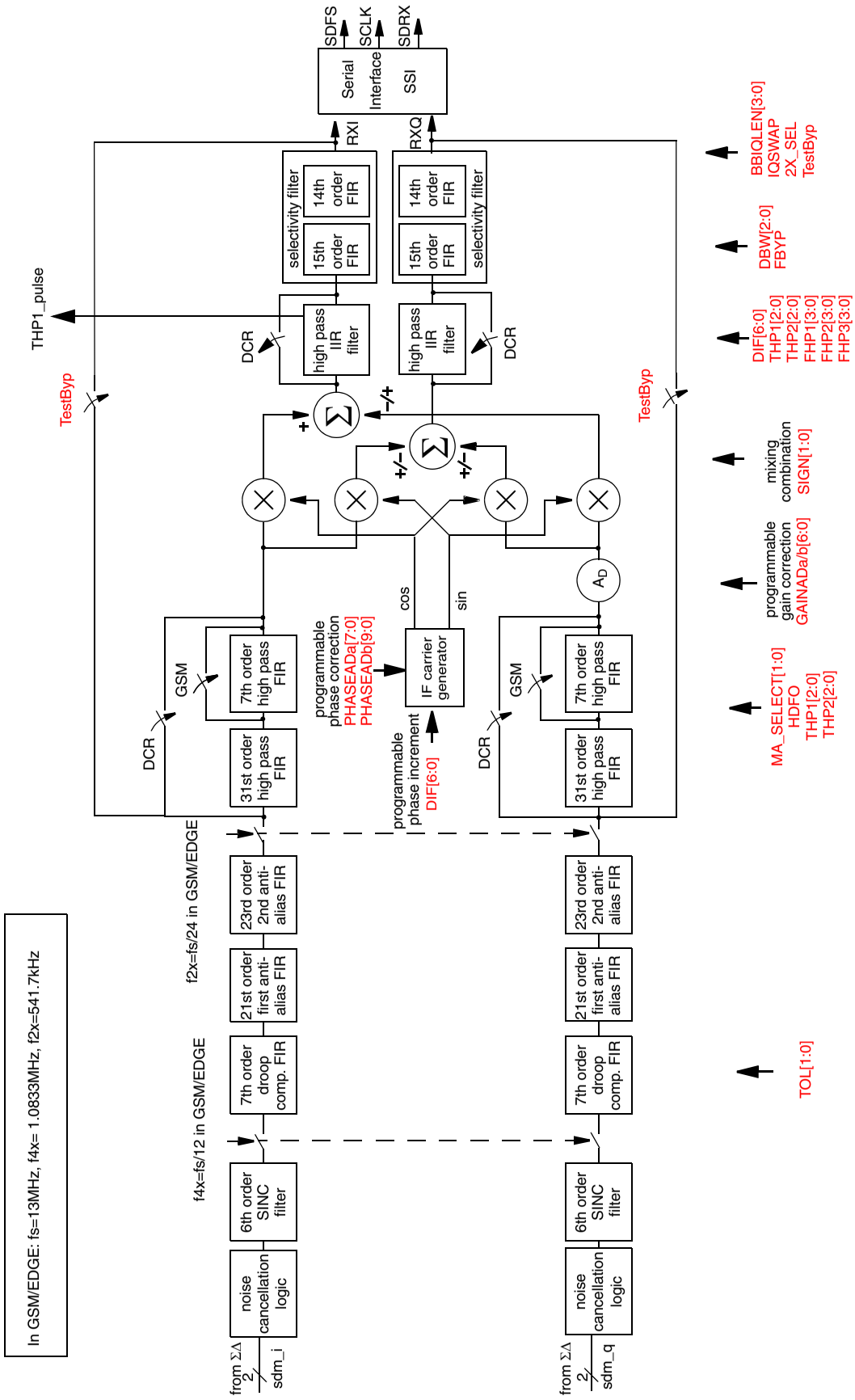


Figure 27-3. Rx coprocessor Block Diagram

27.3.1.1 Noise cancellation block

This block eliminates the second order shaped quantization noise from the sigma-delta modulator. It delivers a 6-bit wide data stream for further processing. The noise cancellation transfer function is given in Equation 27-1 below.

$$h(z) = \text{sdm}[0] \cdot z^{-2} - \text{sdm}[1] \cdot \frac{1}{a_2 b_2 c_2} \cdot (1 - 2z^{-1} + z^{-2}) \quad \textbf{with } a_2 = b_2 = c_2 = 0.5 \qquad \textit{Eqn. 27-1}$$

sdm being either sdm_i or sdm_q.

27.3.1.2 SINC filter and first down sampling

The SINC filter is made of an integrator cascaded with a comb filter. It delivers 17 bit outputs. Data up to the selectivity filter stay 17 bit wide at input and output of each blocks.
The I and Q information is decimated by 12 in GSM/EDGE mode. The sampling clock is then f4x=fs/12 = 1.0833MHz. The goal of this 6th order SINC filter is to provide good attenuation at frequencies above the intermediate f4x sampling rate to avoid aliasing. The integrator is working at the sigma delta output rate, while the comb filter is working at f4x. The overall transfer function is given in Equation 27-2 below.

$$h(z) = \left(\frac{1 - z^{-12}}{1 - z^{-1}} \right)^6 \qquad \textit{Eqn. 27-2}$$

27.3.1.3 Droop compensation FIR filter

This 7th order filter is working at f4x. It helps to compensate the frequency response of the preceding stages made of the analog filters located before the sigma-delta, the sigma-delta modulator and the SINC filter. This response is not flat in the signal bandwidth, especially at the end of the band. A first coarse compensation is realized in this low order FIR filter while a fine equalization is done in the following anti-alias filters. The TOL[1:0] bits allow to select 3 different compensation values. The transfer function is given in Equation 27-3 below and the coefficients in Table 27-2

$$h(z) = \frac{1}{a_0}(b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4} + b_5 z^{-5} + b_6 z^{-6} + b_7 z^{-7}) \qquad \textit{Eqn. 27-3}$$

Coefficient	GSM/EDGE		
	TOL=10	TOL=00	TOL=01
a0	16	16	16
b0=b7	1	1	1

Table 27-2. Anti-droop filter coefficients

Coefficient	GSM/EDGE		
	TOL=10	TOL=00	TOL=01
b1=b6	-3	-2	-3
b2=b5	-1	-3	-2
b3=b4	11	12	12

Table 27-2. Anti-droop filter coefficients

27.3.1.4 Anti-aliasing FIR filter

This filter operates at f4x but the order is different depending on the modes:
This filter is made of a 21st order filter followed by a 23rd order filter making a 45th order filter.The transfer function for the first anti-alias filter is given below and its coefficients in Table 27-3.

$$h(z) = \frac{1}{a_0}(b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + ... + b_{19} z^{-19} + b_{20} z^{-20} + b_{21} z^{-21}) \qquad \textit{Eqn. 27-4}$$

Table 27-3. First anti-alias filter coefficients

Coefficients	values
a0	64
b0=b21	-1
b1=b20	1
b2=b19	2
b3=b18	-1
b4=b17	-3
b5=b16	2
b6=b15	4
b7=b14	-3
b8=b13	-7
b9=b12	8
b10=b11	30

The second anti-alias filter transfer function is given in Equation 27-5 below and the coefficients in Table 27-4 .

$$h(z) = \frac{1}{a_0}(b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + ... + b_{21} z^{-21} + b_{22} z^{-22} + b_{23} z^{-23}) \qquad \textit{Eqn. 27-5}$$

Table 27-4. GSM/EDGE

Coefficients	values
a0	256
b0=b23	-4
b1=b22	0
b2=b21	4
b3=b20	0
b4=b19	-8
b5=b18	-4
b6=b17	12
b7=b16	8
b8=b15	-20
b9=b14	-16
b10=b13	40
b11=b12	112

27.3.1.5 Second down sampling

This block decimates the droop compensation filter output by 2.
The decimation clock is then f2x= fs/24 = 541.667kHz

27.3.1.6 High pass filter

To reduce the DC offset in the receive chain an adaptative scheme is used since the DC offset is dependent on some variable parameters. Refer to the “Fast DC adapt module” for more details.

However some remaining DC offset needs to be removed. In DCR mode the DC offset is located in the middle of the spectrum, while in Very low IF mode the DC offset is located at only one side of the spectrum. This implies two different procedures to filter the remaining DC offset:

- In non DCR mode some high pass notch filters are used to remove the DC offset and a part of the 1/f noise before the complex quadrature mixer. For GSM the filter is a 33st order filter while for EDGE it is a 39th order filter.
- In DCR mode, these filters are bypassed and replaced by high pass IIR filters after the complex quadrature mixer which is used only for quadrature and gain corrections in this case.

27.3.1.6.1 Non DCR mode

The advantage of the GSM filter version is a shorter time response while the EDGE version reduces the frequency response overshoot.

The FIR transfer function for the high pass filter in non DCR mode is given in Equation 27-6 for the GSM mode:

$$h(z) = z^{-16} - \frac{1}{64}(1 + z^{-1} + z^{-2} + z^{-3} + ... + z^{-30} + z^{-31})(1 + z^{-1}) \quad \textit{Eqn. 27-6}$$

The FIR transfer function for the high pass filter in non DCR mode is given in Equation 27-7 for the EDGE mode:

$$h(z) = z^{-19} - \frac{1}{256}(1 + z^{-1} + z^{-2} + z^{-3} + ... + z^{-30} + z^{-31})(1 + z^{-1} + ... + z^{-7}) \quad \textit{Eqn. 27-7}$$

The high pass filter is implemented by subtracting the output of a low pass filter from an all pass filter. The low pass filter allows to estimate the DC offset value. The filter has the capability of holding this DC offset value and using it directly instead of the low pass filter information. This is decided by software through the SPI bit HDFO. HDFO provides the option to stop the high pass filter during the active burst.

If HDFO is set, the high pass filter will be stopped after a programmable delay given by the THP1 and THP2 SPI bits. If an Active Hold Delay word is defined as: AHD[6] = THP2[2], AHD[5] = THP2[1], AHD[4]=THP2[0], AHD[3]=THP1[2], AHD[2]=THP1[1], AHD[1]=THP1[0], AHD[0]=0, the fine dc adapt time is given by:

$$\text{fine dc adapt time} = \frac{AHD[6:0]}{f2x} \quad \textit{Eqn. 27-8}$$

When using the HDFO option, the fine dc adapt time programmed through the THP1 and THP2 SPI bits should correspond at least to the processor latency time up to the high pass filter, plus the delay required for the DC cancellation.

The processor latency time from the sigma-delta up to the high pass filter is 33 f2x clock cycles in GSM and 36 cycles in EDGE. This gives respectively 61us and 65us.

The cancellation of a DC step in the high pass filter requires that it goes all the way through the high pass filter. The processor should be started sooner than the RX burst arrival time by at least half the time needed to go through the high pass FIR. This ensures that when some meaningful information comes out of the processor the DC will have been cancelled as well. The delay required for the DC cancellation corresponds to 17 cycles of the f2x clock in GSM mode and 20 in EDGE. This gives 32us in GSM and 37us in EDGE.

Thus, when using the HDFO option, the fine dc adapt time programmed through the THP1 and THP2 SPI bits should correspond at least to 50 f2x clock cycles in GSM and 56 cycles in EDGE. This gives respectively 93us and 104us

27.3.1.6.2 DCR mode

The IIR transfer function for the high pass filter in DCR mode is given in Equation 27-9:

$$h(z) = (1 - 2^{-k}) \cdot \frac{1 - z^{-1}}{1 - (1 - 2^{-k})z^{-1}} \quad \textit{Eqn. 27-9}$$

In order to speed up the DC offset cancellation, three different k values corresponding to three different corner frequencies may be programmed using the SPI bits FHP1[2:0], FHP2[3:0] and FHP3[3:0]. These values, k_1, k_2 and k_3 are applied sequentially during phases defined by the bits THP1[2:0] and THP2[2:0].
Table 27-5 shows the k values that can be programmed during the THP1 phase.
Table 27-7 shows the k values that can be programmed during the THP2 and remaining phase.

Equation 27-9 can be re-written as Equation 27-10 with h(z) = out(z)/in(z)

$$\text{out}(z) = (1 - 2^{-k}) \cdot (\text{in}(z) + z^{-1}(\text{out}(z) - \text{in}(z))) \qquad \textit{Eqn. 27-10}$$

The delayed out(z)-in(z) represents the DC correction applied to the input signal.

Note from the tables that some k values will cause a hold DC correction or a clear DC conversion.
The hold DC correction means that the quantity out(z)-in(z) will not be re-evaluated but memorized. In this case the output is simply the input minus the stored DC correction. The $1\text{-}2^k$ factor is not applied.
The clear DC correction means that the quantity out(z)-in(z) will not be re-evaluated and forced to zero. In this case the output is simply the input. The $1\text{-}2^k$ factor is not applied.

Table 27-5. high bandwidth coefficients

FHP1[2:0]	k_1	3dB cut-off freq f2x=541.67kHz
000	3	11.5kHz
001	4	5.57kHz
010	5	2.74kHz
011	6	1.36kHz
100	7	678Hz
101	8	338Hz
110	-	clear DC correction
111	-	hold DC correction

Table 27-6. THP1 duration

THP1[2:0]	number of f2x period during which k_1 is selected
000	10
001	20
010	30
011	40
100	50
101	60
110	70
111	80

Table 27-7. medium and low bandwidth coefficients

FHP2/FHP3[3:0]	k_2/k_3	3dB cut-off freq f2x=541.67kHz
0000	4	5.57kHz
0001	5	2.74kHz
0010	6	1.36kHz
0011	7	678Hz
0100	8	338Hz
0101	9	169Hz
0110	10	84.7Hz
0111	11	43.8Hz
1000	12	23.8Hz
1001	13	10.9Hz
1010	14	5.94Hz
1011	15	2.64Hz
1100	16	1.32Hz
1101	17	0.68Hz
1110	-	clear DC correction
1111	-	hold DC correction

Table 27-8. THP2 duration

THP2[2:0]	number of f2x period during which k_2 is selected
000	10
001	20
010	30
011	40
100	50
101	60
110	70
111	80

27.3.1.7 Complex Quadrature Mixer with Gain and Phase correction

The quadrature mixer operates at f2x, the lowest possible rate and then minimizes power consumption. It mixes the very low IF I and Q components with the digital local oscillator (DLO) sines and cosines outputs. Further more, the mixer can compensate the matching errors in gain and phase between the I and Q channels due mainly to the RF mixer and analog filters located in a front end chip. The phase correction is introduced through the DLO while the gain correction is introduced in the Q path.

The SIGN[1:0] bits from the SPI allows to select four possible functions for the mixer.

27.3.1.8 Programmable selectivity filter

After baseband down conversion and image reduction, the quadrature I and Q signals are further processed by a last filter that performs channel selectivity and out of band noise rejection. It is composed of a15th order FIR filter followed by a 14 order FIR filter. The selectivity filter is operating at f2x and delivers a 16bit wide data. Its bandwidth can be controlled by programming the DBW[2:0] bits offering a set of 7 possibilities in GSM. The 14 order filter can be bypassed if the FBYP SPI bit is set. The selectivity filter has been optimized to get good performance in EDGE mode.

The transfer function of the 15th order FIR composing the selectivity filter is given in Equation 27-11 below and the coefficients in Table 27-9

$$h(z) = \frac{1}{a_0}(b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + ... + b_{13} z^{-13} + b_{14} z^{-14} + b_{15} z^{-15}) \qquad \textit{Eqn. 27-11}$$

Table 27-9. Selectivity filter coefficients of the 15th order FIR

DBW[2:0]	3dB BW	a0	b0=b15	b1=b14	b2=b13	b3=b12	b4=b11	b5=b10	b6=b9	b7=b8
000	67kHz	1024	0	-16	-24	-32	0	80	200	288
001	71kHz	1024	0	-16	-28	-40	-16	76	200	304
010	75kHz	1024	0	0	-16	-36	-24	64	200	312
011	79kHz	1024	0	0	-16	-36	-32	48	200	320
100	83kHz	1024	0	8	-8	-32	-44	40	200	340
101	90kHz	1024	-8	0	12	-13	-50	6	189	368
111	100kHz	1024	0	16	20	-16	-64	0	200	400
111	not used									

The transfer function of the 14th order FIR composing the selectivity filter is given in Equation 27-12 below and the coefficients in . Table 27-10 . It provides extra out-of-band rejection

$$h(z) = \frac{1}{a_0}(b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + ... + b_{13} z^{-13} + b_{14} z^{-14}) \qquad \textit{Eqn. 27-12}$$

Table 27-10. Selectivity filter coefficient of the 14th order FIR

Coefficients	values
a0	64
b0=b14	-1
b1=b13	2
b2=b12	2
b3=b11	-3
b4=b10	-5
b5=b9	4
b6=b8	20
b7	28

27.3.1.9 Digital Local Oscillator

The digital local oscillator (DLO) produces sines and cosines outputs at IF frequency based on the f2x clock frequency and the DIF[6:0] bits. It is based on two 64-entry look-up tables and an interpolation algorithm which keeps the harmonic levels low.

The IF frequency is given by: $Fif = DIF \times 2.116kHz$

Setting DIF to zero will not imply DCR mode, to use DCR mode the MA_SELECT bits should be written to 10.

Typical programmed DIF values are:
58 in EDGE for an IF of 122.721kHz
54 in GSM for an IF of 114.258kHz

A phase correction factor PHASEADxa may be applied to the DLO. This 8 bit signed value is programmed through the SPI and allows to compensate phase variation due to the analog stages before the RxCPROC. The phase adjustment is given by:

$$\text{Phase adjust} = \frac{360°}{2^{11}} \times \text{PHASEADxa}$$

This gives a possible phase correction of ± 22 degrees.

A phase correction factor PHASEADxb may be applied to the DLO. This 10 bit signed value is programmed through the SPI and allows to compensate phase variation due to the analog stages before the RxCPROC. The phase adjustment is given by:

$$\text{Phase adjust} = \frac{360°}{2^{11}} \times \text{PHASEADxb}$$

This gives a possible phase correction of ± 90 degrees.

Note that the PHASEADxb_MSB SPI bits described in Table 27-11 are actually the PHASEADxb[9:8]. These 2 bits extension were added to increased the compensation efficiency of the mixer in edge mode.

27.4 SPI interface

The A2DIGL SPI logic section will decode the various bits received serially from the MQSPI and generate the RxCPROC control signals. The A2DIGL provides the right a2di_phaseada[7:0], a2di_phaseadb[9:0], a2di_gainada[6:0] and a2di_gainadb[6:0] based on PHASEADSEL[1:0] and GCSTEP group5 SPI bits.It also controls the clock used by the RxCPROC.

Table 27-11 indicates the relevant SPI bits for the RxCPROC.

Table 27-11.

<table>
<tr><th>Group</th><th>Name</th><th>Description</th></tr>
<tr><td>2</td><td>TOL[1:0]
Bits 57-56
Reset Value: 0</td><td>Selects three different compensations of the droop digital filter. This is used to compensate to low, nominal or high bandwidth of the analog filters which precede the sigma delta converter.
<table><tr><th>TOL[1:0]</th><th>Analog filter compensated</th></tr><tr><td>00</td><td>Nominal</td></tr><tr><td>01</td><td>Minimum BW</td></tr><tr><td>10</td><td>Maximum BW</td></tr></table></td></tr>
<tr><td>2</td><td>THP1[2:0]
Bits 60-58
Reset Value: 0x3</td><td>Programs the time from the end of the fast DC adapt to the end of the time at which the high value corner FHP1 is selected in DCR mode.
<table><tr><th>THP1[2:0]</th><th>FHP1 duration in DCR mode</th></tr><tr><td>000</td><td>10/f2x</td></tr><tr><td>001</td><td>20/f2x</td></tr><tr><td>010</td><td>30/f2x</td></tr><tr><td>011</td><td>40/f2x</td></tr><tr><td>100</td><td>50/f2x</td></tr><tr><td>101</td><td>60/f2x</td></tr><tr><td>110</td><td>70/f2x</td></tr><tr><td>111</td><td>80/f2x</td></tr></table>In EDGE or GSM mode THP2 is concatenated with THP1 and multiplied by 2 to determine the duration time of the fine DC adapt in number of f2x periods</td></tr>
</table>

Table 27-11.

<table>
<tr><th>Group</th><th>Name</th><th>Description</th></tr>
<tr><td>2</td><td>THP2[2:0]
Bits 63-61
Reset Value: 0x3</td><td>Programs the time during which the medium value corner FHP2 is selected
<table>
<tr><th>THP2[2:0]</th><th>FHP2 duration in DCR mode</th></tr>
<tr><td>000</td><td>10/f2x</td></tr>
<tr><td>001</td><td>20/f2x</td></tr>
<tr><td>010</td><td>30/f2x</td></tr>
<tr><td>011</td><td>40/f2x</td></tr>
<tr><td>100</td><td>50/f2x</td></tr>
<tr><td>101</td><td>60/f2x</td></tr>
<tr><td>110</td><td>70/f2x</td></tr>
<tr><td>111</td><td>80/f2x</td></tr>
</table>
In EDGE or GSM mode THP2 is concatenated with THP1 and multiplied by 2 to determine the duration time of the fine DC adapt in number of f2x period.</td></tr>
<tr><td>2</td><td>FHP1[2:0]
Bits 66-64
Reset Value: 0</td><td>Program the high pass value of the first step on the fine digital DC adapt in DCR mode
<table>
<tr><th>FHP1[2:0]</th><th>High Pass Corner (GSM)</th></tr>
<tr><td>000</td><td>11.5kHz</td></tr>
<tr><td>001</td><td>5.57kHz</td></tr>
<tr><td>010</td><td>2.74kHz</td></tr>
<tr><td>011</td><td>1.36kHz</td></tr>
<tr><td>100</td><td>678Hz</td></tr>
<tr><td>101</td><td>338Hz</td></tr>
<tr><td>110</td><td>Clear the DC correction</td></tr>
<tr><td>111</td><td>Hold the DC correction</td></tr>
</table></td></tr>
<tr><td>2</td><td>FHP2[3:0]
Bits 71-68
Reset Value: 1</td><td>Program the high pass value of the second step on the fine digital DC adapt in DCR mode. The high pass corner are programmed in the same way than FHP3 (see below)</td></tr>
</table>

Table 27-11.

<table>
<tr><th>Group</th><th>Name</th><th>Description</th></tr>
<tr><td>2</td><td>FHP3[3:0]
Bits 75-72
Reset Value: 0xF</td><td>Program the high pass value of the last step on the fine digital DC adapt in DCR mode
<table><tr><th>FHP2[3:0]
or
FHP3[3:0]</th><th>High Pass Corner (GSM)</th></tr><tr><td>0000</td><td>5.57kHz</td></tr><tr><td>0001</td><td>2.74kHz</td></tr><tr><td>0010</td><td>1.36kHz</td></tr><tr><td>0011</td><td>678Hz</td></tr><tr><td>0100</td><td>338Hz</td></tr><tr><td>0101</td><td>169Hz</td></tr><tr><td>0110</td><td>84.7Hz</td></tr><tr><td>0111</td><td>43.8Hz</td></tr><tr><td>1000</td><td>23.8Hz</td></tr><tr><td>1001</td><td>10.9Hz</td></tr><tr><td>1010</td><td>5.94Hz</td></tr><tr><td>1011</td><td>2.64Hz</td></tr><tr><td>1100</td><td>1.32Hz</td></tr><tr><td>1101</td><td>0.68Hz</td></tr><tr><td>1110</td><td>Clear the DC correction</td></tr><tr><td>1111</td><td>Hold the DC correction</td></tr></table></td></tr>
<tr><td>3</td><td>PHASEADxb_MSB[1:0]
Bits 37-36
Reset Value: 0</td><td>These are the MSB bits for the PHASEADxb register. To have backward software code compatibility these additional bits were place here.
(see also PHASEADxb description)</td></tr>
<tr><td>3</td><td>HDFO
Bit 38
Reset Value:1</td><td>If programmed high, then the DC offset correction is held after the programmed time (through THP1 and THP2) in the digital high pass filter prior to the complex multiplier. If programmed low then the filter continuously corrects the DC offset value.</td></tr>
<tr><td>3</td><td>GainLUPADR
bit 39
Reset Value: 0</td><td>There are two pairs of internal storage registers used for the phased values of gain correction versus AGC. This SPI group selects which one of the two registers are written in the corresponding SPI initialization write.
<table><tr><th>GainLUPADR</th><th>Gain correction value written</th></tr><tr><td>0</td><td>GAINAD0a/b</td></tr><tr><td>1</td><td>GAINAD1a/b</td></tr></table></td></tr>
</table>

Table 27-11.

<table>
<tr><th>Group</th><th>Name</th><th>Description</th></tr>
<tr><td>3</td><td>GAINADxa[6:0]
Bits 46-40
Reset Value: 0</td><td>Signed gain error adjust for image rejection at IF passband frequency A. There are two registers corresponding to this SPI group. The register used in radio operation is selected by GCSTEP. The register written at initialization or mode change is selected by GainLUPADR</td></tr>
<tr><td>3</td><td>TestByp
Bit 47
Reset Value: 0</td><td>If programmed high then the digital mixer and filter of the IF section are bypassed to allow external digital filtering by the DSP.</td></tr>
<tr><td>3</td><td>GAINADxb[6:0]
Bits 54-48
Reset Value: 0</td><td>Signed gain error adjust for image rejection at IF passband frequency B. There are two registers corresponding to this SPI group. The register used in radio operation is selected by GCSTEP. The register written at initialization or mode change is selected by GainLUPADR</td></tr>
<tr><td>3</td><td>IQSWAP
Bit 55
Reset Value: 0</td><td>If this bit is programmed low then the I bits are sent first on SDFS. If programmed high then the Q bits are sent first.</td></tr>
<tr><td>3</td><td>PHASEADxa[7:0]
Bits 63-56
Reset Value: 0</td><td>Signed phase error adjust for image rejection at IF passband frequency A. There are four registers corresponding to this SPI group. The register used in radio operation is selected by PHASEADSEL[1:0] and is linked to the RF frequency selected by the radio software. The register written at initialization is selected by PhLUPADR[1:0]</td></tr>
<tr><td>3</td><td>PHASEADxb[7:0]
Bits 71-64
Reset Value: 0</td><td>Signed phase error adjust for image rejection at IF passband frequency B. There are four registers corresponding to this SPI group. The register used in radio operation is selected by PHASEADSEL[1:0] and is linked to the RF frequency selected by the radio software. The register written at initialization is selected by PhLUPADR[1:0]
PHASEADxb overall size is10 bits {PHASEADxb_MSB[1:0], PHASEADxb[7:0]}
(See also PHASEADxb_MSB description)</td></tr>
<tr><td>3</td><td>SIGN[1:0]
Bits 73-72
Reset Value: 0</td><td>Program any possible mixing combination for I and Q outputs in the balanced complex multiplier
<table>
<tr><th>SIGN[1:0]</th><th>Combination</th></tr>
<tr><td>00</td><td>(I+jQ)*exp(+j*2π*DFI*t)</td></tr>
<tr><td>01</td><td>(I+jQ)*exp(-j*2π*DFI*t)</td></tr>
<tr><td>10</td><td>(I-jQ)*exp(+j*2π*DFI*t)</td></tr>
<tr><td>11</td><td>(I-jQ)*exp(-j*2π*DFI*t)</td></tr>
</table>
</td></tr>
</table>

Table 27-11.

<table>
<tr><th>Group</th><th>Name</th><th>Description</th></tr>
<tr><td>3</td><td>PhLUPADR[1:0]
Bits75-74
Reset Value: 0</td><td>There are four pairs of internal storage registers used for the phased values of phase correction versus RF frequency. This SPI group selects which one of the four registers are written in the corresponding SPI initialization write.
<table>
<tr><th>PhLUPADR</th><th>Phase correction value written</th></tr>
<tr><td>00</td><td>PHASEAD0a/b</td></tr>
<tr><td>01</td><td>PHASEAD1a/b</td></tr>
<tr><td>10</td><td>PHASEAD2a/b</td></tr>
<tr><td>11</td><td>PHASEAD3a/b</td></tr>
</table></td></tr>
<tr><td>4</td><td>BSAVE
Bit 61
Reset Value: 1</td><td>Program low for normal operation. If BSAVE is programmed high then when CE transition low, the transceiver will go in power save mode. In this mode only the reference oscillator and tracking regulators are active. BSAVE will revert to a logic low on the next CE transition from high to low, regardless of which SPI segment is programmed.</td></tr>
<tr><td>4</td><td>MA_SELECT[1:0]
Bits 63-62
Reset Value: 0</td><td>Select the radio standard used. This selects the order of the digital filters for each standards.
<table>
<tr><th>MA_SELECT</th><th>Standards</th></tr>
<tr><td>00</td><td>GSM</td></tr>
<tr><td>01</td><td>EDGE</td></tr>
<tr><td>10</td><td>DCR wide</td></tr>
</table></td></tr>
<tr><td>4</td><td>DIF[6:0]
Bits 70-64
Reset Value: 0x36</td><td>Selects the digital IF local oscillator. The bandwidth is DIF*2.116kHz in GSM mode.</td></tr>
</table>

Table 27-11.

<table>
<tr><th>Group</th><th>Name</th><th>Description</th></tr>
<tr><td>4</td><td>DBW[2:0]
Bits74-72
Reset Value: 0x5</td><td>Digital filter bandwidth select.
<table><tr><th>DBW[2:0]</th><th>3dB bandwidth</th></tr><tr><td>000</td><td>67kHz</td></tr><tr><td>001</td><td>71kHz</td></tr><tr><td>010</td><td>75kHz</td></tr><tr><td>011</td><td>79kHz</td></tr><tr><td>100</td><td>83kHz</td></tr><tr><td>101</td><td>90kHz</td></tr><tr><td>110</td><td>100kHz</td></tr><tr><td>111</td><td>Not used</td></tr></table></td></tr>
<tr><td>4</td><td>FBYP
Bit 75
Reset Value: 0</td><td>If programmed high the 14th order filter composing the selectivity filter is bypassed</td></tr>
<tr><td>5</td><td>BSAVERX
Bit 2
Reset Value: 1</td><td>If programmed high then the receiver is placed in standby mode.</td></tr>
<tr><td>5</td><td>AB_SEL
Bit 7
Reset Value: 1</td><td>If programmed high then on the falling edge of RXACQ, BSAVE and BSAVERX will activate. Software will need to insure that the last valid data has been transferred from the Rx SSI interface before switching RXACQ.</td></tr>
</table>

Table 27-11.

<table>
<tr><th>Group</th><th>Name</th><th>Description</th></tr>
<tr><td>5</td><td>BBIQLEN[3:0]
Bits 11-8
Reset Value: 0xE</td><td>Configures the BB serial interface for 8-bit, 12 bit, or 16bit I/Q data transfers. Note that 8-bit and 12-bit I/Q data is just the 16-bit I/Q data with 4bits or 8 bits discarded. The dynamic range of the converter will result in14 bits of valid data in wide (>60kHz i.e. GSM/EDGE) bandwidths
<table>
<tr><th>BBIQLEN[3:0]</th><th>Word Width</th></tr>
<tr><td>0000</td><td>Not used</td></tr>
<tr><td>0001</td><td>8 bits (remove 8 MSBs)</td></tr>
<tr><td>0010</td><td>8 bits (remove 7 MSBs & 1 LSB)</td></tr>
<tr><td>0011</td><td>8 bits (remove 6 MSBs & 2LSBs)</td></tr>
<tr><td>0100</td><td>8 bits (remove 5 MSBs & 3 LSBs)</td></tr>
<tr><td>0101</td><td>8 bits (remove 4 MSBs & 4 LSBs)</td></tr>
<tr><td>0110</td><td>8 bits (remove 3 MSBs & 5 LSBs)</td></tr>
<tr><td>0111</td><td>8 bits (remove 2 MSBs & 6 LSBs)</td></tr>
<tr><td>1000</td><td>8 bits (remove 1 MSB & 7 LSBs)</td></tr>
<tr><td>1001</td><td>8 bits (remove 8 LSBs)</td></tr>
<tr><td>1010</td><td>12 bits (remove 4 MSBs)</td></tr>
<tr><td>1011</td><td>12 bits (remove 3 MSBs & 1 LSB))</td></tr>
<tr><td>1100</td><td>12 bits (remove 2 MSBs & 2 LSBs</td></tr>
<tr><td>1101</td><td>12 bits (remove 1 MSB & 3 LSBs)</td></tr>
<tr><td>1110</td><td>12 bits (remove 4 LSBs)</td></tr>
<tr><td>1111</td><td>16 bits</td></tr>
</table></td></tr>
<tr><td>5</td><td>PHASEADSEL
[1:0]
bit 13-12
Reset Value: 0</td><td>These bits select which phase correction values to apply for image rejection. It is assumed that these bits are correlated to frequency ranges of the channel selection to account for variances in phase balance over the entire frequency band of the radio. The phase correction values are loaded into a lookup table using the PhLUPADR and PHASEAD bits in Group 3. The phase correction values are addressed by PHASEADSEL as follows:
<table>
<tr><th>PHASEADSEL[1:0]</th><th>Gain correction value</th></tr>
<tr><td>00</td><td>PHASEAD0a/b</td></tr>
<tr><td>01</td><td>PHASEAD1a/b</td></tr>
<tr><td>10</td><td>PHASEAD2a/b</td></tr>
<tr><td>11</td><td>PHASEAD3a/b</td></tr>
</table></td></tr>
</table>

Table 27-11.

<table>
<tr><th>Group</th><th>Name</th><th>Description</th></tr>
<tr><td>5</td><td>2X_SEL
Bit 14
Reset Value: 0</td><td>Selects the output rate of the A/D as 1X per GSM bit time if programmed low or 2X per GSM bit time if programmed high</td></tr>
<tr><td>5</td><td>GCSTEP
bit 15
Reset Value: 0</td><td>Gain correction step. This bit is used to step between two programmed values of the gain correction of the Balanced Complex Modulator. It is assumed that factory calibration of two settings of the PMA_AGC and AMP_AGC have been determined which provide the desired overall AGC step. When PMA_AGC and AMP_AGC are programmed between two settings then this bit is used to shift the gain correction.
<table>
<tr><th>GCSTEP</th><th>Gain correction value</th></tr>
<tr><td>0</td><td>GAINAD0a/b</td></tr>
<tr><td>1</td><td>GAINAD1a/b</td></tr>
</table>
</td></tr>
</table>

27.5 BBP interface

A serial bus consisting of SDFS and SDRX will transmit the RXI and RXQ data in 2's complement format to the BBP module. Note that if SPI bit TestByp has been programmed high then the complex quadrature filter circuitry is bypassed. Instead of the selectivity FIR outputs RXI and RXQ, the anti alias outputs are serially transmitted to the BBP to be handled directly by the DSP.
SDRX and SDFS are outputs from the RxCPROC. SDFS is a framing signal which marks the beginning of an I,Q transfer. SDRX is the serial data. The clock used for the serial transfer is SCLK. This is the same clock as the RxCPROC clock except it is gated as needed for the data transfer. This means that SCLK is turned off when data is not being transmitted. This includes the idle times, which may occur within each framing signal interval. SCLK is expected to be a square wave with a duty cycle better than 45/55.

When RXACQ goes high it will activate the digital interface section. Note that even though the first SDFS pulse is suppressed the first group of SCLK pulses will still exist during the time interval of the suppressed SDFS pulse. The first SCLK burst will always start 4 RxCPROC clocks after the rising edge of RXACQ as shown on Figure 27-4, regardless of the transmission format and regardless of BSAVErxcproc arrival time.

The data transmission over the serial bus will begin one period of SDFS after RXACQ has gone high. Since the first transition of SDFS is suppressed, valid data is sent following every SDFS pulse. (Note that data before the actual burst during the adapt time will not be meaningful.) It is expected that data is read on the falling edges of SCLK. The first valid data bit occurs on the next falling SCLK edge after SDFS goes high to indicate the beginning of valid I,Q data.

If RXACQ goes low then SCLK, SDRX and SDFS will go low after completing the data transfer in progress. RXACQ falling edge may happen during the transmission, but RxCPROC clock should be kept running for 2 more frame lengths.

The startup of the RX IQ interface is shown in the diagram below for the 2X GSM first format.

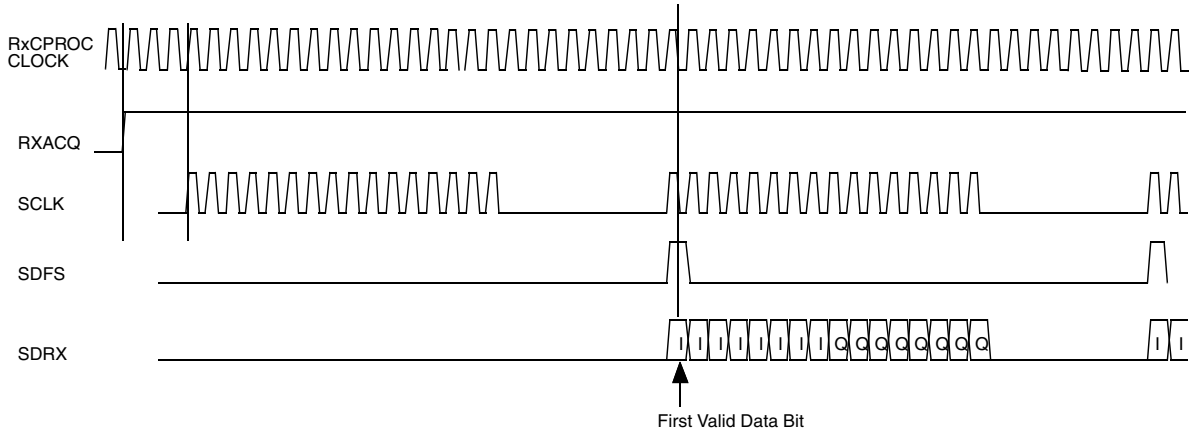


Figure 27-4. Serial Interface startup

In all of the following modes, if IQ_SWAP is programmed to one then Q data is sent first and I data is sent second. If IQ_SWAP is programmed to zero then the transfer is as specified below.

27.5.1 RXI, RXQ Output formats

The RxCPROC can be programmed to output the digital data in three formats for GSM.

For GSM/EDGE signals the A/D converter is equivalent to a 13 bit baseband sigma delta converter. A digital representation of the RXI and RXQ signals can be derived at a 2X or 1X sample rate relative to the GSM bit rate. It is assumed that SCLK is programmed to 13MHz in the GSM/EDGE mode.

27.5.1.1 First GSM format

In the first format, 8 bits of I and Q data are sent for each conversion period. This is selected through the SPI bit BBIQLEN using values from 0000 to 1001. This mode allows full compatibility with the existing MAGIC2 format. The data transmission format is:
for 1X oversample mode: 8 bits for the I channel, then 8 bits for the Q channel followed by 32 zeros.
for 2X oversample mode: 8 bits for the I channel, then 8 bits for the Q channel followed by 8 zeros.

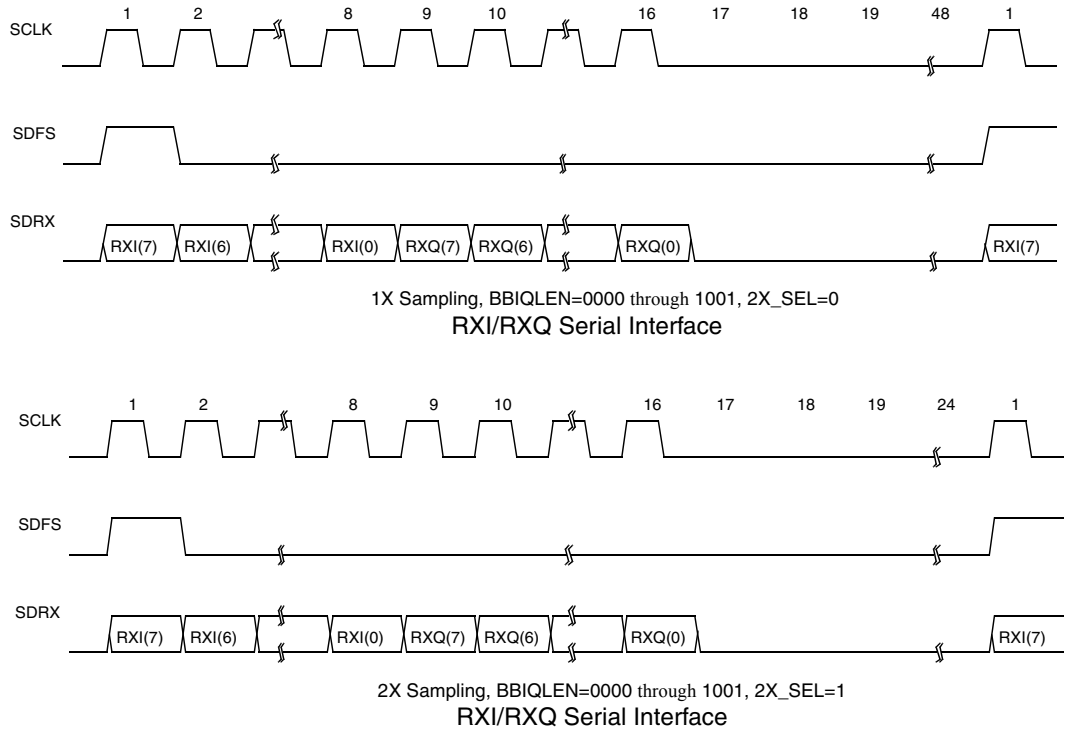


Figure 27-5. First GSM format

In this format BBIQEN can be programmed in 9 different modes to remove various numbers of MSBs or LSBs to fit the 16 bit filter output word to the 8 bit interface word. This is necessary for the following reasons:

- Consider a weak signal with a strong off channel interferer present. The AGC of the receiver is adjusted to reduce the gain of the system so that the input to the sigma delta converter is not excessively large. At the output of the digital filtering the strong interferer is essentially removed and only the signal will remain. In this case the useful signal will reside in the LSBs of the 16 bit word from the digital filters and it would be best to remove MSBs to achieve dynamic range on the signal.
- Now consider the case of a strong signal. The filter output will contain the valid signal within the MSBs of the 16 bit word and thus the MSBs could not be removed and instead the LSBs must be removed to receive the word. The selection of removing MSBs or LSBs can be altered on each timeslot. If the 8 or 12 bit interface modes are used then the software will need to make a decision based on the expected signal strength as to which bits are removed.

27.5.1.2 Second GSM format

In the second format, 12 bits of I and Q data are sent for each conversion period. This is selected through the SPI bit BBIQLEN using values from 1010 to 1110. This mode is to be used for applications requiring greater dynamic range of the receiver. The BBP module can accept the 12 bit format. The data transmission format is:
for 1X oversample mode: 12 bits for the I channel, then 12 bits for the Q channel followed by 24 zeros.
for 2X oversample mode: 12 bits for the I channel, then 12 bits for the Q channel. Note that there are no zero bits in this mode.

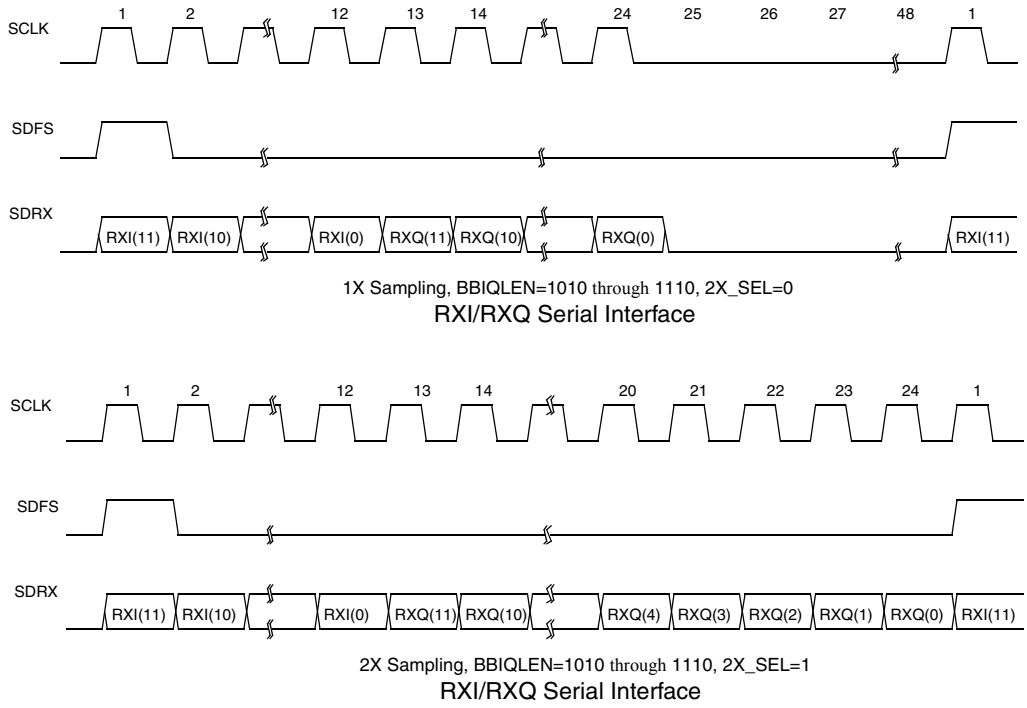


Figure 27-6. Second GSM format

As in the 8 bit mode, the decision to remove MSBs or LSBs is selectable with 5 settings of BBIQLEN.

27.5.1.3 Third GSM format

In the third format, 16 bits of I and Q data are sent for each conversion period. This is selected through the SPI bit BBIQLEN using value 1111. This mode is the most straightforward to use as there is no selection of removal of MSBs or LSBs. Due to the length of the I and Q data this mode will only be available in 1X mode. The format for 1X oversample mode data transmission is 16 bits for the I channel, then 16 bits for the Q channel followed by 16 zeros.

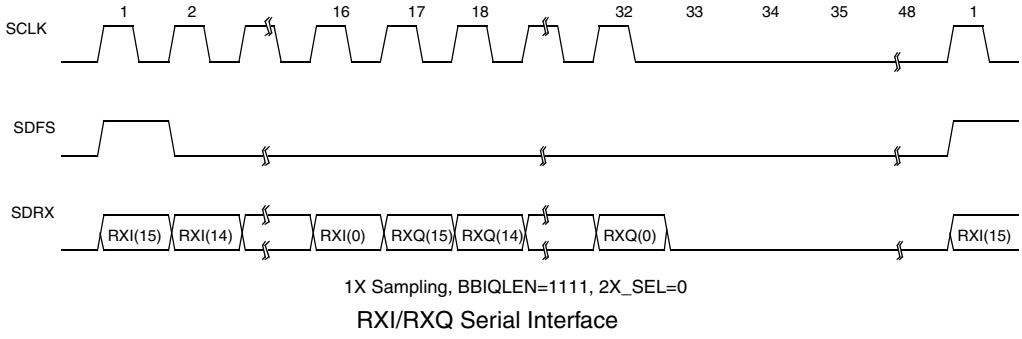


Figure 27-7. Third GSM format

27.6 Power On Off operation

Three SPI bits control the various battery save modes: BSAVE, BSAVERX and BSAVETX. Only BSAVE and BSAVERX are relevant for the RxCPROC.

When the BSAVE SPI bit is programmed high, the transceiver enters battery save mode on the falling edge of the MQSPI chip-select signal CE. Battery save mode shuts down the complete receiver as well as the transmitter. The receiver and transmitter will come out of battery save mode on the next falling edge of CE regardless of which SPI group is programmed or the programmed state of BSAVE.

When the BSAVERX bit is programmed high the complete receiver is shut down.

Automatic setting of the BSAVE, BSAVERX bits is possible through the use of another SPI bit AB_SEL in conjunction with the RXACQ signal from the L1 timer. If programmed true, AB_SEL causes BSAVERX and BSAVE to become true on the falling edge of RXACQ. This control system allows the wake up the receiver by software by sending a MQSPI message in advance of the Rx burst and then allows the receiver to automatically re-enter battery save mode at the end of the Rx sequence.

BSAVE controls the battery save status of the all the transceiver and if set true overrides BSAVERX.

The A2DIGL module uses the BSAVE, BSAVERX and BSAVETX SPI bits to generate battery save control signals. One of these signals, BSAVErxcproc goes directly to the RxCPROC. Two other signals BSAVErxclk and BSAVErxafe are related to the RxCPROC.

BSAVErxclk enables the 13Mhz clock which is used in the receive coprocessor. It lasts for the whole receive sequence.

BSAVErxafe controls the clock for the sigma delta A/D converters. Due to the large pipeline stages in the RxCPROC, the converters can be switched off before the end of operations in the RxCPROC to save power.

BSAVErxcproc starts clocking data through the receive coprocessor when it goes low. BSAVErxcproc must last for the rest of the receive sequence until all valid data have been sent to BBP.

27.7 Timings examples

Figure 27-8 and Figure 27-9 detail typical sequences for the receiver for the VLIF and DCR modes.

The sequence begins with a MQSPI message, programming BSAVERX false and ADAPT_EN true on the falling edge of the MQSPI chip-select signal CE. This initiates a coarse fast DC adapt sequence. The action of CE going from high to low will set BSAVE false and put the RxCPROC out of standby.

At the end of the FastDigitalAdapt, BSAVErxcproc is negated and the RxCPROC will start clocking data. Some time must be allowed for the fine DC adapt tuning before the actual samples from the RX burst are treated:

- In DCR mode, the fine DC cancellation is done in the high pass IIR and the time required for this is given by the THP1, THP2 values. These values are adjusted by the DSP depending on the received signal strength.
- In VLIF mode, the fine DC cancellation is done in the high pass FIR. For GSM this time (noted hpfdel on the figures) corresponds to 17 cycles of the f2x clock and 20 for EDGE. This gives 32us in GSM and 37us in EDGE.

When BSAVErxcproc goes low, data is not yet valid at the RxCPROC output due to the processor latency. When the RXACQ signal from the L1 timer goes from low to high, it indicates that data is valid at the output of the RxCPROC and that transfer to BBP must begin. This avoids the DSP loading non-necessary data. When FBYP is cleared, this latency with respect to the start of the RX burst (noted rxlat on the figures) corresponds to 49 periods of the f2x clock in GSM and 52 in EDGE which gives 91us and 96us respectively. When FBYP is set, rxlat corresponds to 41 periods of the f2x clock in GSM and 44 in EDGE which gives 76us and 83us respectively.

To avoid the last burst sampling being corrupted by a DC offset change at the input of the RxCPROC, the analog front end should not be switched off immediately. The recommended time to wait is 17 f2x clock cycle in GSM and 20 in EDGE.

After the end of the RX burst and after the latency time of the RxCPROC the last sample reaches the SSI interface to be transferred and after the required transmission time the RxCPROC can be switched off.

In these examples it is assumed that AB_SEL is true. Thus when RXACQ goes from high to low BSAVE and BSAVERX will be asserted and the RxCPROC will enter battery save mode. Data transfer to BBP is stopped.

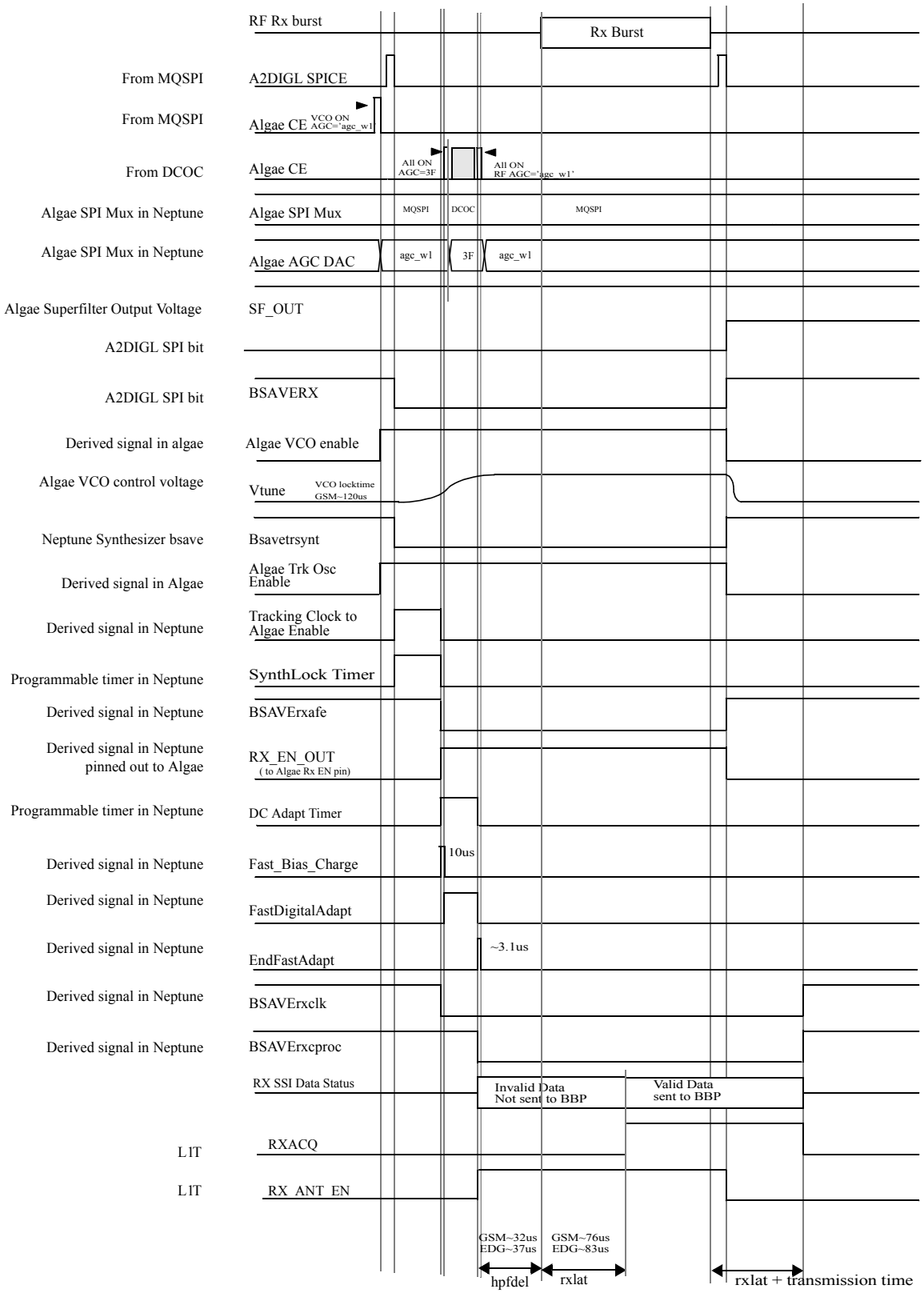


Figure 27-8. Receive timing diagram in VLIF mode

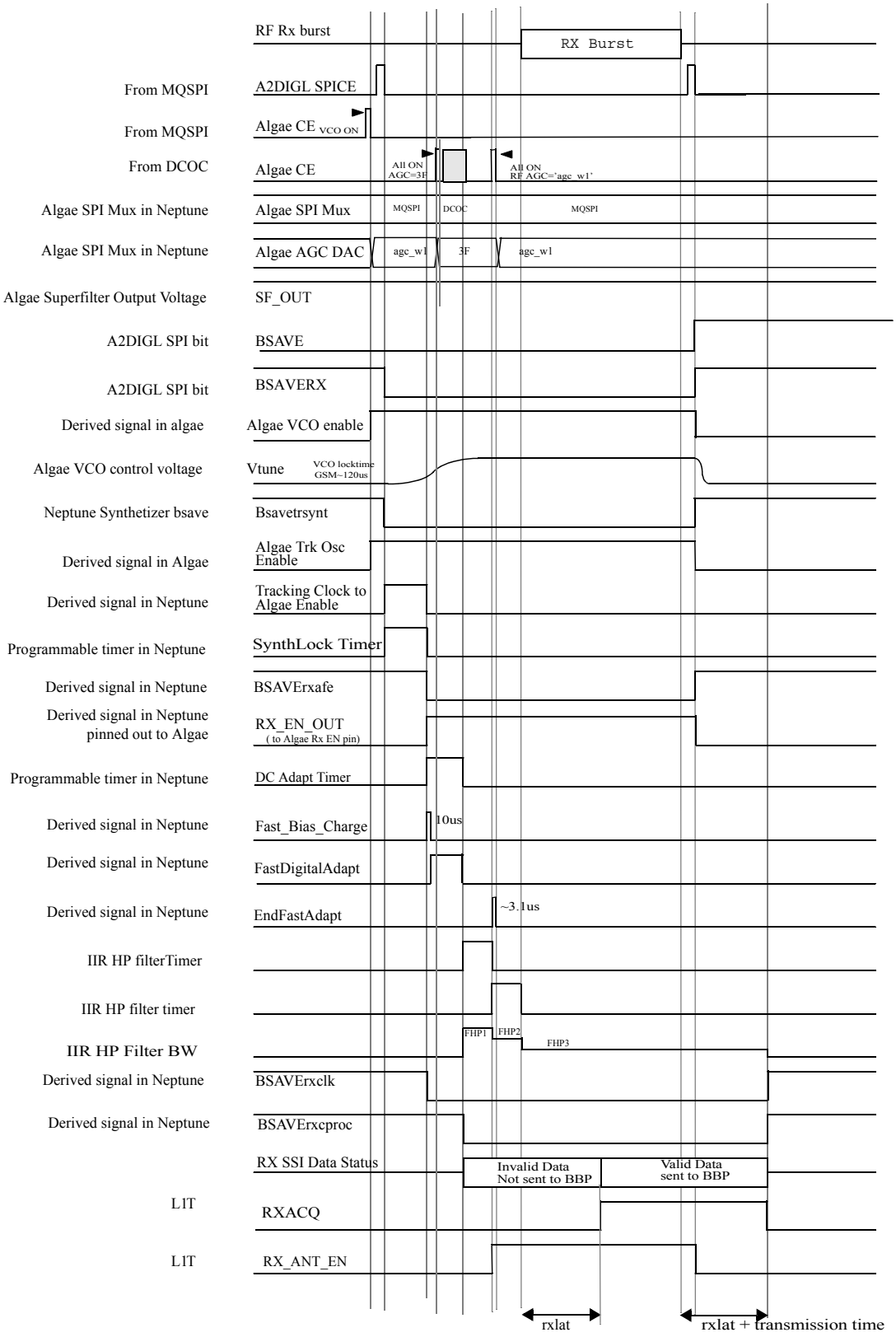


Figure 27-9. Receive timing diagram in DCR mode

27.8 RxCPROC test

The RxCPROC will be functionally tested with the sigma delta in pass through mode. The test sequence is applied on the RX_I,RX_IX,RX_Q and RX_QX pads and goes directly on the RxCPROC inputs. The output of the RxCPROC, which are the serial link signals SDFS, SCLK and SDRX are observed on some GPIO pads. A special test mode also allows to control the SPI bits from the external world.

Chapter 28
Digital Fast DC Adapt (DCADAPT)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	05/07/03		Updated for LTE specification release.

28.1 Introduction

This document describes a DC Offset Correction Control Loop, which is shared between two chips of a low-cost cellular platform for GSM/EDGE market. The ICs in this platform include Algae (RF), Seaweed (power management) and Neptune (ADC, synthesizers, baseband). The DC offset correction loop is a mixed digital/analog type and is split between Algae (analog) and Neptune (digital). For completeness, the whole loop is described in this document, but the specification applies only to the digital part (in Neptune).

For a more in-depth understanding of the Neptune IC, the Algae IC and the offset correction control loop in particular, please refer to the following documents:

- Magic-LV Contract Book
- Neptune Requirements Document
- Algae IC Specification
- Mixed-signal Control Interface (A2DIGL)
- Rx Coprocessor (RxCPROC) Specification

28.1.1 Overview

Figure 28-1 shows a simplified representation of the RX Lineup across Algae and Neptune ICs.

In a Direct Conversion Receiver or a Very Low IF Receiver, the DC offset generated by the self-mixing of the Local Oscillator is generally larger than the signal of interest. This DC offset behaves as an unwanted interferer and so has to be removed.

The DC offset is generally non-static since it depends on the selected frequency and on environmental reflections. Therefore the DC offset cancellation process has to be able to track these variations. In addition, there are other DC offsets generated within the circuit, such as mixer and amplifier offsets, and since the gain of the amplifiers is periodically modified (AGC), these offsets will vary. All of these DC offsets are cancelled using an adaptive scheme which is performed before every reception burst.

In order to minimize the battery current in standby mode, the required adapt time (time to cancel the DC offsets) has to be kept to an absolute minimum.

The overall adapt is performed in two steps:

- A fast (coarse) digital adapt phase. This step reduces the DC offset to a level that is within the dynamic range of the sigma-delta modulator. The target differential DC offset at the sigma-delta inputs is +/-28mV.
- A fine adapt phase. The post-mixing filters will reduce the remaining DC offset to within 30uV.

This specification describes only the Fast Digital DC Adapt components (DCADAPT) i.e. those involved in the fast (coarse) digital adapt phase, which are the shaded blocks in the diagram.

The digital closed loop uses the first bit quantiser of the Sigma-Delta modulator as input to a digital integrator (accumulator), which then goes to a programmable digital first-order low-pass filter. The loop is closed by using the filter output as a correction offset to the IFA amplifier input through a 6-bit D/A converter, which is implemented on the Algae. The filter output is sent to the Algae across a three-wire SPI link.

28.1.2 Features

- Corrects for offset at the inputs to the Sigma-Delta modulator to within +/-28mV
- Settling time of loop less than 38us
- Dedicated SPI connection (13MHz bit-rate) from DCADAPT on Neptune (via Module Control Interface, “A2DIGL”) to DACs on Algae

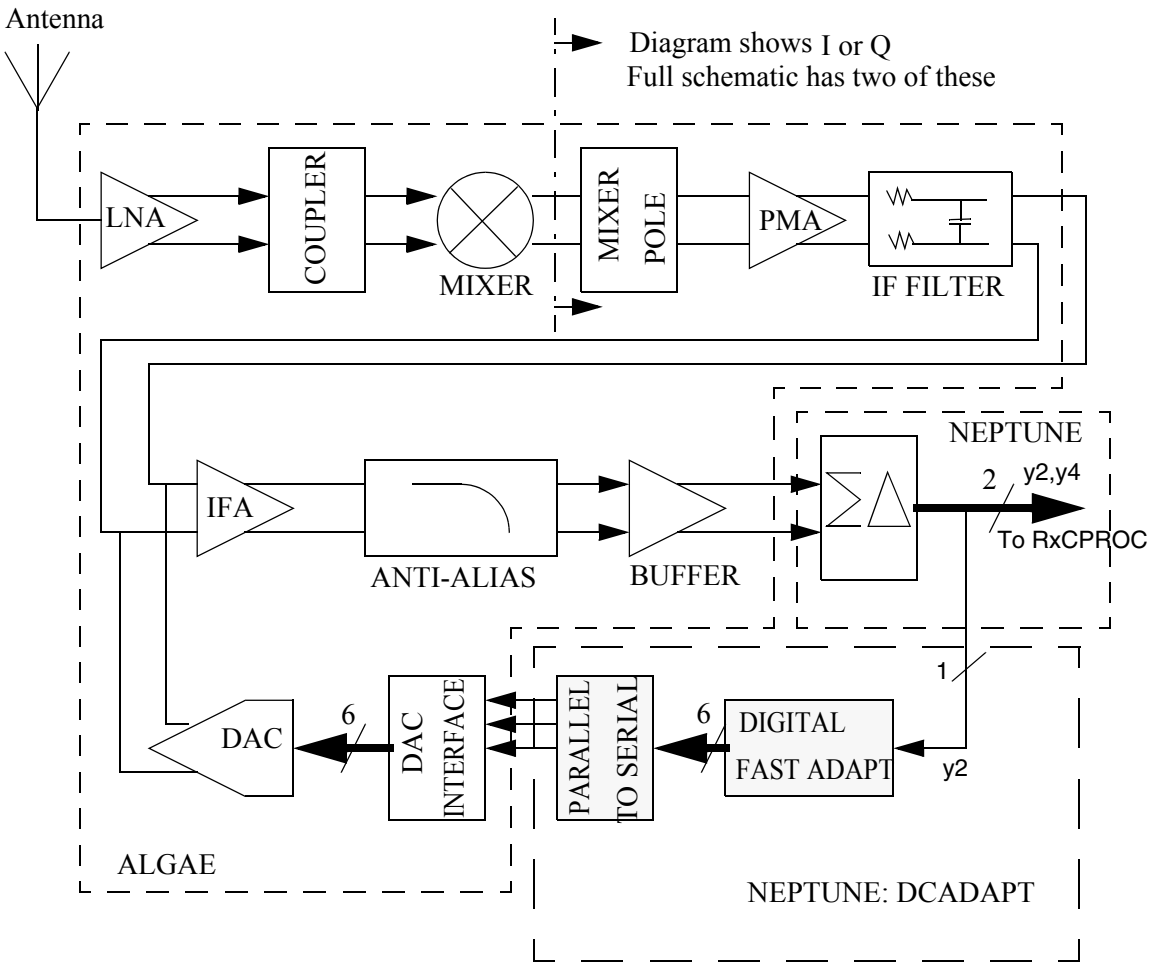


Figure 28-1. RX Lineup (Simplified)

28.2 Input/Output Pins

Figure 28-2 shows the I/O for the DCADAPT module.

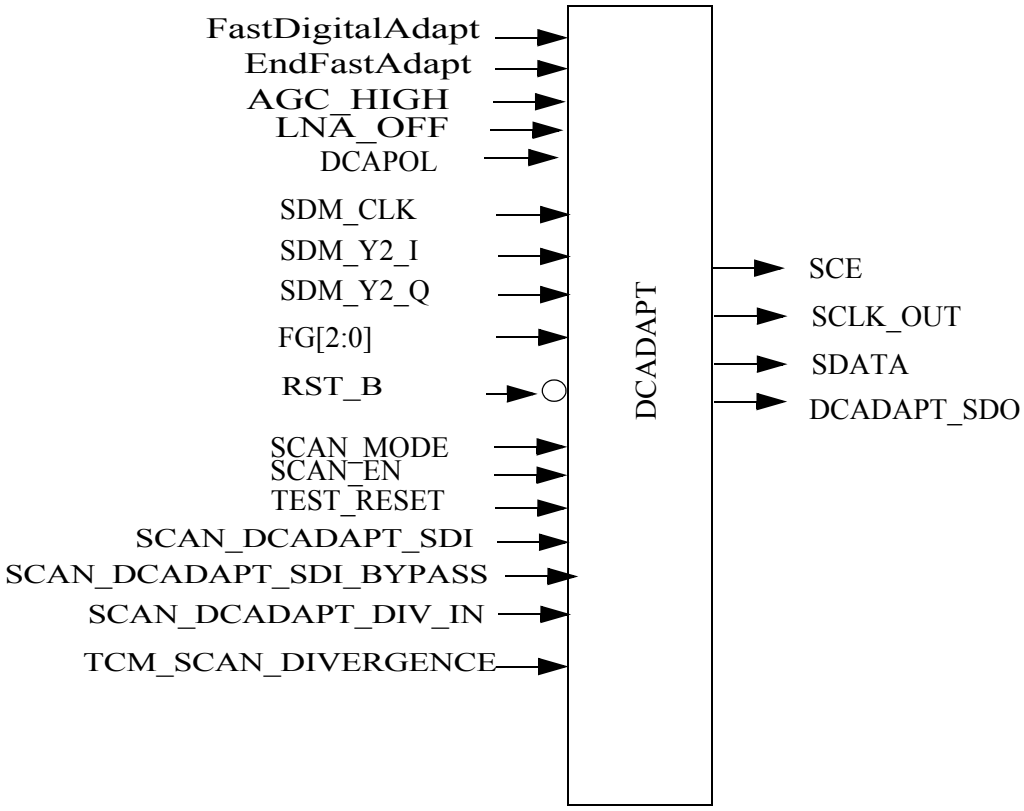


Figure 28-2. DCADAPT I/O

28.2.1 DCADAPT Module Pin List

Table 28-1 gives a description of the I/O pins on the DCADAPT module.

Table 28-1. DCADAPT Module Pin List

<table>
<tr><th>Pin Name</th><th>Direction</th><th>Description</th><th>Reset</th></tr>
<tr><td>fastdigitaladapt</td><td>input</td><td>1: Operate DCADAPT loop; 0: Hold previous DAC value.</td><td>1’h0</td></tr>
<tr><td>endfastadapt</td><td>input</td><td>From A2DIGL. Signals end of adapt phase.</td><td>1’h0</td></tr>
<tr><td>agc_high</td><td>input</td><td>From A2DIGL. Value sent via SPI to Algae to program AGC.</td><td>1’h0</td></tr>
<tr><td>lna_off</td><td>input</td><td>From A2DIGL. Value sent via SPI to Algae to switch off LNA.</td><td>1’h0</td></tr>
<tr><td>dcapol</td><td>input</td><td>From A2DIGL. 1: Invert incoming SDM_Y2. 0: no change.</td><td>1’h0</td></tr>
<tr><td>sdm_clk_mxd</td><td>input</td><td>Clock input from sigma-delta modulator (13MHz). (Muxed with test_clk)</td><td>NA</td></tr>
<tr><td>sdm_y2_i</td><td>input</td><td>Channel I sigma-delta output (from first stage of sigma-delta).</td><td>NA</td></tr>
<tr><td>sdm_y2_q</td><td>input</td><td>Channel Q sigma-delta output (from first stage of sigma-delta).</td><td>NA</td></tr>
<tr><td>fg[2:0]</td><td>input</td><td>Program bits for gain of LP filter in digital adapt loop (from SPI).
<table>
<tr><th>FG[2:0]</th><th>Gain</th></tr>
<tr><td>000</td><td>2^{-5}</td></tr>
<tr><td>001</td><td>2^{-6}</td></tr>
<tr><td>010</td><td>2^{-7}</td></tr>
<tr><td>011</td><td>2^{-8}</td></tr>
<tr><td>100</td><td>2^{-9}</td></tr>
<tr><td>101</td><td>2^{-10}</td></tr>
<tr><td>110</td><td>2^{-11}</td></tr>
<tr><td>111</td><td>2^{-12}</td></tr>
</table>
nominal</td><td>3’h3</td></tr>
<tr><td>rst_b</td><td>input</td><td>Reset pin from Chip Reset (CCM).</td><td>Reset</td></tr>
<tr><td>scan_mode</td><td>input</td><td>Scan Test mode.</td><td>NA</td></tr>
<tr><td>scan_en</td><td>input</td><td>Scan enable.</td><td>NA</td></tr>
<tr><td>test_reset</td><td>input</td><td>Reset input. Used during scan mode.</td><td>NA</td></tr>
<tr><td>scan_dcadapt_sdi</td><td>input</td><td>Data input for scan test mode.</td><td>NA</td></tr>
<tr><td>scan_dcadapt_sdi_bypass</td><td>input</td><td>Scan Bypass</td><td>NA</td></tr>
<tr><td>tcm_scan_divergence</td><td>input</td><td>Scan Divergence</td><td>NA</td></tr>
</table>

Table 28-1. DCADAPT Module Pin List

Pin Name	Direction	Description	Reset
scan_dcadapt_div_in	input	scan divergence in	NA
dcadapt_sdo	output	Data output in scan test mode.	NA
sce	output	SPI Chip Enable - to SPI MUX in Control Interface (A2DIGL).	NA
sclk_out	output	SPI Clock - to SPI MUX in Control Interface (A2DIGL).	NA
sdata	output	SPI Serial Data for DACs on Algae (to A2DIGL).	NA

28.3 Block Diagram

Figure 28-3 shows a block diagram of the DCADAPT and its connections to the Control Interface on Neptune and how it interfaces with the DACs on Algae.

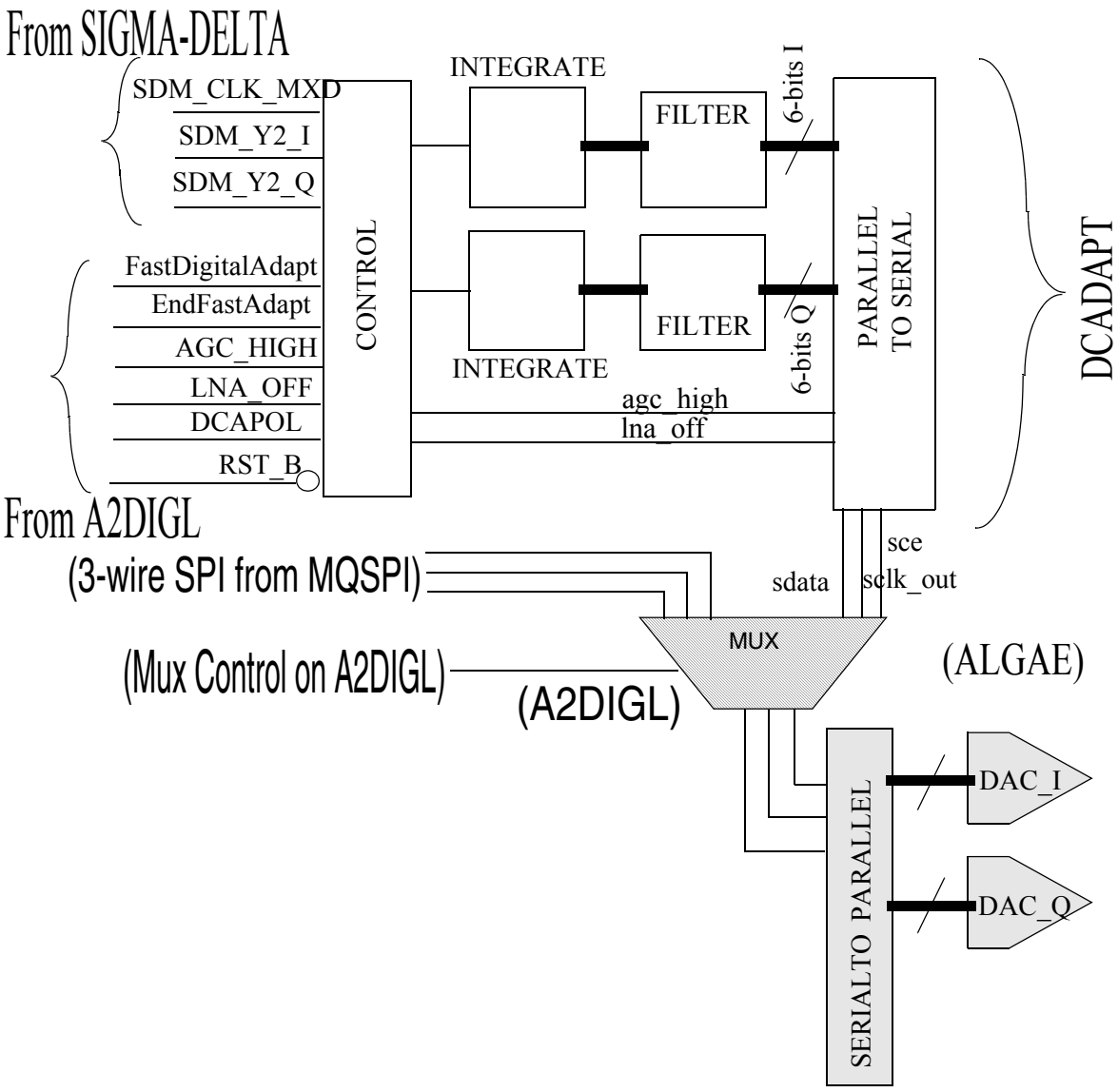


Figure 28-3. DCADAPT: Block Diagram

28.3.1 Interface to Algae

The Control Interface module on Neptune (A2DIGL) decodes serial data from the MQSPI, converts it to parallel format and sends the data to the required module. Since the DCADAPT loop has to be fast, special provision is made for data that has to be sent to the Algae: any data from the MQSPI which has to go to the Algae is immediately steered, unmodified, towards Algae's 3-wire SPI bus. If Neptune is in DC adapt mode (FastDigitalAdapt asserted), then the DCADAPT module takes over Algae's SPI bus via a multiplexer in the Control Interface (A2DIGL).

During the fast DC adapt phase the DCADAPT will continuously send data to the Algae IC in 17-bit data packets (see Figure 28-4). These bits are described in Table 28-2. If the address field of the 17-bit data-stream contains 2'b10 (address field corresponds to bits 15,14, or second and third bits to be sent), then the data is considered to be DCADAPT information and the DC offset correction data (bits 0 through 11) is then immediately directed to the DACs. Consequently, any data stream whose address field contains 2'b10 and which is not precisely 17 bits long will be ignored by Algae. The DCADAPT is designed to always send complete 17-bit data-streams.

As shown in Table 28-2, bits 13 and 12 (fourth and fifth bits to be sent) reflect the states of pins AGC_HIGH and LNA_OFF. Provision has been made to turn the attenuation of Algae's AGC to maximum and to switch off Algae's LNA when these bits are set. If they are cleared, then the AGC gain and the LNA setting are as programmed in Algae's control register. During DC adapt phase, A2DIGL will provide the values of these two bits and DCADAPT will reflect these values in the SPI bit-stream. At the end of the DC adapt phase, the bits AGC_HIGH and LNA_OFF will be cleared, EndFastAdapt will be asserted and DCADAPT will resend the last 17-bit data-stream but this time with bits 13 and 12 reflecting the new values of AGC_HIGH and LNA_OFF.

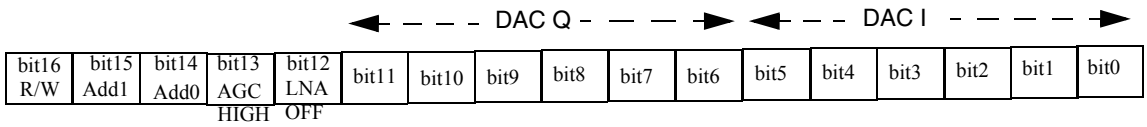


Figure 28-4. SPI bit positions

Table 28-2. SPI bit description

Bit Number	Name	Description
0	DC_DAC_I[0]	DC Offset Correction for I Channel DAC, bit 0
1	DC_DAC_I[1]	DC Offset Correction for I Channel DAC, bit 1
2	DC_DAC_I[2]	DC Offset Correction for I Channel DAC, bit 2
3	DC_DAC_I[3]	DC Offset Correction for I Channel DAC, bit 3
4	DC_DAC_I[4]	DC Offset Correction for I Channel DAC, bit 4
5	DC_DAC_I[5]	DC Offset Correction for I Channel DAC, bit 5
6	DC_DAC_Q[0]	DC Offset Correction for Q Channel DAC, bit 0
7	DC_DAC_Q[1]	DC Offset Correction for Q Channel DAC, bit 1
8	DC_DAC_Q[2]	DC Offset Correction for Q Channel DAC, bit 2
9	DC_DAC_Q[3]	DC Offset Correction for Q Channel DAC, bit 3
10	DC_DAC_Q[4]	DC Offset Correction for Q Channel DAC, bit 4
11	DC_DAC_Q[5]	DC Offset Correction for Q Channel DAC, bit 5
12	AGC_HIGH	Reflects state of AGC_HIGH signal from A2DIGL
13	LNA_LOW	Reflects state of LNA_LOW signal from A2DIGL

Table 28-2. SPI bit description

Bit Number	Name	Description
14	Add0	LSB address bit. Always set to 0 in order to address DACs
15	Add1	MSB address bit. ALways set to 1 in order to address DACs
16	R/W	Set to 1 to write to DACs on Algae. Clear to 0 to read from Algae.

28.4 Fast DCADAPT Algorithm

The DC adapt loop is a mixed-mode circuit implementation comprising a digital delaying integrator (digital accumulator decrement/increment depending on the state of the one-bit quantiser from the sigma-delta) and a first order digital filter.

The digital delaying integrator has a transfer function:

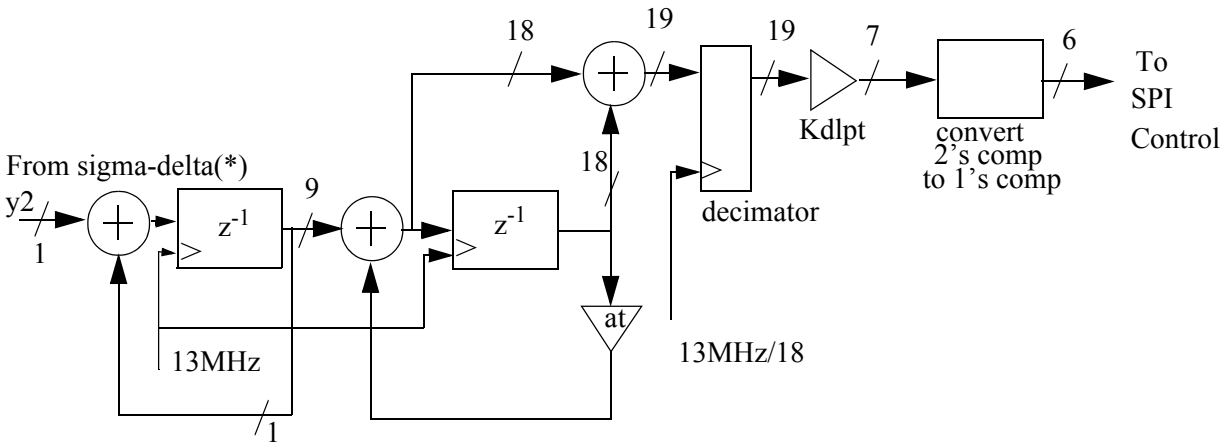
$$H(z) = z^{-1}/(1-z^{-1}) \qquad \textit{Eqn. 28-1}$$

The first order digital filter has the transfer function:

$$H(z) = Kdlpt(1+z^{-1})/(1-at*z^{-1}) \qquad \textit{Eqn. 28-2}$$

In order to simplify its implementation, “Kdlpt” and “at” are chosen to be a power of 2, i.e. $Kdlpt=2^{-x}$ and $at=1-2^{-y}$. The value of x varies from 5 to 11 to set various high-pass corners for the overall coarse adapt loop. For a low-pass corner frequency of 66kHz, y=5. The value of x is chosen to be 5.

Figure 28-5 shows a block diagram of the digital section (up to the DACs) of the Fast DCADAPT algorithm.



(*) y2 = DCAPOL ? SDM_Y2 : ~SDM_Y2

Figure 28-5. Fast DCADAPT Algorithm (One Channel, I or Q)

In GSM/EDGE the low-pass filter would normally have operated at 13MHz with the output decimated by 13 to 1MHz and then sent to a 6-bit DAC that covers a range of +/-300mVDC. In this design however, the DAC is off-chip and so the data is sent serially via a three-wire SPI bus. The DACs are updated with new data on the falling edge of dcad_sce (SPI Chip-Enable), therefore the effective decimation rate seen by the

DACs is dependant on the number of data bits being sent in an SPI transfer. The total number of bits to be sent in one SPI transfer is 17. This means that the DACs can only be updated once every 18 cycles. This effectively fixes the decimation rate as seen by the DACs and so in this design the decimator is operated at 13MHz/18.

28.5 SPI Interface

The SPI interface consists of a state machine and a shift register. The ouputs from the two adapt algorithms (2x6 wires) are fed to multiplexers controlled by a spi-load-enable signal. The other input to the multiplexers is the shift register itself - so the register can be loaded or shifted. The adapt algorithm is started a few clock cycles after FastDigitalAdapt has been detected high. After a further 18 clock cycles the first valid data appears at the output of the adapt filter. The SPI then becomes active and spi-load-enable is activated thus allowing the 12 adapt bits to be loaded into positions 0 to 11 of the shift register. At the same time positions 12 to 16 are filled with their required values corresponding to lna_off, agc_high, address and R/W.

With the SPI active, SPI chip-enable is pulled high. The SPI clock is started - it has the opposite phase to the internal clock. The SPI clock has a latch which prevents it from generating glitches.

While the SPI is active, the data is shifted out of the shift register to the SPI serial-data wire, MSB first. The MSB is also fed back to the LSB so that by the time a complete shift has been made the data is back to where it was before shifting. Data is always shifted in blocks of 17, even if the FastDigitalAdapt signal goes away. After 17 bits of data have been sent an extra clock pulse is used to stop the SPI and pull the SPI chip-enable low. If FastDigitalAdapt is still high, then the SPI will be immediately reactivated and the next data will be sent.

At the end of the adapt phase the signal EndFastAdapt will be activated, signalling that the last-sent data has to be resent (with bit-positions 12 and 13 forced to zero). Since data was always sent in complete blocks of 17, this guarantees that when EndFastAdapt signal is received the previously-sent data is in the correct position and ready to be resent. The SPI interface can tolerate all timing relationships between FastDigitalAdapt and EndFastAdapt - that is to say they can overlap or they can be separated by several clock pulses.

28.6 Test

The DCADAPT module is a modification of the DC Offset Correction Loop that was used on the “MAGIC_LV” design. DCADAPT differs from this in that the DACs are now off-chip. The test of DCADAPT has therefore been simplified since there is no longer a requirement for an analog test: a standard scan test methodology has been adopted for DCADAPT. The DAC test provision which was included in the MAGIC_LV loop should now be transferred to Algae.

Chapter 29
Receiver Saturation Detection and SDM Dither Generation (RxSDG)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	05/07/03		Updated for LTE specification release.

29.1 Introduction

This chapter describes the Neptune receiver saturation detection function and the Neptune receiver A/D sigma-delta modulator dither generation circuit. The purpose of the saturation detection function in Neptune is to allow the RF circuit in Algae IC to amplify the incoming Rx signal as much as is tolerable. If the RF circuit amplifies the incoming signal such that exceeds the maximum input range of the Neptune receiver, then a saturation condition will be detected. Once the saturation condition is detected the MCU will be informed, via interrupt, to lower the receiver gain in the RF circuitry. It is the software's responsibility to adjust the receiver gain so that the Neptune receiver will see as large an incoming signal as its input range will allow without saturating. The receiver saturation detection circuit is a part of Automatic Gain Control (AGC) loop of the "Triton" Low Cost Architecture (LCA) receiver.

The Neptune receiver sigma-delta modulator dither generation circuit is an optional function of the receiver. The purpose of the dither function is to reduce the possibility of having unwanted inband error tones at the output of the sigma-delta modulator. There are as many pros as there are cons for using the dither circuit in the receiver sigma-delta modulator. It is up to the radio designers to decide whether the dither circuit will be activated or not. Since the dither function shares the same threshold level detection circuit as the saturation detection function, it is natural to combine the two functions into one block in circuit design.

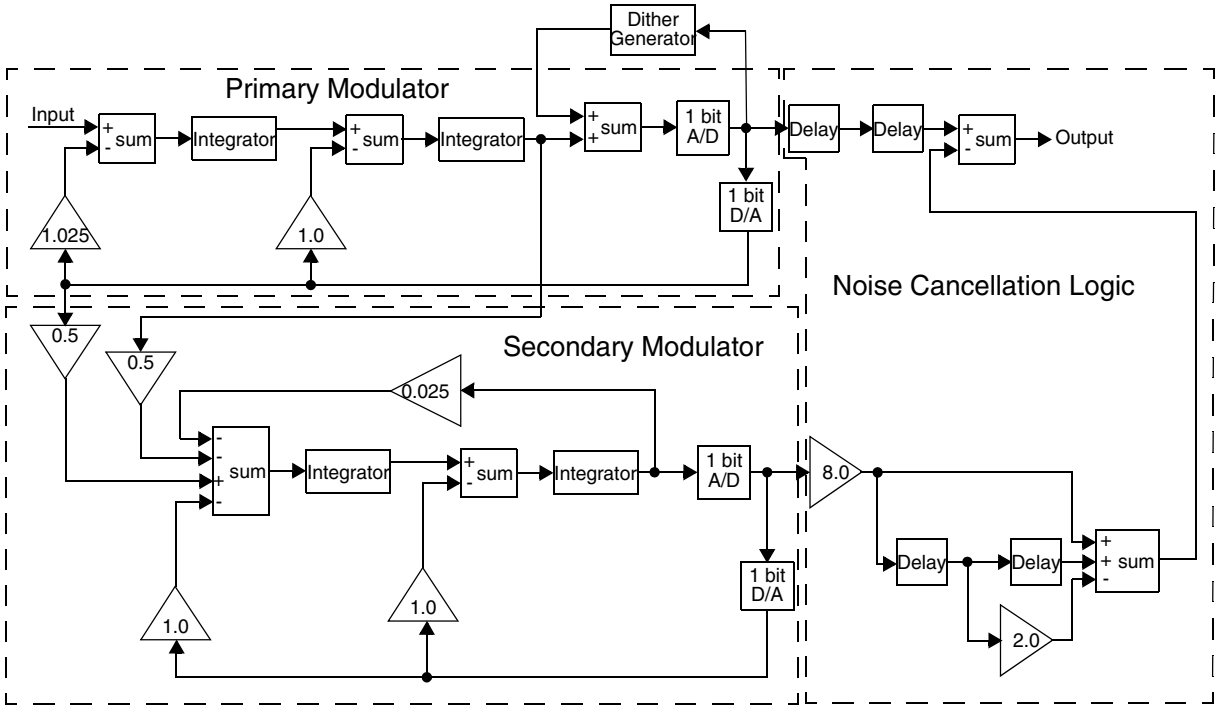


Figure 29-1. Block Diagram of the Receiver Sigma-Delta Modulator with Dither Generator

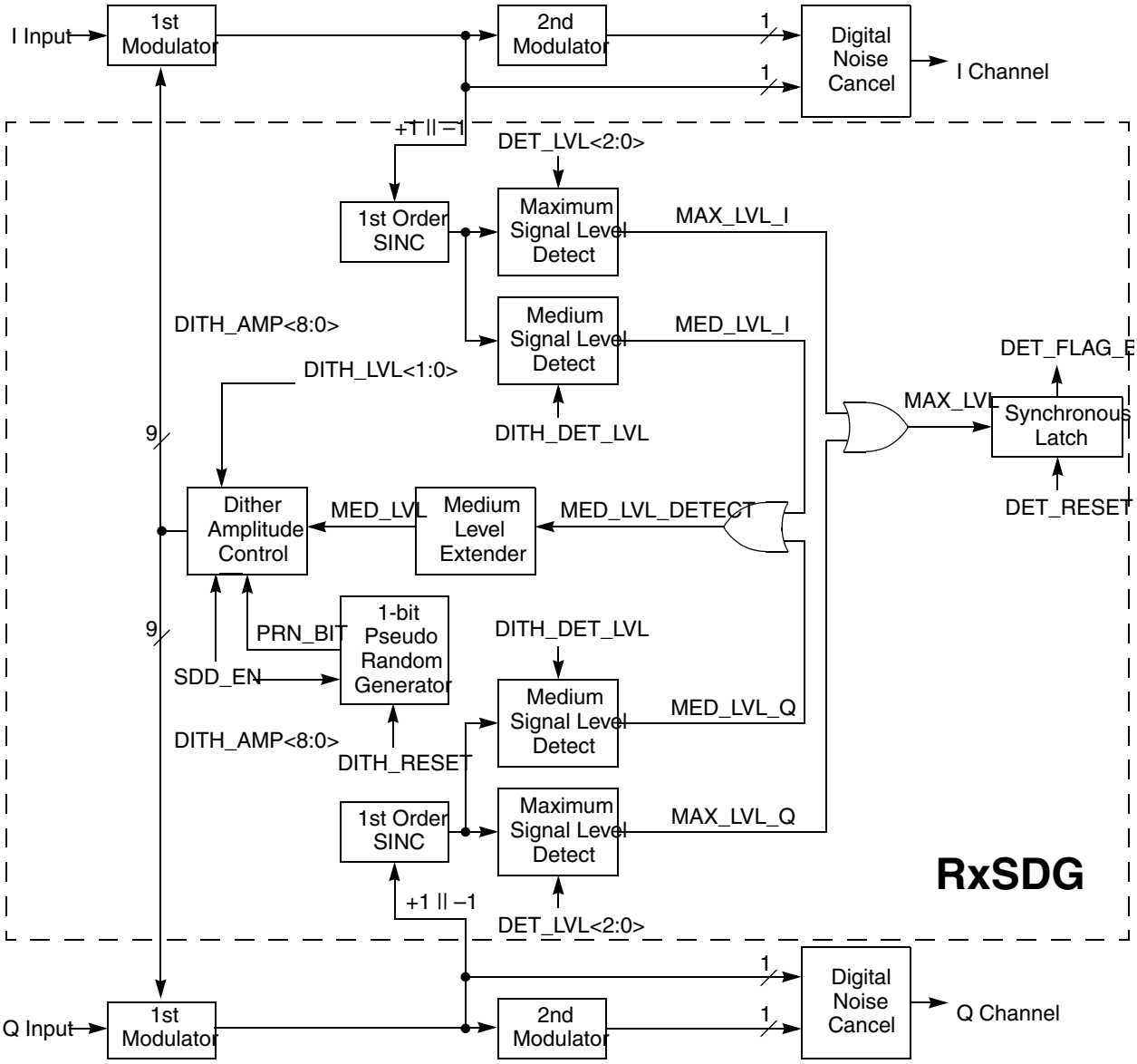


Figure 29-2. Block Diagram of the Receiver Saturation Detection and SDM Dither Generation

29.2 Input and Output Description

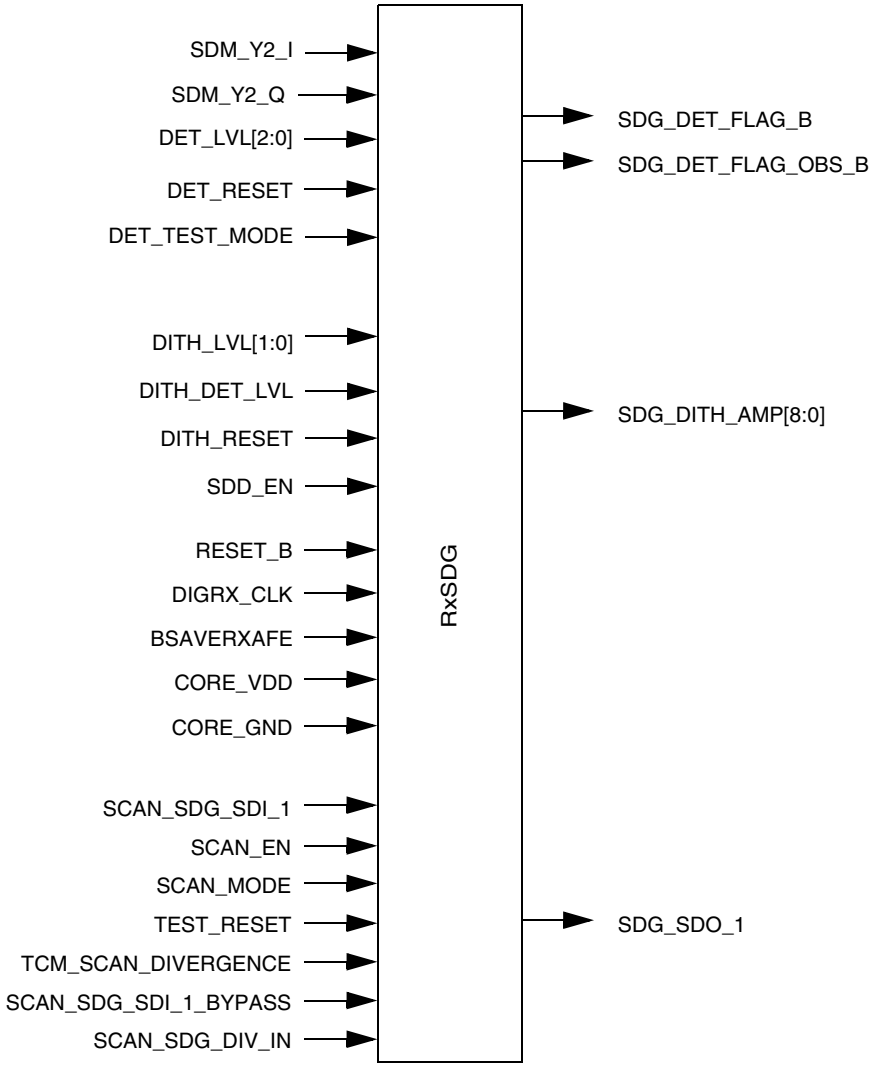


Figure 29-3.

29.2.1 RxSDG Module Pin List

Table 29-1 is a list of all the pins in the RxSDG module.

Table 29-1. RxSDG Module Pin List

Pin Name	Direction	Description	Reset Value
sdm_y2_i	input	Channel I first stage second order sigma-delta modulator output	NA
sdm_y2_q	input	Channel Q first stage second order sigma-delta modulator output	NA

Table 29-1. RxSDG Module Pin List (Continued)

Pin Name	Direction	Description	Reset Value
det_lvl[2:0]	input	Program the level comparison threshold of the amplitude detector on the sigma-delta output. DET_LVL[2:0] Threshold with respect to 250mV peak 000 -2.9dB 001-1.0dB 010+0.6dB 011+2.0dB 100~111+3.1dB	3'h3
det_reset	input	If programmed high (set) then the DET_FLAG_B pin is set. Once DET_FLAG_B is set, A2DIGL MUST clear DET_RESET within two cycles of DIGRX_CLK. Whenever DET_FLAG_B is cleared, then A2DIGL MUST set DET_RESET in order to set DET_FLAG_B.	1'h1
det_test_mode	input	If programmed high then the DET_FLAG_B will be observed at the DET_FLAG_OBS_B and the DET_FLAG_OBS_B will be connected to ATST_P through anatest block.	1'h0
sdg_det_flag_b	output	Digital output, a logic low indicates the sigma-delta input level is excessive.	1'h1
sdg_det_flag_obs_b	output	If DET_TEST_MODE is high, the DET_FLAG_B signal will pass through the DET_FLAG_OBS_B output to ATST_P pin for test.	NA
dith_lvl[1:0]	input	Selects the differential dither levels for the sigma-delta converter when the level selected by DITH_DET_LVL is not exceeded. If DITH_DET_LVL is exceeded then the digital dither pattern is disabled. DITH_LVL[1:0] Dither Level 00 +/-30mV 01 +/-60mV 10 +/-90mV 11 +/-120mV	2'h0
dith_det_lvl	input	If programmed high then the input signal threshold to remove the digital dither pattern is set at -5.4dB from the full scale of the sigma-delta input. If the bit is programmed low then the input signal threshold is set to -8.9dB from the full scale of the sigma-delta input.	1'h0
dith_reset	input	If programmed high then the Pseudo random number generator in the dither circuit will be reset. The DITH_RESET bit will self clear after outputting a pulse of two 13MHz clocks to the dither circuit.	1'h0
sdd_en	input	If programmed high then dither is enabled on the sigma-delta modulator. If programmed low then the Pseudo Random number generator will be in reset and the dither is completely disabled.	1'h0
sdg_dith_amp [8:0]	output	Dither amplitude control signals to analog sigma-delta modulator	NA
reset_b	input	Reset the RxSDG block. From chip reset (CCM).	Reset Pin

Table 29-1. RxSDG Module Pin List (Continued)

Pin Name	Direction	Description	Reset Value
digrx_clk	input	Clock used in functional mode. This is a 13MHz clock. It is coming from RxAFE block.	NA
bsaverxafe	input	When high, performs a clock gating of digrx_clk though a negative latch, as specified in Neptune DFT Methodology. When low, digrx_clk is enabled inside RxSDG module.	1'h1
core_vdd	power supply	Core digital 1.5 Volt power supply.	NA
core_gnd	power supply	Core digital ground supply.	NA
scan_sdg_sdi_1	input	Scan data input.	NA
scan_en	input	Module scan enable.	NA
scan_mode	input	If this signal is high then the block will go into scan mode.	NA
test_reset	input	Reset signal in scan mode.	NA
tcm_scan_ divergence	input	Selects scan divergence input.	NA
scan_sdg_sdi_1_by pass	input	Scan data input bypassing the block.	NA
scan_sdg_div_ in	input	Scan divergence data input.	NA
sdg_sdo_1	output	Scan data output.	NA

29.3 Receiver Saturation Detection Function

The purpose of the receiver saturation detection circuit is to detect excessive input signals at the Neptune front end receiver. It uses an amplitude threshold method to detect the overloading condition of the sigma-delta modulator. The threshold level can be programed to one of five different levels through DET_LVL[2:0] bits. If the saturation detection circuit detects that the input signal level (from I channel or Q channel) exceeds the programed threshold level then DET_FLAG_B bit will be asserted. Once the DET_FLAG_B is asserted, the MCU must write DET_RESET bit in order to negate the DET_FLAG_B. In DET_TEST_MODE, the DET_FLAG_B signal will pass through the DET_FLAG_OBS_B output to ATST_P output pin for testing. When the receiver saturation condition is detected, the MCU will receive an interrupt request via the DET_FLAG_B. The interrupt signals the MCU to lower the receiver gain in the Algae IC.

The first stage second order sigma-delta modulator output from the I channel and the Q channel are used to calculate the overloading condition of the receiver. The percentage of ones and zeros present in the sixteen most recent output bits from the I channel (and the percentage of ones and zeros present in the sixteen most recent output bits from the Q channel) are constantly evaluated. The DET_FLAG_B will be asserted if the percentage of ones (or zeros) from the I channel (or the Q channel) exceeds the programmed level. The programmed level is set by DET_LVL[2:0] bits. The following table shows the encoding method of the signal level.

Table 29-2. Saturation Detection Signal Level Threshold Encoding Method

Number of Ones	Percentage of Ones	Number of Zeros	Percentage of Zeros	DET_LVL[2:0]				
				000	001	010	011	100~111
0	0%	16	100%	set flag	set flag	set flag	set flag	set flag
1	6.25%	15	93.75%	set flag	set flag	set flag	set flag	no change
2	12.5%	14	87.5%	set flag	set flag	set flag	no change	no change
3	18.75%	13	81.25%	set flag	set flag	no change	no change	no change
4	25%	12	75%	set flag	no change	no change	no change	no change
5	31.25%	11	68.75%	no change	no change	no change	no change	no change
6	37.5%	10	62.5%	no change	no change	no change	no change	no change
7	43.75%	9	56.25%	no change	no change	no change	no change	no change
8	50%	8	50%	no change	no change	no change	no change	no change
9	56.25%	7	43.75%	no change	no change	no change	no change	no change
10	62.5%	6	37.5%	no change	no change	no change	no change	no change
11	68.75%	5	31.25%	no change	no change	no change	no change	no change
12	75%	4	25%	set flag	no change	no change	no change	no change
13	81.25%	3	18.75%	set flag	set flag	no change	no change	no change
14	87.5%	2	12.5%	set flag	set flag	set flag	no change	no change
15	93.75%	1	6.25%	set flag	set flag	set flag	set flag	no change
16	100%	0	0%	set flag	set flag	set flag	set flag	set flag

NOTE:

The DET_FLAG_B will initially be asserted after negation of the RESET_B signal due to the register initialisation in the RxSDG. This will occur until valid data has propagated through from the Sigma-Delta Converter. The DET_FLAG_B will reflect the condition of valid data by the time the ARM core has been initialised and ISR installed to service any saturation condition.

It is important to note that if we wish to prevent the converter from entering into saturation on signals near the maximum operating range, then we must have a fast acting detector. To that end, the detector utilizes a very short length SINC filter to eliminate most (but not all) of the quantization noise. Note also, the following mathematical equations can be used to describe the counting of consecutive ones and zeros.

$$H(z) = \frac{1}{16}(1 + z^{-1} + \ldots + z^{-15}) = \frac{1}{16}\left(\frac{1 - z^{-16}}{1 - z^{-1}}\right) = \frac{z^{-15/2}}{16}\left(\frac{z^{16/2} - z^{-16/2}}{z^{1/2} - z^{-1/2}}\right)$$

We know that $z = e^{j\theta}$, $\theta = \omega T$, $\omega = 2\pi f$, and $T = 1/f_s$. So, we get $z = e^{j2\pi f/f_s}$. Use this relation we can get

$$H(z) = \frac{z^{-15/2}}{16}\left[\frac{e^{j2\pi(16f)/2f_s} - e^{-j2\pi(16f)/2f_s}}{e^{j2\pi f/2f_s} - e^{-j2\pi f/2f_s}}\right]$$

We also know that $\sin x = (e^{jx} - e^{-jx})/(2j)$. So we can rewrite the equation as follows:

$$H(z) = \frac{z^{-15/2}}{16}\left[\frac{\sin\left(\frac{2\pi 16}{2f_s} \times f\right)}{\sin\left(\frac{2\pi}{2f_s} \times f\right)}\right]$$

where $f_s = 13MHz$

So the group delay of this SINC filter is 7.5 samples and the filter is fully loaded by the 16th sample. The -3 dB corner of this filter is located at about 360 kHz, and the -10 dB corner of this filter is located at about 600 kHz. The first zero of the filter is at 812.5 kHz. Clearly, prior to the sigma-delta A/D some anti-alias function is required in the Algae IC for the receiver saturation detection circuit to work properly. The circuit will not be able to detect any incoming signals at the frequency range around zeros. Figure 29-4 and Figure 29-5 show the frequency response of the SINC filter and the residual sigma-delta quantization noise at the output of the SINC filter respectively.

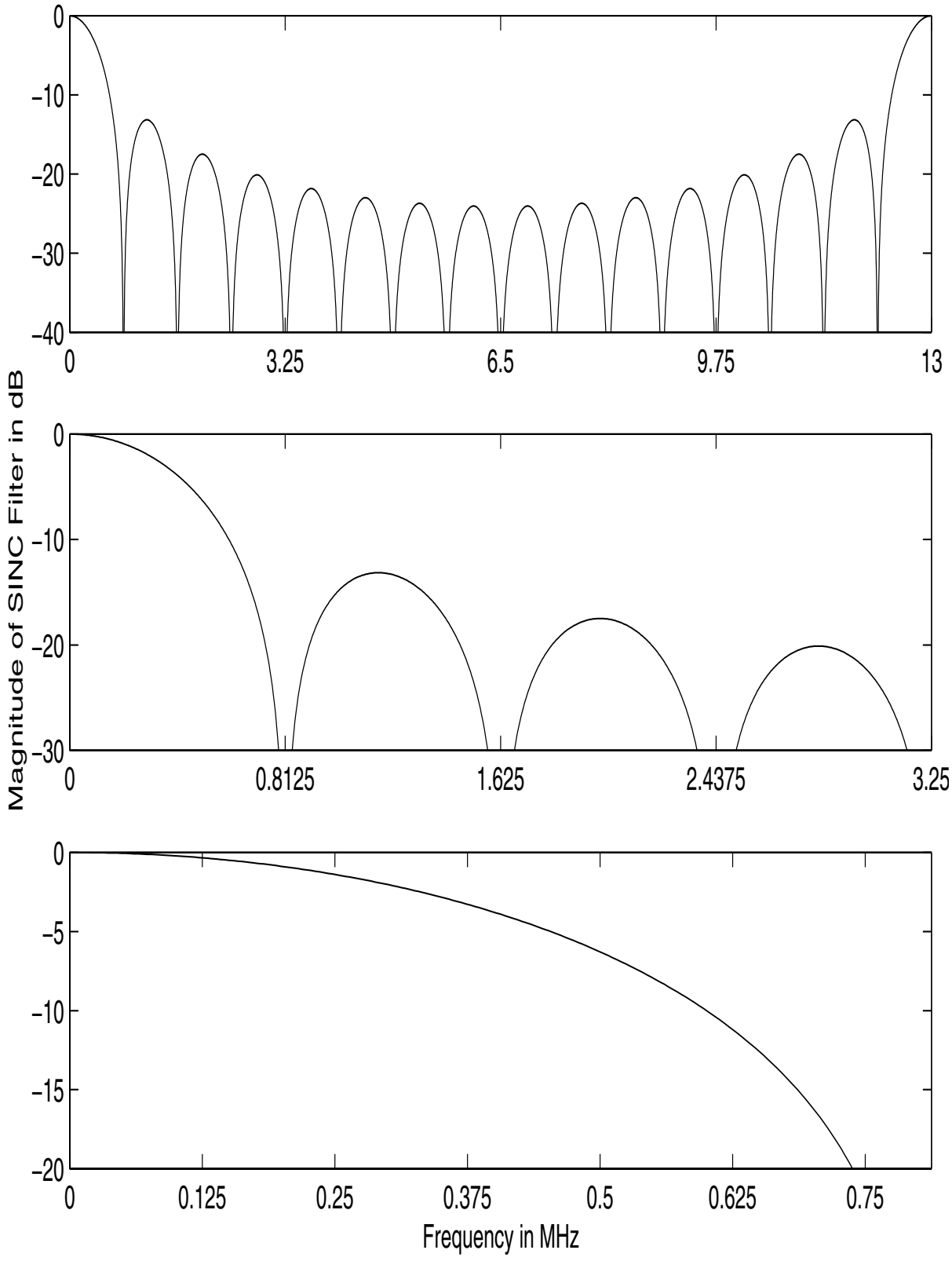


Figure 29-4. Zoomed views of the Frequency response of the SINC filter

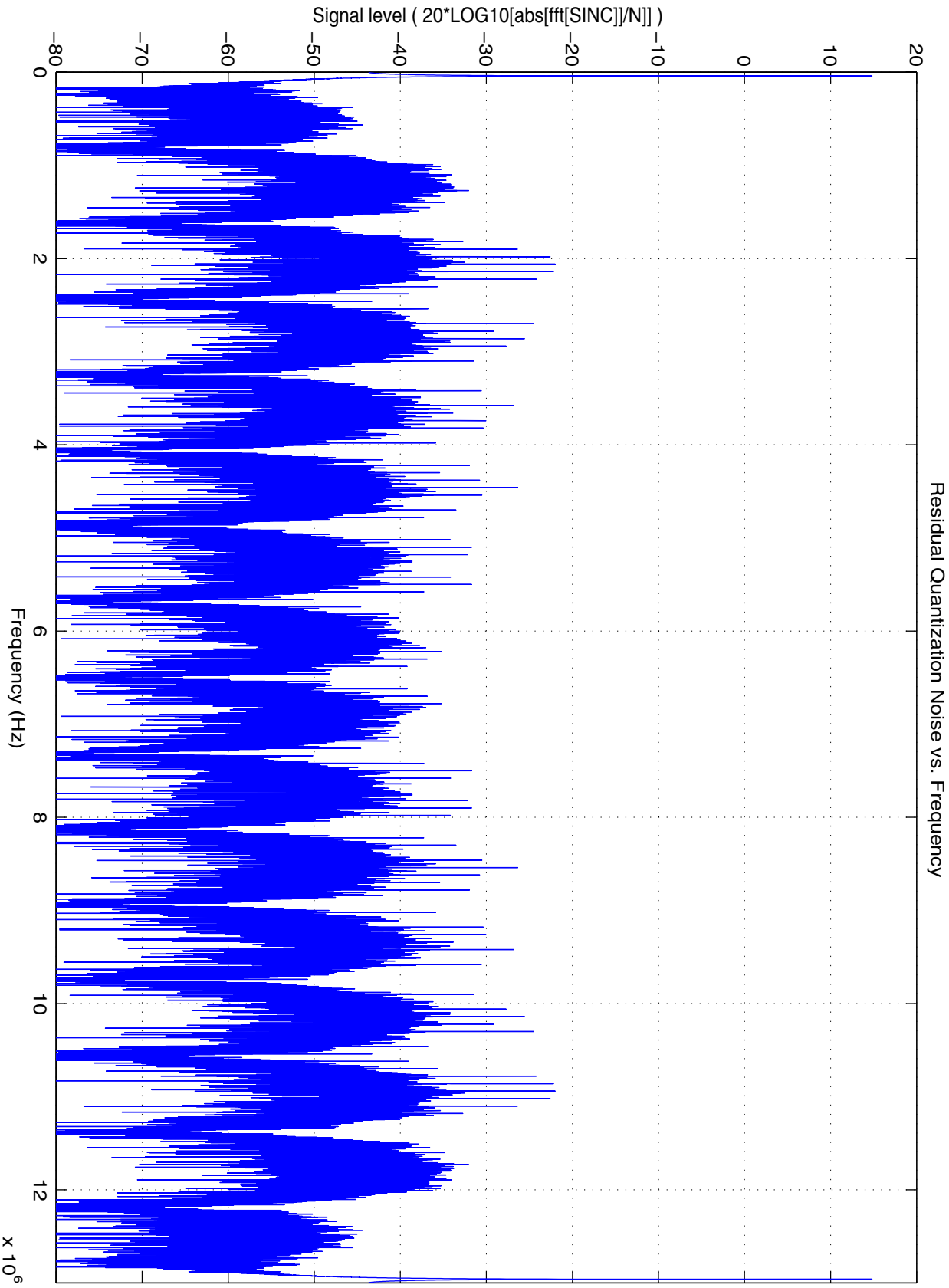


Figure 29-5. The Residual ΣΔ Quantization Noise at the output of the SINC filter

29.4 Receiver Sigma-Delta Modulator Dither Generation

The Neptune analog front end receiver uses two second-order single bit cascade sigma-delta low pass modulators to convert the received analog signal to an over-sampled digital data stream. One of the requirements for the sigma-delta encoding is that the quantization error term needs to be spectrally flat (has no strong peaks) across the frequency band of interest. But in theory, to meet this requirement, the sigma-delta modulator quantization error must be statistically independent of the input signal. Unfortunately, the quantization process is not a random process but a deterministic process. Therefore, there is no guarantee of non-decorrelation of the input signal with the sigma-delta quantization error. In fact, in literature, simulation results show that the sigma-delta quantization error can produce unwanted error tones in the frequency band of interest.

To solve this problem, it has traditionally been acceptable to add an internally generated random signal (dither) to the input of a quantiser along with the signal to be quantized. The dither signal is added to try to randomize the quantization process (“to break the tones” so to speak). In literature, it has also been shown that for an input signal whose characteristics are known, onto which we superimpose a “right kind” of dither signal, the signal to noise ratio may even be improved beyond the best theoretical performance of the A/D converter without dither. The trick is how to get a perfect dither signal.

In this dither generation circuit design, the dither signal amplitude can be programmed to +/-30mV, +/-60mV, +/-90mV or +/-120mV through DITH_LVL[1:0] bits. If the input signal level exceeds a threshold level, the dither signal amplitude will be reduced to minimum. In this case, the digital dither pattern is not used. Instead, an analog thermal noise (“analog dither signal”) is used to add to the input of the first sigma-delta quantiser along with the input signal. The threshold level of the input signal can be programmed by DITH_DET_LVL bit. The dither circuit can be enabled or disabled by SDD_EN bit. And the random number generator can be reset by DITH_RESET bit.

A 33 bit Pseudo random number generator with a primitive polynomial of $S[0] = XOR(S[1] + S[2] + S[3] + S[5] + S[7] + S[32])$ is used to generate the digital random pattern. The S[0] signal is used as a digital dither pattern. A 9 bit DITH_AMP[8:0] is used to control the analog circuit to inject 9 different levels of the analog dither signal at the input of the 1st quantiser for both I channel and Q channels. A negative logic is used at the DITH_AMP[8:0] bits. The following table shows the dither amplitude control decoding.

Table 29-3. Dither Amplitude Control Decode

SDD_EN	Input signal threshold	DITH_LVL [1:0]	random pattern bit	DITH_AMP[8:0]								
				8	7	6	5	4	3	2	1	0
0	X	X	X	1	1	1	1	1	1	1	1	1
1	0	3	1	0	1	1	1	1	1	1	1	1
1	0	3	0	1	0	1	1	1	1	1	1	1
1	0	2	1	1	1	0	1	1	1	1	1	1
1	0	2	0	1	1	1	0	1	1	1	1	1
1	0	1	1	1	1	1	1	0	1	1	1	1

Table 29-3. Dither Amplitude Control Decode (Continued)

SDD_EN	Input signal threshold	DITH_LVL [1:0]	random pattern bit	DITH_AMP[8:0]								
				8	7	6	5	4	3	2	1	0
1	0	1	0	1	1	1	1	1	0	1	1	1
1	0	0	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	0	1
1	1	X	X	1	1	1	1	1	1	1	1	0

If the input signal is below the threshold level set by the DITH_DET_LVL bit for at least 32 consecutive (13MHz) clock cycles, then the large amplitude dither signal with the digital dither pattern will be added to the input of the first sigma-delta modulator quantiser along with the input signal. Otherwise the “analog dither” scheme is used (see below). This is to ensure the large dither signal will not be used to dither a large input signal.

Very similar to saturation level calculation, the threshold level for using analog thermal noise dithering is calculated by monitoring the first stage sigma-delta modulator quantiser output. The most current 16 output bits from both I and Q channels are constantly evaluated. If the percentage of ones or zeros exceeds the programmed level, then the threshold condition will be set and the analog thermal noise will be used for dithering. Threshold level for using analog thermal noise dithering and disabling the large signal digital dither pattern is shown in the following table.

Table 29-4. Digital DIther Disable Threshold Level Encode

Number of Ones	Number of Zeros	Percentage of Ones	Percentage of Zeros	DITH_DET_LVL	
				0	1
0	16	0%	100%	threshold	threshold
1	15	6.25%	93.75%	threshold	threshold
2	14	12.5%	87.5%	threshold	threshold
3	13	18.75%	81.25%	threshold	threshold
4	12	25%	75%	threshold	threshold
5	11	31.25%	68.75%	threshold	not threshold
6	10	37.5%	62.5%	not threshold	not threshold
7	9	43.75%	56.25%	not threshold	not threshold

Table 29-4. Digital DIther Disable Threshold Level Encode (Continued)

Number of Ones	Number of Zeros	Percentage of Ones	Percentage of Zeros	DITH_DET_LVL	
				0	1
8	8	50%	50%	not threshold	not threshold
9	7	56.25%	43.75%	not threshold	not threshold
10	6	62.5%	37.5%	not threshold	not threshold
11	5	68.75%	31.25%	threshold	not threshold
12	4	75%	25%	threshold	threshold
13	3	81.25%	18.75%	threshold	threshold
14	2	87.5%	12.5%	threshold	threshold
15	1	93.75%	6.25%	threshold	threshold
16	0	100%	0%	threshold	threshold

In theory, applying a dither signal to the input of the sigma-delta modulator may improve the receiver's performance. In practice, the added "random" dither signal may increase the total noise in the band of interest of the receiver. Especially for the high order and high precision A/D that we are designing, dither may even do more harm than good. **So, it is extremely important that we will make 100% sure that dither function can be completely disabled.** To completely disable the dither function, the SDD_EN bit need to be programmed low to set the DITH_AMP[8:0] control signals to all high state.

There are three things in the receiver A/D design which reduce the possibility of having unwanted error tones at the output of the receiver. First, when the receiver is in receive mode, the input signal will never be a DC signal. So, the DC input induced unwanted error tones will not appear at the output. Second, there are two 2nd order modulators in cascade configuration in the receiver A/D. Even if there might be some unwanted quantization error tones coming out of the primary modulator, the secondary modulator will measure the quantization error from the primary modulator and subtract the error in the digital noise cancellation logic. Third, the analog circuit has many mitigating properties, such as flicker noise, thermal noise, finite amplifier gain, comparator input offset, amplifier input offset, clock jitter noise, finite signal cross-talk rejection, and finite power supply rejection. These analog circuit properties randomize the quantization error just like the dither circuit does. Any amount of unwanted tones that slip through the two modulators because of mismatch between the primary and secondary modulators in layout will be reduced by the analog circuit noise.

Therefore, in a real phone application, the receiver sigma-delta modulator dither function may be disabled. Just in case there are some unwanted tones which get through in spite of the above three quantization error tone reduction mechanisms, the dither circuit may be used to further eliminate the unwanted tones. The dither circuit here acts as a "just in case" function of the Neptune receiver design.

29.5 Testability

The DET_FLAG_B status may be monitored on external pins for the purpose of testing and evaluation.

By setting the ANA_CNTL & ANA_DATA1 registers appropriately (see ANATEST section 64.3 for more details), the RxSDG det_test_mode input can be driven high to enable the DET_FLAG_OBS_B and output the flag on the ATST_P pad.

The DET_FLAG_B may also be observed directly on the INT0 pad by appropriately configuring the Port A in the GPIO (see sections 2.2 & 2.5 for details).

There is presently no way to observe the DITH_AMP output directly on external pins. The effect of the dither may be subjectively assessed on the outputs of the Sigma-Delta Converter.

Chapter 30
Transmit (TX)

Revision History

Revision	Date	Author	Changes
0.1	8-Sept	David Redmond, Morgan Fitzgibbon, Conor O' Keeffe	Draft Release.
0.2	11-sept	David Redmond, Morgan Fitzgibbon, Conor O' Keeffe	Draft Release. some typos on the initial release
0.3	27-Sept-2000	Kathy Flories, David Redmond	Spec Review Meeting Added blackbox pin diagram, and current consumption
0.4	12-jan-01	David Redmond	Update Spec ADD SSI_SIG, and DN_DLY
0.5	09-Feb-01	Morgan Fitzgibbon David Redmond	Updated Prescaler/Phase Detector and Charge Pump Sections.
0.6	15-Feb-01	Shannon Osgood	Updated Pin List table format, and made updates per DDTS action item ID# DSPhl10496
0.7	10-Apr-01	Shannon Osgood	Added V2.0 updates.
0.8	20-Apr-01	Paul Kelleher	Add diagram and explanation for fn_mode and dp_mode usage. Made updates per DDTS action item ID# DSPh11103, DSPh11218 and DSPh10971
0.9	22-May-01	Paul Kelleher	Made updates per DDTS action item ID# DSPhl11573,
1.0	25-May-01	Paul Kelleher	Made updates per DDTS action item ID# DSPhl11820,
1.1	30-July-01	David Redmond	Updated ACC0 description from high to low.
1.2	15-Aug-01	Shannon Osgood	Update per DDTS action items #DSPhl12240, TLSbo18286, DSPhl12197. Also verified pin names against stub file.
1.3	23rd-Aug-01	Morgan Fitzgibbon	Updated the Prescaler/Phase Detector and Charge Pump sections.
1.4	4th-Sept-01	Shannon Osgood, Bob Wollscheid	Updated per DDTS#DSPhl12310

Revision History

Revision	Date	Author	Changes
1.5	10/04/01	Joseph Chan, Greg Black	Updated per DDTS#DSPhl12380 and updated synthesizer noise req.
1.6	9th of October, '01.	Morgan Fitzgibbon	Updated Figures 29-4 and 29-5 Updated Programming Example for (PCS/DCS - TX_VCO_DIV2=1 modes). Changed the min. pulse width from 11ns to 7ns as per DDTS#DSPhl12464 DDTS#TLSbo18286 DDTS#DSPhl12278
1.7	16 November 01	Shannon Osgood	Updated per DDTS# DSPhl12594 and DSPhl12595
1.8	14 January 02	David Babin, Shannon Osgood	-Updated per DDTS# DSPhl12819 and DSPhl12818. -Updated Table 29-8 per Morgan Fitzgibbon. -Changed description for cb2p4_tx to "not used". -Removed Table 29-7.
1.9	4/30/02	Morgan Fitzgibbon Shannon Osgood	Updated per DDTS# DSPhl14243.
2.0	05/03/02	Morgan Fitzgibbon Paul Kelleher Shannon Osgood	Updated per DDTS# DSPhl14237/DSPhl14449 and Morgan Fitzgibbon.
2.1	06/12/02	Morgan Fitzgibbon, Shannon Osgood	Updated per DDTS# DSPhl14707/DSPhl15078/ DSPhl14447
2.2	08/28/02	Daniel O'Hare	DP LTS updates per DDTS#s DSPhl13241, DSPhl13191, DSPhl12576
2.3	10/16/02	Shannon Osgood	Updated per DDTS# DSPhl15335.
2.4	11/12/02	Alan Casey, Shannon Osgood	Updated per DDTS# DSPhl15745/DSPhl15746. Add pac_dig_edge_mode signal for ULS/LTE only.
2.5	12/05/02	Mark Babcock, Shannon Osgood	Updated per DDTS# DSPhl15800.
2.6	04/02/03	David Babin, Shannon Osgood	Updated per DDTS# DSPhl16033 and DSPhl16594.
2.7	05/07/03		Updated for LTE specification release and DDTS# DSPhl17557.

30.1 Introduction

Several different methods of transmission are defined in Neptune, GSM normal mode, GSM Dual Port mode, and GSM predistortion mode, DCS/PCS mode.

In GSM normal mode Neptune will use an internal look-up table to trace out frequency versus time corresponding to GMSK modulation. The modulator will use the present data bit and the previous three data bits and will perform the required differential encoding of the transmit data prior to transmission.

In GSM Dual Port mode the data signal going to the FracN is converted into an analog representation and passed to an external pin. It is ac coupled to the VCO adding the higher freq components of the modulation which may have been attenuated in the main PLL path.

In GSM predistortion mode the encoded data is passed through a predistortion filter which shapes the data in anticipation to the distortion in the loop filter of the synthesizer.

In DCS and PCS modes the Tx VCO signal in Algae is divided by 2 before being sent to Neptune. To keep the maximum synthesizer deviation to +/- 67kHz the effective deviation in the loop in Neptune is also divided by 2, giving 33.5kHz. To do this we will need to modify the digital word going into the FracN.

The TX block contains the following sub-blocks: TRSYNT (analog) OSC26M (analog), dual_port (analog) and TX_dig (digital)

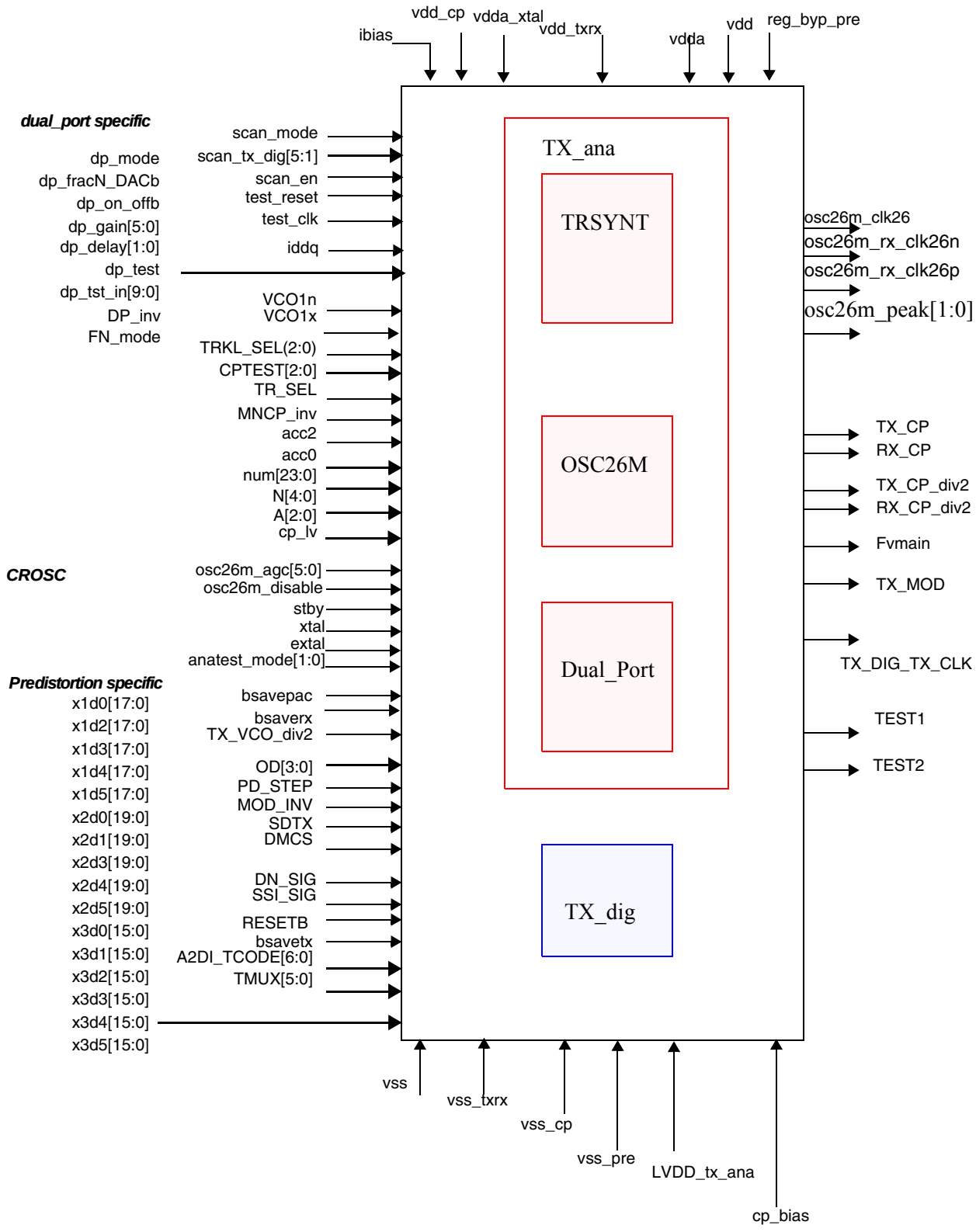


Figure 30-1. Black Box Pin List

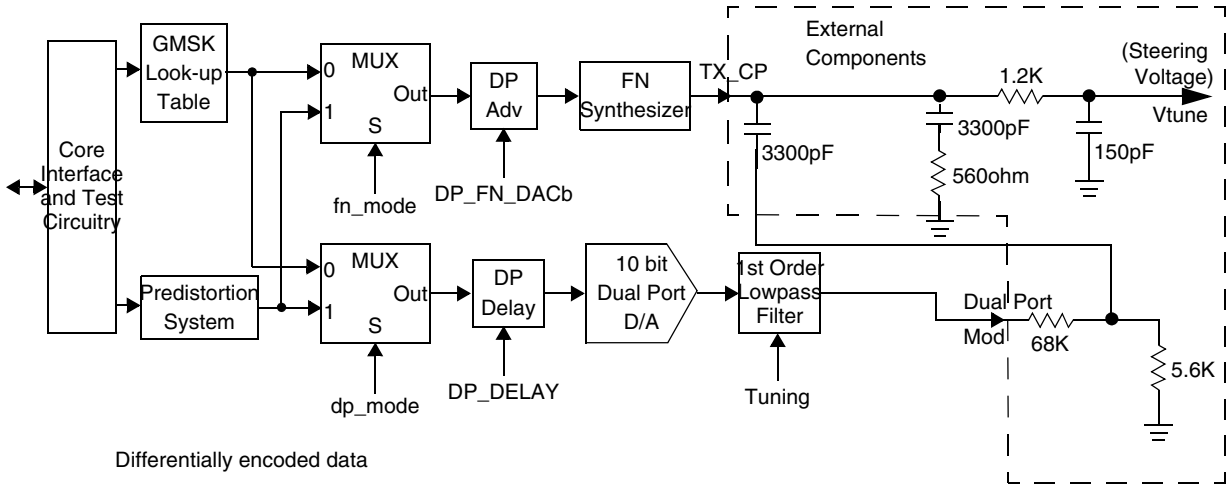


Figure 30-2. Transmit Architecture

Table 30-1. Nominal Conditions Current Drain[1]

Mode	TX Locked[2,3]	RX Locked[2,3]	bsave_TX/ bsave_RX High
4.75(Vdda_CP) +5.15/+4.5V	~2.4mA	~2.3mA	~44uA
2.5V	~2.4mA	~2.5uA	~2.5uA
1.575V (REG_BYP_PRE)+/-5%	~2.5mA	~2.5mA	~0uA
1.575(Vdd)	~2.3mA	~2.3mA	~0uA
2.775 (VDD_TXRX) +/- 4%	~2.0mA	~1.8mA	~0uA

1. Values listed are typical; the maximum will be 20% higher over temperature.
2. Conditions on the Phase Detector Charge Pump. Fvmain Leads Fref by 7ns phase error (TRKL_MODE 0)
3. Lab characterization conditions - RX locked loop at 750MHz, TX locked loop at 900MHz, trickle=0, input to prescaler approx -10dBm single-ended.

30.2 TX Module Pin List

Table 30-2 is a list of all the pins in the TX module.

Table 30-2. TX Module Pin List

Pin Name	Direction	Description	to/from	Comments
tcm_scan_mode	input	Test mode	PAD	Active high
scan_tx_dig[5:1]		Scan_chains	from TCM	active during scan_mode
test_rest		Test_Scan Reset Async	from TCM	Active during TM
test_clk		Test_Scan clock	from TCM	Active during TM
iddq		Test_iddq mode	from TCM	Active during TM
tcm_scan_en	input	Test_Scan Enable	from TCM	Active during TM
tx_fvmain	output	26 MHz clock	need to be observable	26 MHz clock is the FN output signal
a2digl_dmcs	input	Signals beginning of transmit burst	from a2digl	Active high needs to be observable
a2digl_resetb	input	Asynchronous Reset signal	Dedicated i/p pin	Put circuit in Known state Active LOW
a2digl_sdtx	input	the serial TX data	from DSP	TX_DIG_TX_CLK is the demand clock needs to be observable
tx_dig_tx_clk		270.833kHz bit clock	to DSP	TX_DIG_TX_CLK is the demand clock needs to be observable
a2digl_dp_mode	input	selects source for dual_port	from a2digl	hi=> Predistortion filter is used lo=> GMSK lookup table used
a2digl_dp_fracn_dacb	input	selects the data path to be delayed	from a2digl	hi=>no delays on the FracN path lo=>delays dual_port path
a2digl_dp_on_offb	input	enable the dual_port system	from a2digl	hi=>enabled lo=> disabled i.e no delay
a2digl_dp_gain	input	dual_Port programmable gain	from a2digl	6-bit
a2digl_dp_ofs	input	dual_port programmable offset	from a2digl	10-bit

Table 30-2. TX Module Pin List (Continued)

Pin Name	Direction	Description	to/from	Comments
a2digl_dp_delay	input	datapath delay in 230ns steps	from a2digl	2-bit 00 =>1 OSclk delay 01=> 2 OSclk delay 10 =>3 OSclk delay 11 =>4 OSclk delay
a2digl_dp_test	input	testmode for dual_port DAC	from a2digl	1-bit
a2digl_dp_tst_in	input	testmode for dual_port DAC	from a2digl	9-bits
a2digl_trkl_sel	input	alterUP trickle current to charge pumps	from a2digl	Alters the Min. Pulse Width from 10us - TRKL_SEL=000 to 20us - TRKL_SEL=111 for both TX & RX Charge Pumps. 3 bits.
a2digl_cptest	input	places change pumps in a Testmode	from a2digl	3-bit
a2digl_mod_inv	input	inverts the output of the differential encoder	from a2digl	hi=> normal encoding lo=> inverted encoded output
a2digl_mncp_inv	input	select sink or source for charge pump	from a2digl	hi=>charge pump will sink current to increase freq lo=>charge pump will source current to increase freq.
a2digl_fn_mode	input	Selects source of data for FracN	from a2digl	hi=>source is predistortion lo=>source is GMSK LUT
a2digl_dp_inv	input	selects sense for data in dual_port	from a2digl	hi=>data inverted lo=> data not inverted
a2digl_od	input	divider ratio for Oversampling clock	from a2digl	4-bit fixed to divide by 6 down counter, set to 6 for 26/6 = 4.33MHz OSclk
a2digl_acc2	input	selects number of accumulators	from a2digl	hi=>2 accumulators used lo=>3 accumulators used
a2digl_acc0	input	disables accumulators	from a2digl	Lo=>all accumulators are disabled overrides acc2 setting(INTEGER MODE)
a2digl_tx_vco_div	input	digital divide by 2 before going into FracN	from a2digl	Lo=>not active: 900MHz hi=> activated:1800,1900 MHz
a2digl_x1d0	input	predistortion coefficient (velocity)	from a2digl	18-bit

Table 30-2. TX Module Pin List (Continued)

Pin Name	Direction	Description	to/from	Comments
a2digl_x1d1	input	predistortion coefficient(velocity)	from a2digl	18-bit
a2digl_x1d2	input	predistortion coefficient(velocity)	from a2digl	18-bit
a2digl_x1d3	input	predistortion coefficient(velocity)	from a2digl	18-bit
a2digl_x1d4	input	predistortion coefficient(velocity)	from a2digl	18-bit
a2digl_x1d5	input	predistortion coefficient(velocity)	from a2digl	18-bit
a2digl_x2d0	input	predistortion coefficient(acceleration)	from a2digl	20-bit
a2digl_x2d1	input	predistortion coefficient(acceleration)	from a2digl	20-bit
a2digl_x2d2	input	predistortion coefficient(acceleration)	from a2digl	20-bit
a2digl_x2d3	input	predistortion coefficient(acceleration)	from a2digl	20-bit
a2digl_x2d4	input	predistortion coefficient(acceleration)	from a2digl	20-bit
a2digl_x2d5	input	predistortion coefficient(acceleration)	from a2digl	20-bit
a2digl_x3d0	input	predistortion coefficient(jerk)	from a2digl	16-bit
a2digl_x3d1	input	predistortion coefficient(jerk)	from a2digl	16-bit
a2digl_x3d2	input	predistortion coefficient(jerk)	from a2digl	16-bit
a2digl_x3d3	input	predistortion coefficient(jerk)	from a2digl	16-bit
a2digl_x3d4	input	predistortion coefficient(jerk)	from a2digl	16-bit
a2digl_x3d5	input	predistortion coefficient(jerk)	from a2digl	16-bit
a2digl_dp_step	input	all 0's or 1's definition for predistion	from a2digl	hi=>modulation forced to +/-67kHz lo=>hold last value
a2digl_num	input	numerator for FracN system denominator is fixed at 2^24	from a2digl	24-bit

Table 30-2. TX Module Pin List (Continued)

Pin Name	Direction	Description	to/from	Comments
n[4:0]		input for main loop fractional divider	from a2digl	5-bit
a[2:0]		input for main loop fractional divider	from a2digl	3-bit
a2digl_tr_sel	input	selects the charge pump in use the CP not selected is floating	from a2digl	hi=>CP_TX active lo=> CP_RX active
bsavepac (bsavetx)	input	low for normal operation (bsave to pac and tx equivalent)	from a2digl	battery save for TRSYNT Transmit
bsaverx		low for normal operation	from a2digl	battery save for TRSYNT receive
stby		Powers down the oscillator when asserted. Reset state is low.	from a2digl	Reset state is low.
osc26m_disable		This signal is not used.	from a2digl	
osc26m_agc[5:0]		6 bit input for AGC of osc amplitude	from a2digl	-
osc26m_peak[1:0]		2 bits to indicate current osc amplitude level	from a2digl	-
osc26m_clk26		Output of the 26 MHz clock to clock monitor, USB PLL, reference PLL, and the gated 26MHz clock block.	to cmon	-
osc26m_rx_clk26p		Positive 26 MHz clock differential output to Neptune receiver.	refosc	
osc26m_rx_clk26n		Negative 26 MHz clock differential output to Neptune receiver.	refosc	
xtal		Connecting to one side of the crystal oscillator outside of the chip.	PAD	analog
extal		Connecting to one side of the crystal oscillator outside of the chip.	PAD	analog
vddaxtal		Power supply to the OSC26M block.	supply	Power
vssaxtal		Power supply to the OSC26M block.	supply	Ground
1p575rtc		1.575 RTC supply	regulator	-
tx_tx_mod	output	TX_dual_port modulation output	to PIN	1V pk-pk

Table 30-2. TX Module Pin List (Continued)

Pin Name	Direction	Description	to/from	Comments
pad_vdd_cp	input	Charge Pump Supply	from supply	Tolerance between 4.5V and 5.15V. Nominal 4.75V. Requires 4.7uF ext. Cap
pad_vss_cp	input	Charge Pump Supply	from supply	CP_gnd
vdd_txrx		Supply for Charge Pump Current Ref.	from supply	2.775V +/-5% Requires 4.7uF ext. Cap
vss_txrx		Supply for Charge Pump Current Ref.	from supply	0V +/-5% Requires 4.7uF ext. Cap
vdd		digital supply	from supply	
vss		digital supply	from supply	
pad_vdda	input	analog supply for Dual_port 2.5V	from supply	
pad_vssa	input	analog supply for Dual_port 2.5V	from supply	
regul_reg_byp_pre	input	analog supply for Prescaler, Phase Detector, 1.575V	from supply	Tolerance +/- 5%. Requires 4.7uF ext. Cap.
LVDD_txana		1.575 RTC supply	regulators	
vddaxtal		analog supply for OSC block	from supply	
vssaxtal		analog supply for OSC block	from supply	
regul_ibias	input	Analog reference bias current 10uA	from REGULATOR	
tx_cp	output	TX charge pump	to PIN	< 26MHz (4.75V supply)
rx_cp	output	RX charge pump	to PIN	< 26MHz (4.75V supply)
tx_cp_div2	output	0 - Vdda_CP/2	to GPADC	0-2V range
rx_cp_div2	output	0 - Vdda_CP/2	to GPADC	0-2V range
pad_cp_bias	input	requires 1% 10K Ohm Resistor	to PIN	1.575 CP BIAS Voltage derived from Vdda_XTAL.
vco1n		Positive Differential i/p from VCO	from PIN	2 GHz (REG_BYP_PRE)
vco1x		Negative Differential i/p from VCO	from PIN	2 GHz (REG_BYP_PRE)
pad_vss_pre	input	prescaler gnd	form supply	1.575 supply

Table 30-2. TX Module Pin List (Continued)

Pin Name	Direction	Description	to/from	Comments
anatest_tmux	input	test mux bus	from a2digl	6-bits shared internally between TRSYNT, OSC26M, and dual_port
tx_test1	output	observation port 1	to ANATEST	if not selected tied to gnd
tx_test2	output	observation port 2	to ANATEST	if not selected tied to gnd
a2di_tcode		tuning word for analog ccts	from RC tuning	6-bit
ssi_sig		Signal used to qualify generation of tx_dig_tx_clk	from PAC	
dn_sig		Signal used to identify the end of a transmission	From PAC	
pac_dig_edge_mode	input	Edge mode enable	Pac	Behaves as in receive mode

30.3 FN_Mode, DP_Mode

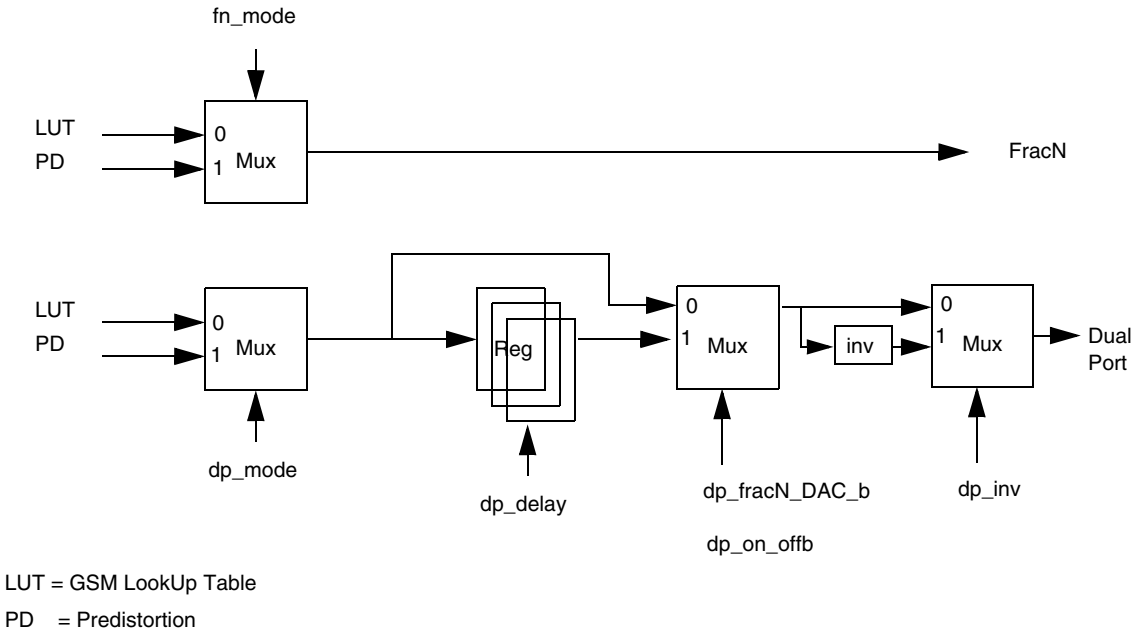


Figure 30-3. FN_Mode/DP_Mode

No delay elements in the fracN path. The input dp_francN_DAC_b does not have any effect on the fracN path.

30.4 Module Operation (Digital Modulation)

30.4.1 Normal GSM Mode

bits TX_VCO_div= 0, DP_mode=0, FN_mode=0

In this mode of operation, Neptune will use an internal look-up table to trace out frequency versus time corresponding to GMSK modulation. The modulator will use the present data bit and the previous three data bits. Neptune will perform the required differential encoding of the transmit data prior to transmission. On the following edge of DMCS, Neptune will sample a TX data bit from the SDTX signal from the DSP. All additional TX data bits are sampled from the SDTX line on every bit clock interval until DMCS goes low.The transmission of data to the power amplifier is delayed by a number of clock cycles (determined by implementation) due to the fractional divider system. The bitclock within Neptune is synchronized to the main PLL reference frequency with the phase set by the rising edge of DMCS. A gated bit clock signal is provided at TX_DIG_TX_CLK. This clock is inverted from the internal bit clock. Thus when TX_DIG_TX_CLK transitions from low to high, a new SDTX bit should be sent. Neptune will read SDTX on the high to low transition of TX_DIG_TX_CLK. TX_DIG_TX_CLK may be high or low before the rising edge DMCS. The state will depend on the state of TX_DIG_TX_CLK when DMCS went high to low on the previous burst.

The oversample divider is programmed with OD[3:0] of the SPI. Nominal programming for GSM is 6 if a 26MHz reference is used. The center frequency of the channel selection is offset by -101.562kHz so that an output of 10000H from the Look-Up ROM will correspond to a 0Hz offset.

When dn_sig (Signal used to identify the end of a transmission) goes high the data to accumulators from the GSMK lookup table should reset to 10000 hex when fn_mode is low and 20000 hex when fn_mode is high.

When dn_sig goes high, a2di_bsavetrsynt goes high asynchronously, turning off the fvmain block from the prescalar to the digital section.There will be no fvmain clock edge after dn_sig goes high.

The resetting of the data to the accumulators from the GSMK can not be done synchronously, only asynchronously.

30.4.2 Edge Mode

If the signal pac_dig_edge_mode is high, then there is no modulation.

30.4.3 GSM Dual_port Mode

TX_VCO_div=0, DP_mode=1

This mode of operation is nearly identical to the Lookup table method described above. The only difference is that a 9-bit D/A which follows the modulation look up table output waveform is output on the TX_MOD pin. This signal is then coupled into the loop filter to add in the higher frequency components of the modulation which may have been distorted in the main PLL path. This will allow the use of a lower bandwidth main PLL to improve the spectral purity of the transmit signal while maintaining near linear phase on the DATA path.

DP_GAIN will be adjusted to set the proper drive level and offset from the DAC path. DP_DELAY will set the relative delay of the two paths. DP_FN_DACb will determine which path has the added delay, either the dual_port DAC path or the FracN path.

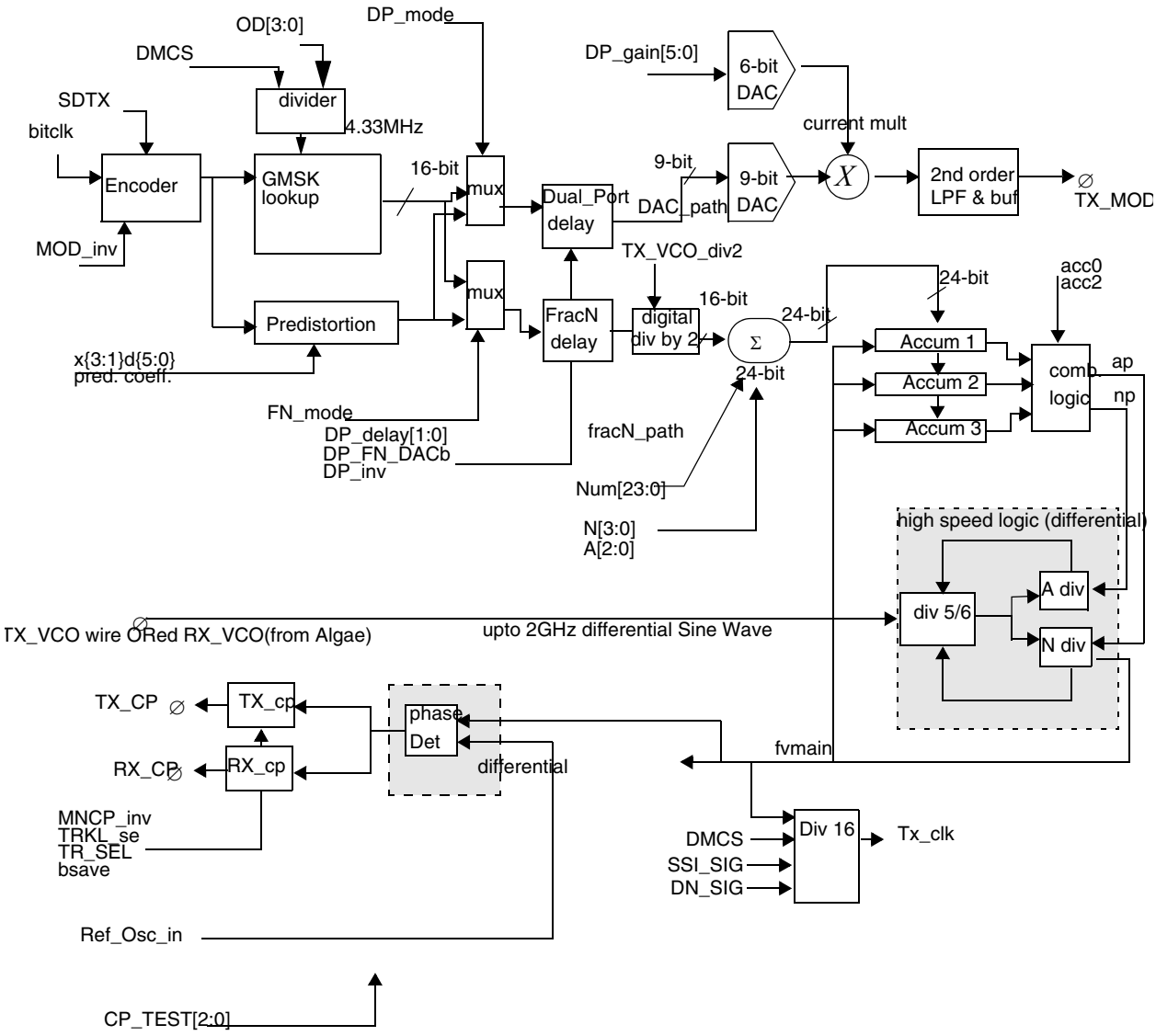


Figure 30-4. Detailed Block Diagram of Tx Data System

30.4.4 Predistortion GSM Mode:

bits TX_VCO_div= 0, FN_mode=1

In this mode of operation, Neptune will accept data as described above. The modulator will use the present data bit and the previous **five** data bits. Neptune will perform the required differential encoding of the transmit data prior to transmission. The center frequency of the channel selection is offset by 203.125kHz. This is to accommodate the 18 bit twos compliment transmit modulation word.

The shaping of the frequency versus time waveform, selected by the **six** data bits, are determined by eighteen two s compliment coefficients. These coefficients are taken directly as inputs from the A2DIGL block.

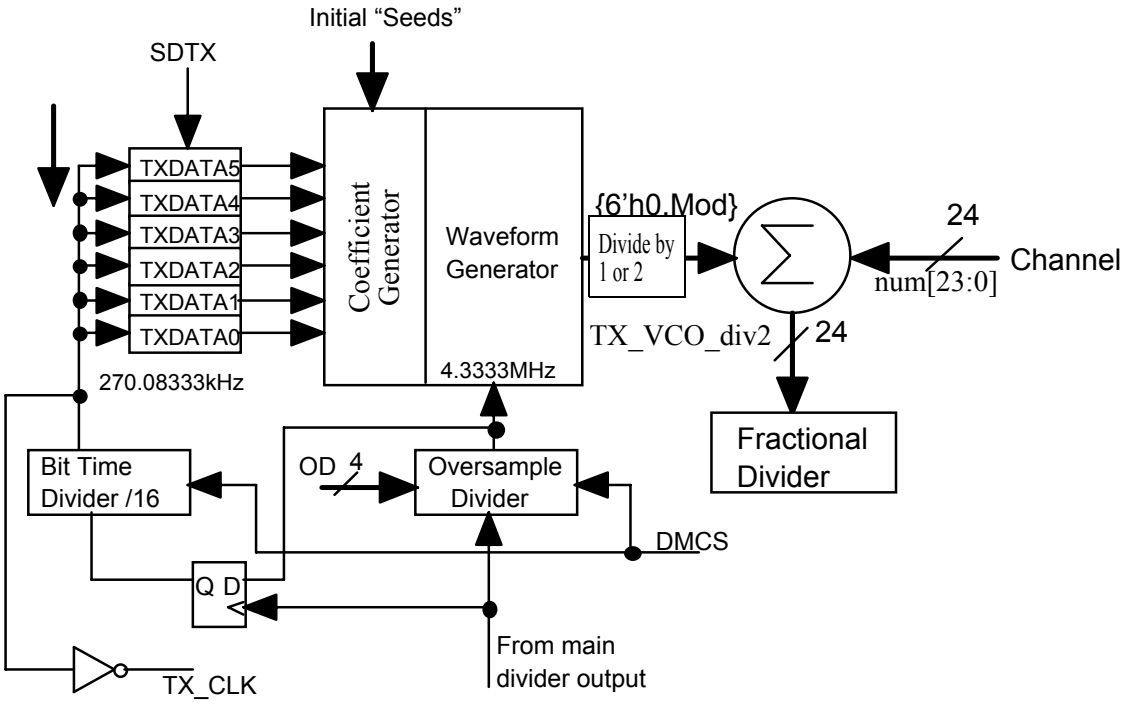


Figure 30-5. GSM Predistortion Mode

The first coefficient is x1d. This coefficient represents the first derivative of the frequency. The second coefficient is x2d. This coefficient represents the second derivative of the frequency. The third coefficient is x3d. This coefficient represents the third derivative of the frequency. When a data bit is loaded into Neptune, the six coefficients for each order of derivative of frequency are multiplied by the present data bit and the previous five data bits of the data stream respectively (using 1 for +1 and -1 for 0) and added together to form three words used to program a waveform generator. This is shown in the diagram below. Note that the adder sizes are the same as the word sizes. Care must be taken in the selection of the pre-distortion coefficients to insure that an adder overflow does not occur.

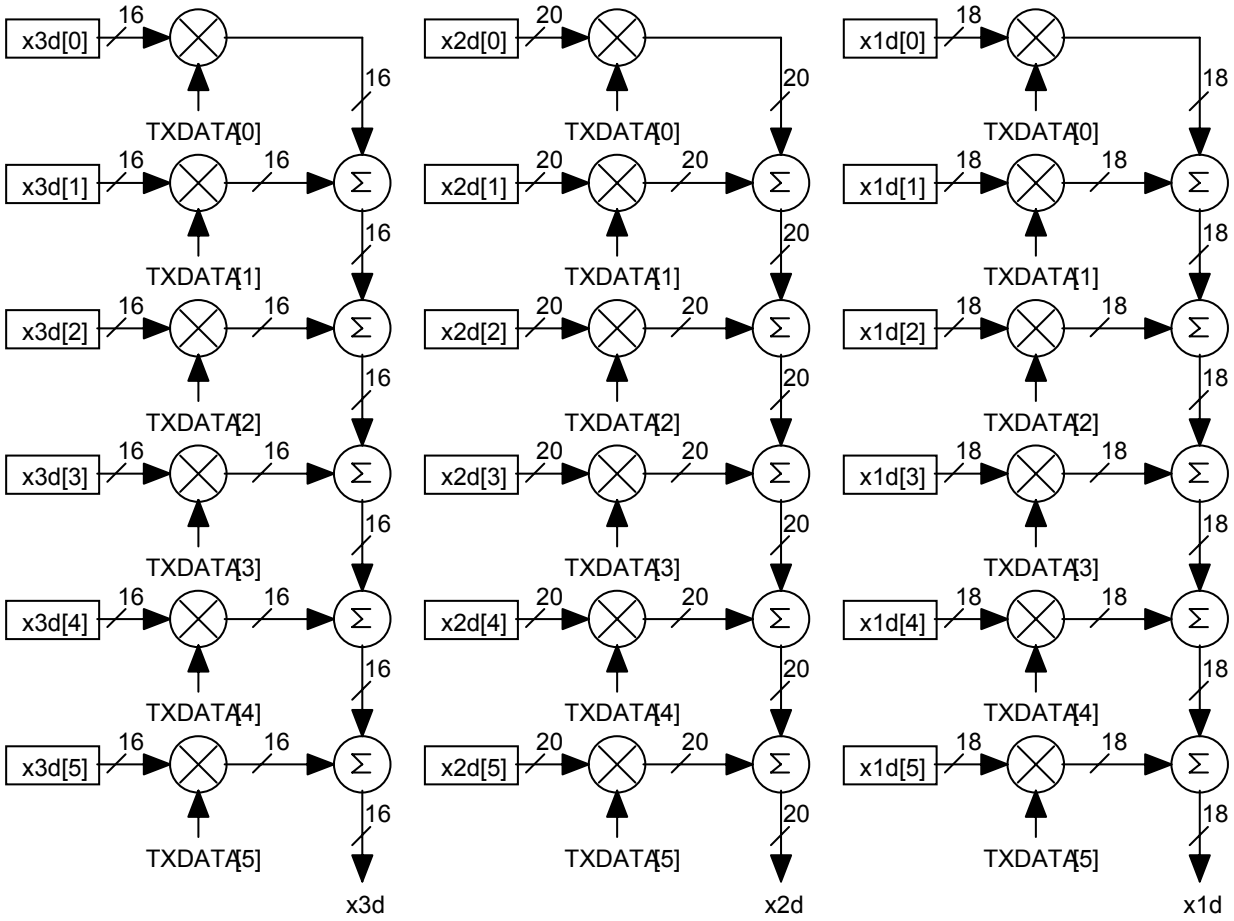


Figure 30-6. Coefficient Generator

The waveform generator consists of three accumulators. The highest order accumulator provides the tuning word to the fractional N synthesizer. The contents of this accumulator represent the frequency of the modulation. On the rising edge of DMCS, this accumulator is preloaded to 67kHz (35555H). This defines the starting point of the waveform generator for each transmission time slot. The contents of the second highest order accumulator represent the linear slope of the modulation frequency versus time (x1d) or the first derivative of the frequency versus time. The contents of the second highest order accumulator are input to the highest order accumulator. Since an accumulator performs a digital integration, the highest order accumulator will convert the derivative of frequency (x1d) to frequency. In a similar fashion, the contents of the second accumulator represent the second derivative of frequency versus time and the input to the third accumulator (x3d) represents the third derivative of frequency versus time. This creates a waveform from the frequency and the first three derivatives of the frequency.

Each lower order accumulator is fed to the next higher order accumulator with a right shift of 4 bits. Thus each position of each lower order accumulator is a factor of 16 less than the corresponding position in the next higher order accumulator. Note that x3d is only 16 bits long. This is because it is a constant input to the x2d accumulator and thus the four LSBs of x3d are truncated. Therefore, it is actually 20 bits before the shift by 4. A diagram of the bit alignment of the various pre-distortion coefficients is shown below. For a 26MHz reference, x[0] is 1.55Hz.

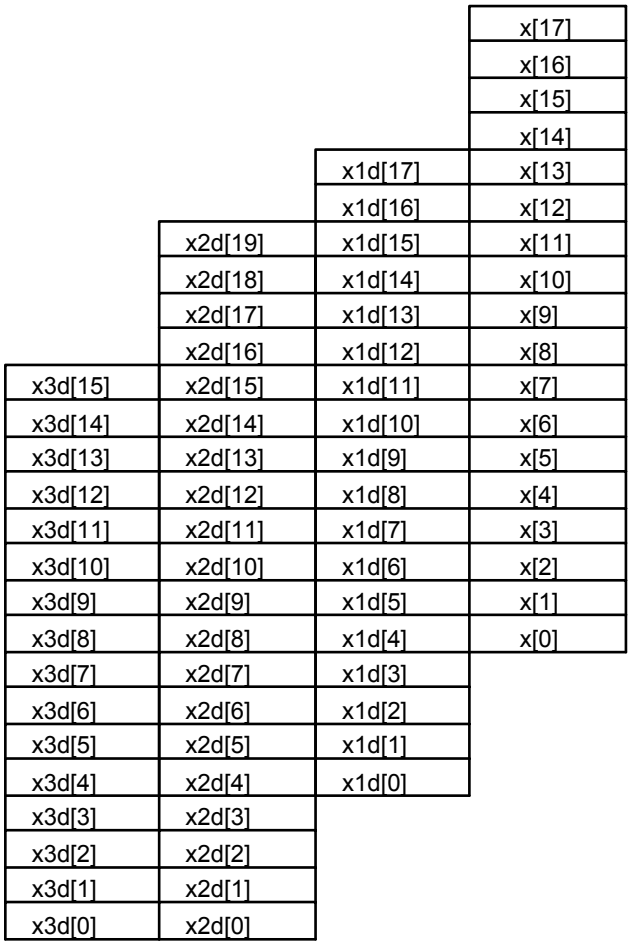


Figure 30-7. Bit Alignment of the Various Pre-distortion Coefficients

All of the accumulators are clocked at a rate of 16 times the bitclk rate. Thus if the second highest order accumulator contains a fixed positive number, the highest order accumulator is linearly incremented by this number 16 times. At this time a new data bit is received and the contents of the lower order accumulators are reloaded with new slope information.

Note that the MSB of x2d is only two bits below the MSB of x1d and that the MSB of x3d is only four bits below the MSB of x2d. This is because each higher order derivative coefficient has one more bit of range than the next lower order derivative coefficient (except xd2 which has 2 more). Care must be taken that the higher order derivatives do not overflow the lower order accumulators in the 16 clock cycle interval.

A block diagram of the waveform generator is shown below. The waveform generator is realized in two s compliment format. Upon application to the main FN system, TW is converted to unsigned format.

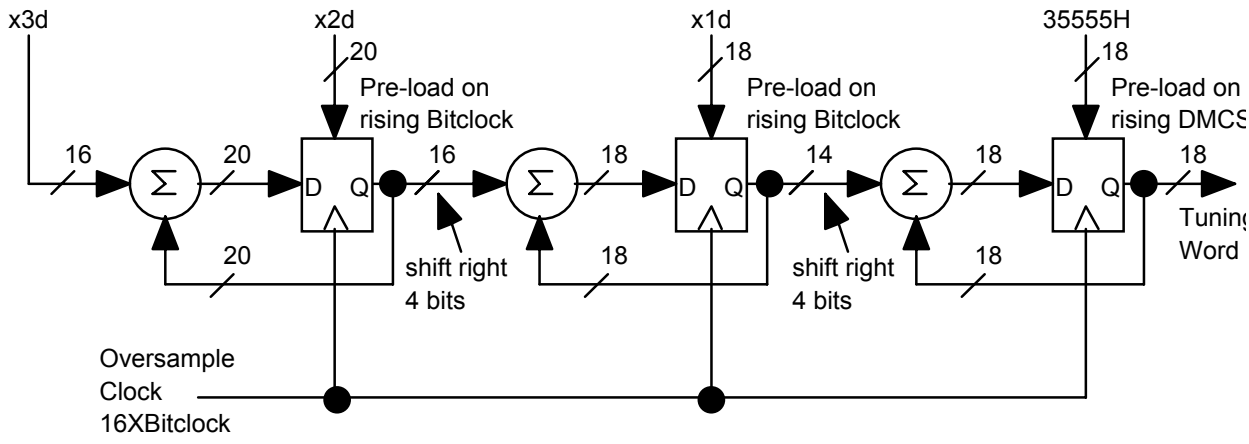


Figure 30-8. Waveform Generator Block Diagram

Finally, if pd_step is programmed to 1 then if TXDATA5-TXDATA0 =000000, then the frequency is forced to 0AAABH (+67kHz in 2 s compliment) for the Bitclock period. Similarly, if pd_step is programmed to 1 then if TXDATA5-TXDATA0=111111, then the frequency is forced to 35555H (-67kHz in 2 s compliment)for the bitclock period. These two cases represent constant frequency situations and thus the best waveform shaping possible is to force the desired constant.

If pd_step is programmed to 0 then the data is not forced to +/-67kHz. It will hold the last value obtained by the waveform generator before the all 0 or all 1 situation occurs.

Note that the data transmission in this mode will be delayed by two bitclock periods as compared to the normal transmission mode. This is to allow time for the addition system to determine the frequency coefficients.

30.4.5 PCS (1800MHz) and DCS(1900 MHz) Mode

TX_VCO_DIV= 1

When transmitting PCS(1800MHz) or DCS (1900MHz) the Tx VCO signal in Algae is divided by 2 before being sent to Neptune. To keep the maximum synthesizer deviation to +/- 67kHz the effective deviation in the loop in Neptune is also divided by 2, giving 33.5kHz. To achieve this we need to modify the digital signal as it goes into the FracN. A divide by 2 is enabled by the signal TX_VCO_div.

We can use all of the above mentioned methods of transmission in this mode.

30.5 Dual Port D/A Chain(Analog)

30.5.1 Introduction

The DP D/A chain forms the analog interface to the transmit RF VCO. The waveform generated through a 9-Bit DAC, addressed by the output of the LUT, corresponds to the frequency deviation of the wanted modulation, versus time. As such it consists of a 9-bit DAC, a 6-bit scaling DAC which scales the 9-bit DAC output, and an output buffer with a single pole filter located at 750kHz. A block diagram of the D/A chain is as follows:

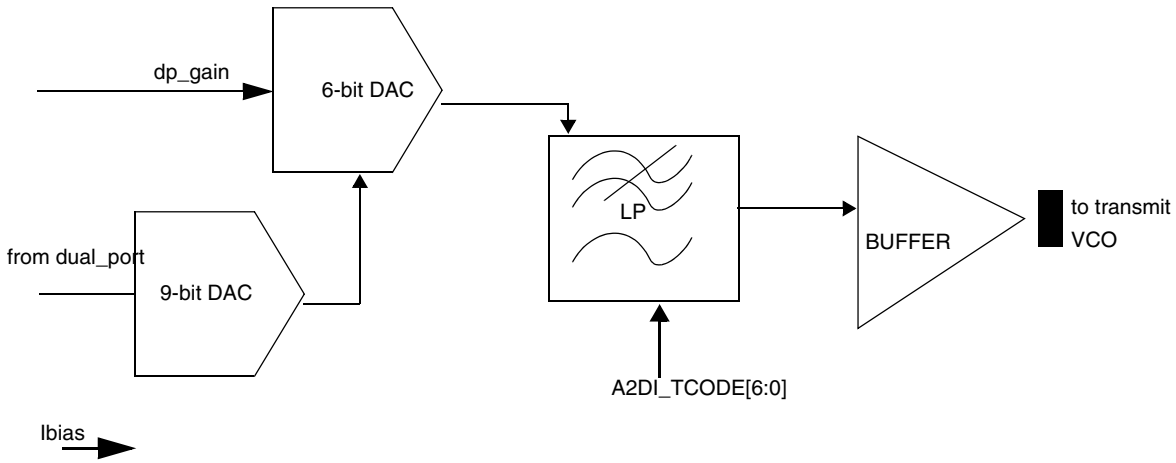


Figure 30-9. Block Diagram of the D/A Chain - Analog Part

30.5.2 Dual_Port Port Description

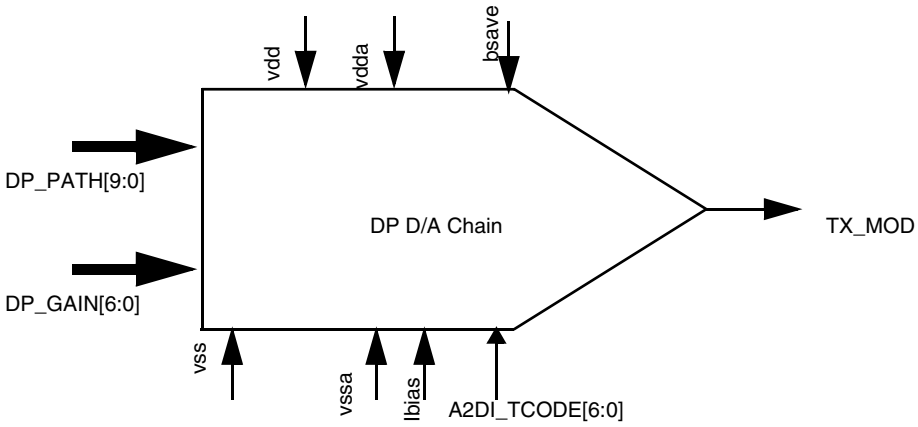


Figure 30-10. Block Pin-out

Table 30-3. Port Description

Port Name	Function	to/from	Comments
DP_PATH[8:0]	digital I/P to DAC	TX Data Dig	Input signal for 9-bit DAC From GMSK LUT in TX_DIG
DP_GAIN[5:0]	digital gain DAC	from digital	Scales the dac9 output
VDD	1.5V dig supply	tbd	digital supply
AVDD1	2.5V dedicated analog supply	vdd_txrx	4mA current (100nA in bsave)

Table 30-3. Port Description (Continued)

Port Name	Function	to/from	Comments
TX_MOD	output	pad	drive up to 25pF Capacitive load
AVSS1	dedicated analog gnd	pad	5mA current (100nA in bsave)
Lvdd	1.575 analog reference	from Seaweed	
Lvss	Ground match of Lvdd	from Seaweed	100uA Current
VAG	1.25 analog reference	from Regulator (Buffered in RXAFE)	no Current No longer used as a refernce in the Dualport
Jvdd	Unbuffered 2.775 Supply	pad	1mA Current (100nA for bsave)
Ibias	Bias current	Regulator block	5uA Bias for internal current references
A2DI_TCODE[6:0]	tuning word	RC tuning	From tunec block
bsave	battery save	TX Dig	active H (Bsave current is 100nA)i

30.5.3 Dual-Port Specifications

Table 30-4. Dual-Port Specifications

Parameter	Condition	Symbol	Guaranteed Limit LVDD = 1.575V ±2% (Application)	Guaranteed Limit LVDD = 1.575 ±6% (Test Bench)	Unit
6-bit DAC Integral Non-linearity	6-bit DAC = 0 to 63; 9-bit DAC = 0, 356, 511; Measure at TX_MOD_OUT	INL	±2	±2	LSB
6-bit DAC Differential Non-linearity	6-bit DAC = 0 to 63; 9-bit DAC = 0, 356, 511; Measure at TX_MOD_OUT	DNL	±1	±1	LSB
6-bit DAC Offset	6-bit DAC = 0; 9-bit DAC = 511; Measure at TX_MOD_OUT	OS	1.0 ±0.1	1.0 ±0.1	V
6-bit DAC Gain Error	6-bit DAC = 0, 63; 9-bit DAC = 0, 511; See note 1		±10	±11	%
9-bit DAC Integral Non-linearity	9-bit DAC = 0 to 511; 6-bit DAC = 31, 63; Measure at TX_MOD_OUT	INL	±2	±2	LSB

Table 30-4. Dual-Port Specifications

Parameter	Condition	Symbol	Guaranteed Limit LVDD = 1.575V ±2% (Application)	Guaranteed Limit LVDD = 1.575 ±6% (Test Bench)	Unit
9-bit DAC Differential Non-linearity	9-bit DAC = 0 to 511; 6-bit DAC = 31, 63; Measure at TX_MOD_OUT	DNL	±1	±1	LSB
9-bit DAC Gain Error	9-bit DAC = 0, 511; 6-bit DAC = 31, 63; See note 2		±10	±11	%
Filter 3 dB Frequency	TUNEC Operating		750 ±150	750 ±150	kHz
Spot Noise Measured at output for the following conditions 1) dac6=32 dac9=256 2) dac6=63 dac9=0 3) dac6=63 dac9=127 4) dac6=63 dac9=371 5) dac6=63 dac9=511	Spot Noise at 100kHz Spot Noise at 400kHz			< 500 <400	nV/rtHz nV/rtHz
Maximum Load	External load from TX_MOD_OUT to GND	CL	25	25	pF

Note 1: Guaranteed limit = 100(Vdelta – Videal)/Videal
where Videal = 0.55 V when 9-bit DAC = 0 or 511, Vdelta is the difference in voltage at TX_MOD_OUT between 6-bit DAC = 63 and 0.

Note 2: Guaranteed limit = 100(Vdelta – Videal)/Videal
where Videal = 1.1 V when 6-bit DAC = 63 and 0.55 V when 6-bit DAC = 31, Vdelta is the difference in voltage at TX_MOD_OUT between 9-bit DAC = 511 and 0.

30.5.4 Dual_Port Functional Testing

Here are the Anatest settings for LT. The filter can be tested by applying an ac signal and a cm signal with the appropriate anatest setting bellow.

Table 30-5.

Bit[5]	Bit[4]	Bit[3]	Bit[2]	Bit[1]	Bit[0]	anatest_p	anatest_n	Description
0	0	0	0	0	0	outn	outp	I to V outputs
0	0	0	0	0	1	follow_out	cmfb	Dac6 and CM outputs
0	0	0	0	1	0	ac_sig+cm in	cm input	filter test (anatest n&p are inputs)
0	0	0	0	1	1	avdd1	avss1	Check supply voltages

Table 30-5. (Continued)

Bit[5]	Bit[4]	Bit[3]	Bit[2]	Bit[1]	Bit[0]	anatest_p	anatest_n	Description
0	0	0	1	0	0	avss1	buffer_cm _level	Checks input common mode voltage to Buffer
0	0	0	1	0	1	vdd_dac	avss1	Dac voltage level
0	0	0	1	1	0	spareA	spareB	Spare anatest pins for fibs

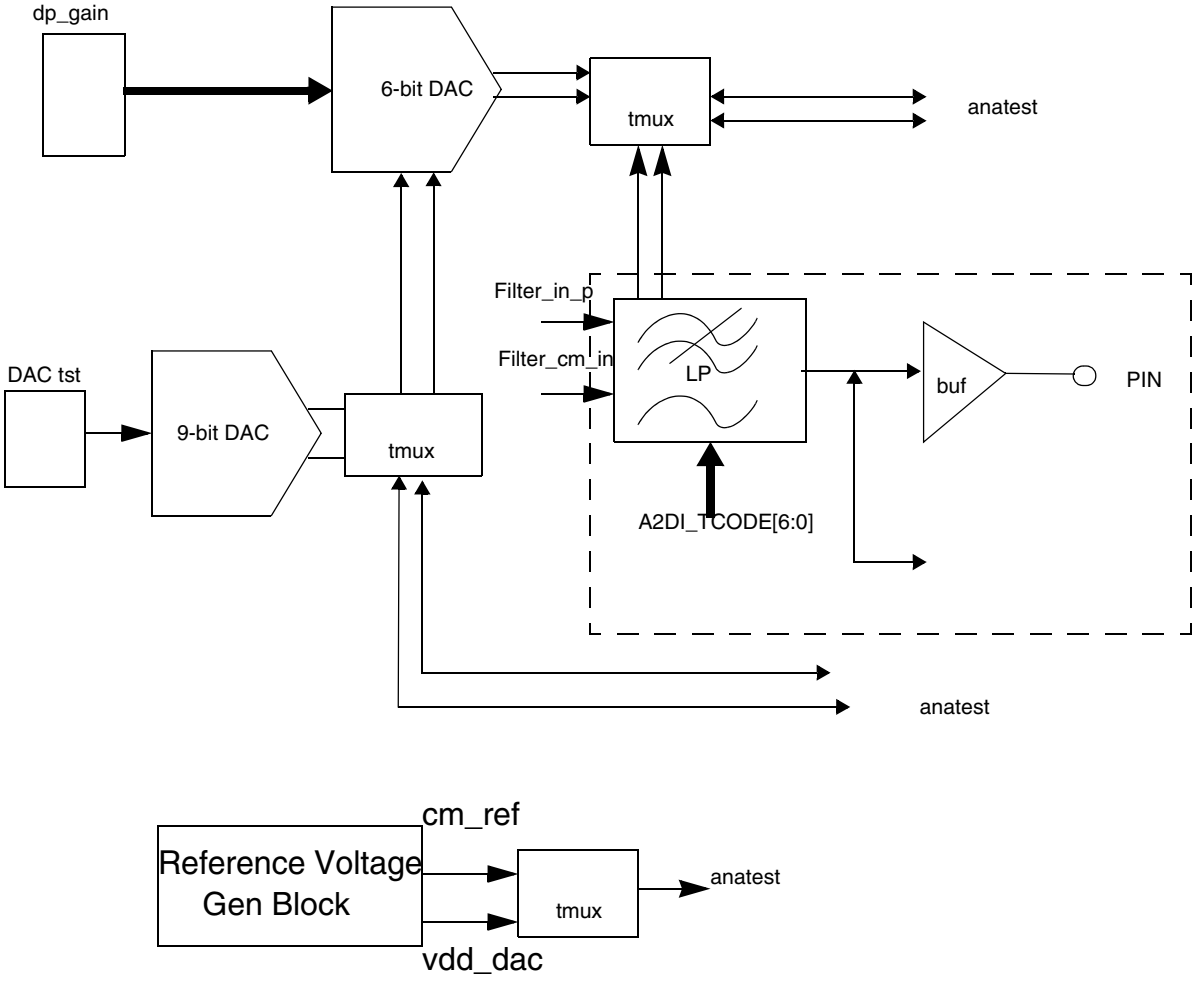


Figure 30-11. Dual_port Analog Testing

30.6 TRSYNT -TX & RX Synthesizer (Analog)

30.6.1 Introduction

This Synthesizer is to interface with the RX and TX Synthesizer Feedback on the ALGAE IC which is AC coupled due to the DC bias on the Prescaler Buffer. The operating frequency range for is between 695MHz and 2GHz. The Modulo 5/6 divider, A and N counters are differential MOS design but the Phase Detector takes the output of the Prescaler (FVMAIN) and XTAL_OSC block and locally inverts the signal to clock the internal Flip-Flops.

The TX Charge Pump Up & Down currents are 1.26mA nominally. An intentional leakage current 246uA(TRICKLE Mode 0)(7ns) sets the Integer Locked condition so that in Fractional Mode the modulation is LINEAR and does not encroach the dead zone region of the PFD/Charge Pump. The Trickle current can be increased by programming the TRKL_SEL<2:0 SPI bits to move the locked condition.

The RX Charge Pump Up & Down currents are 506uA nominally. Its intentional leakage current is 101uA(TRICKLE Mode0)(7ns) to set the Integer Locked Condition. Again the Trickle Current can be increased to move the locked condition. Note that the TRKL_SEL<2:0 SPI bits are shared between TX & RX modes of operation.

On Neptune LT there is a facility at the output of the Charge Pumps to measure the Kv of the VCO for the different Bands for the Dual Port Gain calculation. There is a 200k Ohm resistor to ground at the output of both the RX & TX Charge Pumps. The leakage due to this resistor is offset by the TRICKLE currents.

For Neptune LTE the 200k resistor at the output of the RX Charge Pump has been removed as it causes too much Charge Pump Compliance variation. Also the Kv measurement is specific to TX Mode of operation to set the Dual Port Gain so it is not required for RX Mode. This resistor is now being used to increase the impedance at the output of the TX Charge Pump (to 400K Ohms) BUT preserving the Tuning Line measurement(CP_TX_DIV2 i.e. TX Tuning Line voltage divided by 2). The CP_RX_DIV2 is now being tied off such that it measures CP_TX_DIV4 i.e. TX Tuning Line Voltage divided by 4.

Since the RX TRICKLE currents now have to be readjusted to remove the compensation due to the 200k leakage resistor Table 30-10 below has been updated to reflect this.

Operating Supply Voltage for the Charge Pump is from 4.5V to 5.15V max.

Table 30-6. Synthesizer Specific Pin Table

Name	Description	Type	Freq	Supply Group	Connects To
VSSA_CP	Charge Pump Ground	P	DC	4.75V Group (VM_REG from SEAWEED)	
TX_CP	TX Charge Pump	A	<26MHz	4.75V Group	SEAWEED VM_REG/GCAP
VDDA_CP	4.75V source for charge pump	P	DC	4.75V Group	
RX_CP	RX Charge Pump	A	<26MHz	4.75V Group	
CP_BIAS	(REQUIRES 1% External Resistor)	A	DC	1.575V REFERENCE from XTAL OSC.	

Table 30-6. Synthesizer Specific Pin Table (Continued)

Name	Description	Type	Freq	Supply Group	Connects To
VCO1n VCO1x	Feedback from VCO DIFFERENTIAL	RF	2GHz	1.575V REG_TXRX_O GROUP	REDUCED ESD is expected 500V min. From Algae
OSC26m_syn_clk26p OSC26m_syn_clk26n	26MHz clock DIFFERENTIAL	V	26MHz		From CROSC
VSS_PRE	Dedicated Prescaler GND	V	DC	1.575V REG_TXRX_O GROUP	
VDD_TXRX	2.65V(Pass 1 of Neptune) 2.775 (Pass 2 of Neptune)	V	DC	2.65V Group (RF_REG) from SEAWEED	from regulator
VSS_TXRX	0V	V	DC	2.65V Group (RF_REG) from SEAWEED	from regulator
CP_TX_DIV2	A2D I/P (KVCO)	V	DC	Tuning Line Voltage/2	to GPADC
CP_RX_DIV2	A2D I/P (KVCO)	V	DC	Tuning Line Voltage/2	to GPADC
REG_BYP_PRE	1.575V	V	DC	REG_TXRX_O Group	

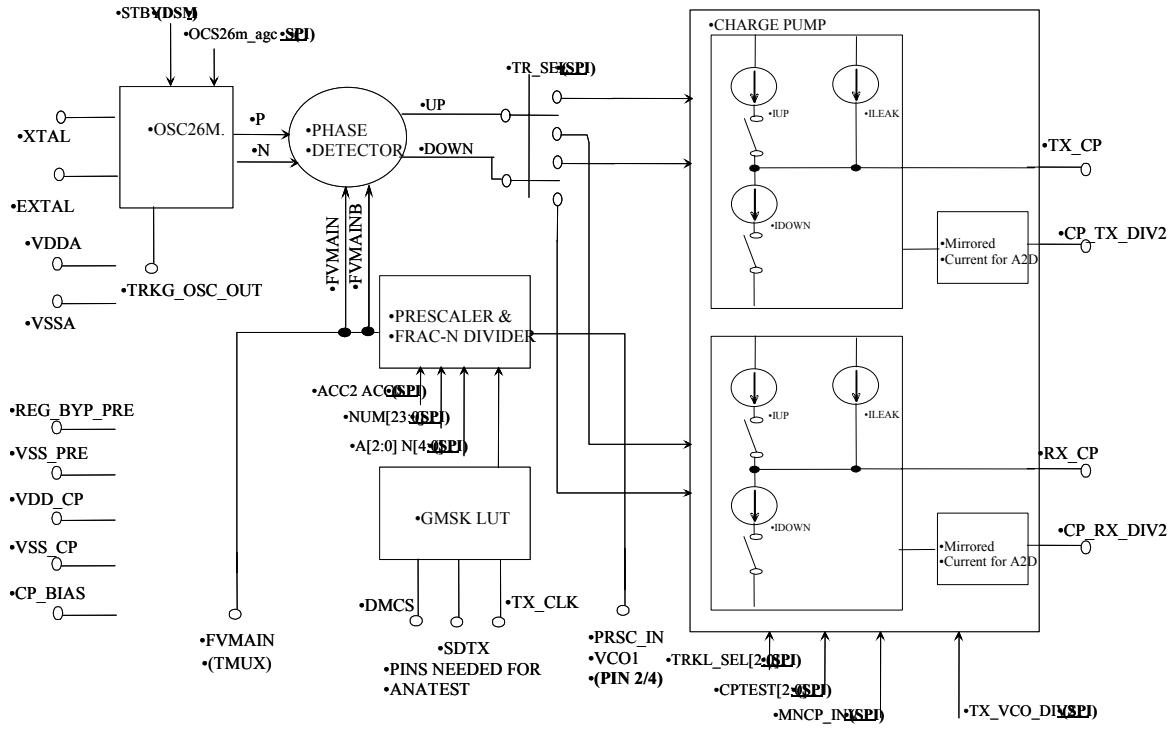


Figure 30-12. TX Synthesizer Block Diagram

30.6.2 Power Supply Specification

- Supply voltage:
  - — 4.75V to +5.15V/-4.5V on 4.75V on VM REG from SEAWEED. Requires 4.7uF External Capacitor.
  - — 2.775V ± 5% on RF_REG from SEAWEED. Requires 4.7uF External Capacitor.
  - — 1.575 +/- 5% on REG_BYP_PRE from REG_TXRX Synthesizer Regulator. Requires 1uF External Capacitor.
- Simulation range -40'C to +110'C.

Table 30-7. Power Consumption

MODE	VDDA_CP Supply Current @25'C	VDD_TXRX Supply Current @25'C
RX Mode	Max. 3mA	Max. 3mA
TX Mode	Max. 5mA	Max. 8mA
BSAVE	<50uA, for VDD_TXRX to ~0uA	Between -10µA to 30µA

NOTE:

There can be a 20% increase in power consumption due to temperature extremities.

30.6.3 Main Divider Specification

- **Technology**: The Fractional part of the Divider is implemented in Synthesized CMOS technology. The remaining part as differential MOS.
- **Architecture**: Dual Modulus Prescaler (5/6) that operates up to 1GHz.

30.6.3.1 GSM Mode:

(TX_VCO_DIV2=0) - Lo Band For **Fractional -N** Mode of operation the following equation applies for both RX & TX Modes and **FN_MODE=0;**

$$Fvco = \left(5N + A + 3 + \left(\frac{Num + 10000(HEX)}{2^{24}}\right)\right) XFref$$

(TX_VCO_DIV2=1) For **Fractional -N** Mode of operation (RX and TX) is as follows;

$$Fvco = \left(5N + A + 3 + \left(\frac{Num + 8000(HEX)}{2^{24}}\right)\right) XFref$$

Programming Example (GSM Mode): (TX_VCO_DIV2=0)

Fvco = 900MHz/26MHz = 34.61538462 = Nint+Nfrac.

Nint = 34 = 5N + A + 3 => N= 6 and A =1.

Nfrac = Num = 0.61538462 * 2^24 - 20000(HEX) = 9C89D8(HEX).

Notes:

TX_VCO_DIV2 = 0 -> GSM Mode. TX_VCO_DIV2=1 -> PCS, DCS Mode.

This is for 2 & 3 Accumulator Modes of Operation.

The Constraint on Num is that Num[23:0] = Required Numerator + AFC.

The Offset 10000(HEX) corresponds to an offset frequency of 101.5625 kHz and is independent of the Numerator i.e The GSM Modulation does not set a constraint on the NUM value. This in fact is the condition of the output of the GSM Look Up Table when DMCS is in the low condition i.e not transmitting or RX Mode.

Fref is XTAL OSC nominal frequency i.e. 26 MHz.

N is 5 bits, A is 3 bits. (SPI PROGRAMMABLE)

Programming Example(PCS/DCS Mode): (TX_VCO_DIV2=1)

Say an fvco=1.8GHz is required, then the total division becomes;

Ntot = 1800/26 = 69.23076923

To allow for the extra division by 2(out of Algae) the Prescaler Divider in Neptune must be programmed as Neptune = Ntot/2 = 34.61538462 = Nint + Nfrac.

Nint = 34 = 5N+A+3 => N=6 and A=1.

Nfrac = 0.61538462 = (NUM + 8000 HEX) => NUM = 9D09D8 (HEX)

30.6.3.2 EDGE MODE:

(TX_VCO_DIV2=0) - Lo Band For **Fractional -N** Mode of operation the following equation applies for both RX & TX Modes and **FN_MODE=1 and RAMP_SEL=2(with DMCS low);**

$$Fvco = \left(5N + A + 3 + \left(\frac{Num + 20000(HEX)}{2^{24}}\right)\right)XFref$$

DMCS HI:

$$Fvco = \left(5N + A + 3 + \left(\frac{Num + 15555(HEX)}{2^{24}}\right)\right)XFref$$

(TX_VCO_DIV2=1) - Hi Band For **Fractional -N** Mode of operation the following equation applies for both RX & TX Modes and **FN_MODE=1 and RAMP_SEL=2(with DMCS low);**

$$Fvco = \left(5N + A + 3 + \left(\frac{Num + 30000(HEX)}{2^{24}}\right)\right)XFref$$

DMCS HI:

$$Fvco = \left(5N + A + 3 + \left(\frac{Num + AAAA(HEX)}{2^{24}}\right)\right)XFref$$

Notes(1):

Do not operate under the following conditions **RAMP_SEL=2, FN_Mode=0 and TX_VCO_DIV2=X.**

Notes(2):

Note that the frequency deviation is corrected at circuit level, while the carrier frequency correction is done by the program in the DSP(AFC). In the TX_VCO_DIV2 mode the division ratio has doubled so the noise contribution due to the reference XTAL will increase by 6dB. In TX_VCO_DIV2 mode if the loop gain is increased by 2 to compensate for the extra division by 2(so that the loop Bandwidth and transient response don't change between modes) then the noise contribution from the FRAC-N modulator and the DUAL PORT DAC will increase by 6 dB.

For **Integer Mode** of operation the following equation applies for RX and TX modes;

$$Fvco = (5N + A)XFref$$

- **Input Impedance**: The prescaler input impedance is characteristic of a typical high-impedance MOSFET input. The 50 Ohm termination is not used for a good voltage match.
- **Divider Outputs:** All logic to work on positive edge of fvmain in the Differential MOS ccts. CMOS Logic will be working on the positive edge of fvmain
- **Power Consumption:** 100mW @ 25'C
- **Power Consumption** in Battery Save: -10uA to 50uA.

- **Bandwidth:** 695MHz to 2GHz. Prescaler Dynamic Range: 200 mVp-p differential (70 mVrms differential) over all the bandwidth. This is limited by the loading of the input PAD, which may not be a 50Ω load.
- **Prescaler Coupling**: The input has to be AC coupled from ALGAE due to DC Bias in the prescaler Buffer.
- **Battery Save**: Prescaler enters battery save when the bsave_tx and bsave_rx are both high.

Table 30-8.

a2di_bsavetx	Tr_sel	En_tx
0	0	0
0	1	1
1	0	0
1	1	0

Table 30-9.

a2di_bsaverx	Tr_sel	En_rx
0	0	1
0	1	0
1	0	0
1	1	0

Table 30-10.

En_rx	En_tx	Prescaler Buffer & Dividers	Ph. Detector	Current Reference	Charge Pump RX	Charge Pump TX
0	0	0	0	0	0	0
0	1	1	1	1	0	1
1	0	1	1	1	1	0
1	1	1	1	1	1	1

Notes: Battery Save Active HIGH.

- 1 – Block running/ functioning
- 0 – Block is in Battery Save Mode.

TR_SEL - 0 – RX Mode

- 1 – TX Mode

For Neptune 2pX the Current Reference is common for both the RX and TX Charge Pumps so that it is always enabled if ‘en_rx’ is high or ‘en_tx’ is high or both are high.

- Therefore only the UP and DOWN signals from the Phase Detector are gated by ‘en_rx’ or ‘en_tx’.

30.6.4 FRAC-N Architecture

24-bit Accumulators are used in the FRAC-N implementation. 26 MHz is the reference frequency that the Prescaler input is being divided down to. 24-bit resolution in the accumulators corresponds to ~1.5Hz resolution for DDM(Direct Digital Modulation). This is from GSM mode. For DCS (1800MHz) and PCS (1900MHz) the TX_VCO_DIV2 spi bit is set high and 23 bits is effectively being used with the MSB sign extended to give 24 bits.

Numerator[23:0] is summed with the modulation data to yield the total numerator value.

The total numerator value, NUM[23:0] = Required Numerator + AFC < 2^24 - 1.

The AFC data is contained in the Numerator Data.

30.6.5 Max. Division Ratios with (P/P+1):

The overall division is formulated below and uses the

following constraint;

From Section 29.6.3 the Fvco (Ntotal) is given by(just taking into account the P/P+1 part)

Ntotal = (33 - np)*5 + (4-ap) where (33-np)=N[4:0] (spi programmed) and(4-ap)=A[2:0] (spi).

i.e. Ntotal = 5N + A from a spi point of view.

'np' and 'ap' are the outputs of the Synthesized Digital Block driving the Analog Digital Blocks. The reason for the change is that the Synthesized Digital Core uses **DOWN** counters but the Analog Digital Blocks uses **UP** counters and also to incorporate a **FAILSAFE** condition that has been added to the Synthesized Digital core (which is being described hereon).

for (33-np) >= (4-ap)

Where np is 5b-bit and ap is 3-bit. While np can take on values from 0

to 31, and ap can take on values from 0 to 7, this divider functions

properly only in the following range;

Usable programmable range of the divider is given as;

0 <= np <= 29

0 <= ap <= 4

Therefore the possible division ratios are;

20 <= Ntotal <= 169 ... Integer Mode (5N + A)

23 <= Ntotal <= 172 ... Fractional N Mode(3 Accumulators) (5N + A + 3)

For Neptune the lowest expected frequency is 695MHz giving N_L = 26.7XXXX.

For the high range a frequency of 2GHz gives an N_L of 76.8XXX.

30.6.6 Phase Detector and Charge Pump Specification

The main loop phase detector is tri-state with an intentional leakage current to linearize the output charge per cycle versus the input phase difference. This is what was used in previous Magic Chips.

The output of the Recombination Logic in the FRAC-N dividers can modulate 'ap' by +4/-3 counts around the INTEGER LOCKED CONDITION. This condition is chosen so to meet the FRAC-N modulation and also without encroaching on the minimum pulse width (~2ns), the Dead zone of the Phase Error V Charge Linearity characteristic of the Phase Detector/Charge Pump.

The Lowest expected frequency determines the region of modulation that has to occur and which must be linear i.e. away from the Dead Zone.

For Neptune the lowest expected frequency is 695MHz(~1.44ns). Modulation - +4(1.44ns), -3(1.44ns) around the Integer Locked condition. The Integer Locked Condition is set to ~7ns. This gives an intentional leakage current of 232uA.

It can be shown that I_{LEAK} x I_{DOWN} = Locked Phase Error Condition(7ns)/38ns.

In Integer Mode the Min. Pulse Width is ~7ns. In Fractional mode the Min. Pulse width can decreases to;

7ns - 3 X (1/695MHz). The extra leakage current required is given by the above equation but in the Charge Pump the TRKL_SEL spi bits moves the Integer Pulse Width from ~7ns for TRKL_SEL=000(232uA) to ~18ns for TRKL_SEL=111(598uA).

Dynamic Range of CP Outputs: The current output of the Charge Pumps has to be maintained 0.5V to VDDA_CP-0.5V.

Minimum Pulse Width Integer mode: ~7nS.

Minimum Pulse Width Fractional Mode: ~7ns - 3 X (1/695MHz).

Phase Detector Frequency: 26 MHz both for the reference and the Divider output.

Feedback delay on Phase Detector reset: <1ns over PVT.

Table 30-11. Charge Pump Current Output:

TR_SEL	CP_TEST	TRKL_SEL	Min	Typ	Max
1	001	000	197uA	246uA	295uA
1	001	001		+50uA	
1	001	010		+50uA*2	
1	001	111		+598uA	
1	011	000	1.21mA	1.52mA	1.82mA
1	101	000	1.00mA	1.26mA	1.512mA
0	001	000	800uA	101uA	120uA
0	001	001		+17uA	
0	001	010		+17uA*2	
0	001	111		+219uA	
0	011	000	488uA	611uA	733uA
0	101	000	403uA	504uA	605uA

Notes:

Limits - +/-20% over PVT over Typical Currents listed above.

- TR_SEl = 1 - Transmit. TR_SEL=0 - Receive.
- CPTEST Mode 000 - Normal Mode - *.
- CPTEST Mode 001 - Itrickle Mode - *.
- CPTEST Mode 011 - Iup + Itrickle - *
- CPTEST Mode 101 - Idown + * (Note that TRKL_SEL spi bits can NOT be changed in this mode.)

* - Leakage Current to the Resistive divider on the Tuning Line Voltage of the TX Charge Pump for calculating TX_CP_DIV2. Leakage = Vtune/400k. Note that this is 0 for RX Mode of operation.

Table 30-12. Neptune Synthesizer Noise Requirement[1]

Name	Δf	Specification
Charge Output Noise Current	Δf=100 kHz Δf=400 kHz	132pA/rtHz. 31pA/rtHz.
VCO Gain(KV)[2]		200-325 MHz/V.
Phase Noise at 1GHz, TX mode	Δf=100kHz[3] Δf=400kHz	-105dBc/Hz. -116dBc/Hz.[4]
Phase Noise at 750MHz, RX mode	Δf=15–25kHz (inside loop BW) Δf=3MHz	-92dBc/Hz -136dBc/Hz
Lock Time, recovery from standby	Stby to 772MHz +/- 200Hz	180us
XTAL OSC		Differential to the Phase Detector. 'CLK_OUT' output from the XTAL_OSC cct. Output to 'TRKG_OSC_OUT' pin is 26MHz
General Purpose ADC		CP_RX_DIV2 & CP_TX_DIV2 (0 - 2V)

1. Typical loop bandwidth: TX=150kHz, RX=75kHz.
2. For frequency output of VCO which is 4 times the TX frequency for GSM and 2 times the TX frequency for DCS1800 and PCS1900.
3. The 100KHz offset frequency is intended to measure in-band noise of the synthesizer. As such, VCO component of the noise should be avoided by using a low noise VCO, or a wideband PLL for this measurement. For a loop BW of approximately 125kHz, a commensurate delta f inside loop BW of approximately 25kHz should be used.
4. -110.8 dBc/Hz max at 400 kHz, extra 6dB margin required for the Merchant Market.

4.)For PCS TX mode the Charge Pump TX Noise Current is from 98 to 31pA/rtHz at 400Khz.

For GSM TX mode, the Charge Pump TX Noise Current is from 155 to 49pA/rtHz at 400Khz.

For PCS RX mode, the Charge Pump RX Noise Current is from 397 to 126pA/rtHz at 400Khz.

For GSM RX mode, the Charge Pump RX Noise Current is from 824 to 262pA/rtHz at 400Khz.

These limits produce in-band noise levels as follows:

PCS TX mode -105dBC/Hz
GSM TX mode -107dBC/Hz
PCS and GSM RX mode -92dBC/Hz

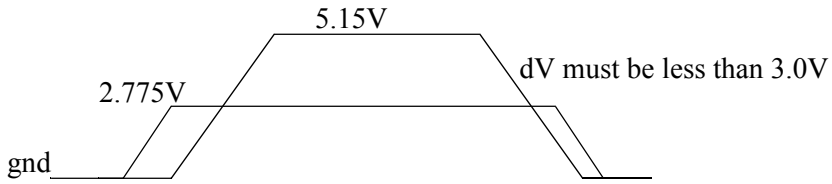


Figure 30-13. Power Sequencing

- XTAL OSC: Differential as well.(See Chapter 29, “Receiver Saturation Detection and SDM Dither Generation (RxSDG).”)

Table 30-13. MAIN PLL SPI Bit Definition:

Name	# of Bits	Description
TRKL_SEL[2:0]	3	intentional leakage Offset currents
ACC2	1	High => 2 Accumulators.are used instead of 3
ACC0	1	Low => accumulators are disabled- No accumulators.
TR_SEL	1	Low = Activates CP_RX. High = Activates CP_TX. When the TX_CP is not selected the output impedance is 400K Ohms to ground due to the CP_TX_DIV2 operation. For the RX_CP this is much higher (ideally infinite) since there is no CP_RX_DIV2 action required.
TX_VCO_DIV2	1	High=> TX_Mode divide by 2 mode Low => TX_Mode no extra division
CPTEST	3	CPTEST2 CPTEST1 CPTEST0 0 0 0 - Normal Operation. 001 - Itrickle Mode. 0 1 1 - All Charge Pump o/ps to source Current(Iup + Itrickle). 1 0 1 - All Charge Pump o/ps to sink Current (Idown). 1 1 1 - Not Used (Idown-Itrickle).
MNCP_INV	1	High => Q-pump sinks current to increase freq Low => Q-pump sources current to increase freq.
NUM	24	Numerator Programming for the 24 bit FRAC-N divider
N	5	Pulse Swallow Mod 5/6
A	3	5N+A+3
BSAVE	1	Battery Save
EXT_OSC	1	High=> Ext XTAL_OSC Low=>XTAL_BASE input
OSC26m_agc	6	Default setting is all 1’s
OSC26m_peak	2	AGC loop control

30.6.7 ANATEST

SPI Bits will be allocated to have the TX_DIG_TX_CLK, DMCS, FVMAIN available for Prescaler Sensitivity testing. SDTX available for checking the Digital Modulation and for the Dual Port System Testing i.e. Global Phase error.

Table 30-14.

Bit[5]	Bit[4]	Bit[3]	Bit[2]	Bit[1]	Bit[0]	Description
1	0	0	0	0	1	fvmain on Test1(ATST_P)
1	0	0	0	1	0	fvmain/2 on Test1(ATST_P)
1	0	0	0	1	1	fvmain/4 on Test1(ATST_P)
1	0	0	1	0	0	nb1_rx on Test2(ATST_N)
1	0	0	1	0	1	Not Used
1	0	0	1	1	0	nb2_rx on Test2(ATST_N)
1	0	0	1	1	1	Not Used
1	0	1	0	0	0	pb1_rx on Test2(ATST_N)
1	0	1	0	0	1	Not Used
1	0	1	0	1	0	pb2_rx on Test2(ATST_N)
1	0	1	0	1	1	Not Used
1	0	1	1	0	0	cb2p4_rx on Test2(ATST_N)
1	0	1	1	0	1	Not Used

(*) fvmain - Is the divided down version of the Prescalar input, typically around 26MHz. This is the input to the Phase Detector.

(*) fvmain/2 and fvmain/4 are divided down versions of fvmain by two and four just in case there is two much parasitics on the way to the pin and to be 50/50 duty cycle.

The others are biases from the Charge Pump circuitry for evaluation purposes and debug;

nb1_tx - ~662mV.

nb2_tx - ~1.466V

pb1_tx - ~2.008V

pb2_tx - ~1.442V

cb2p4 - ~2.35V(VDDA_CP=4.75V)

30.7 CROSC- 26 MHz Reference Oscillator (analog)

Revision History

Revision	Date	Author	Changes
0.0	10/Nov/00	Samuel E. Tang	Preliminary Release.
0.1	14/Nov/00	Samuel E. Tang	Changed synthesizer clock from differential to single ended.
0.2	27Dec2000	Jeremy Moore	Rework of spec. Added details on architecture, test modes.
0.3	2Jan2001	Jeremy Moore	Added details on startup, testing. minor updates, supply names changed for clarity.
0.4	11Jan2001	Jeremy Moore	Revisions / clarifications based on feedback.
0.5	13Jan2001	Jeremy Moore	Yet another rev....
0.6	03 Apr2001	Jeremy Moore	Updates to reflect design changes after design review

30.7.1 Introduction

The reference oscillator provides a stable frequency reference for use by the radio. The oscillator will use a crystal at 26 MHz. Compensation for all types of frequency error in the crystal (temperature, make tolerance, and aging) will be accomplished by offsetting the fractional N division ratio in the synthesizer or the reference PLL. Three separate PLLs will be used in the Triton chip set. The first one will be used to generate the main receiver/transmitter LO signal. The second one will be used to generate the 13 MHz corrected clock which is required by the digital sections of the ratio. The third one is used for USB clock. All three of the PLLs will be the fractional N type. The reference oscillator should be within 150 ppm of 26 MHz within 5 ms of power on.

All the clock outputs are buffered by the OSC26M clock. The 26 MHz reference oscillator block provides a pair of dedicated differential 26MHz clock signal to the receiver block, a pair of dedicated differential 26MHz clock signal to the synthesizer block, and a copy of single ended 26MHz clock signal to the reference PLL, USB PLL, the clock monitoring circuit and the gated 26MHz clock block.

30.7.2 Block Description

The crystal oscillator will be a Pierce oscillator design. The oscillation signal swing will be controlled by an “open loop” digital AGC; microprocessor interaction will be required to tune the oscillator (i.e. “close” the loop). The block diagram below illustrates basic functionality

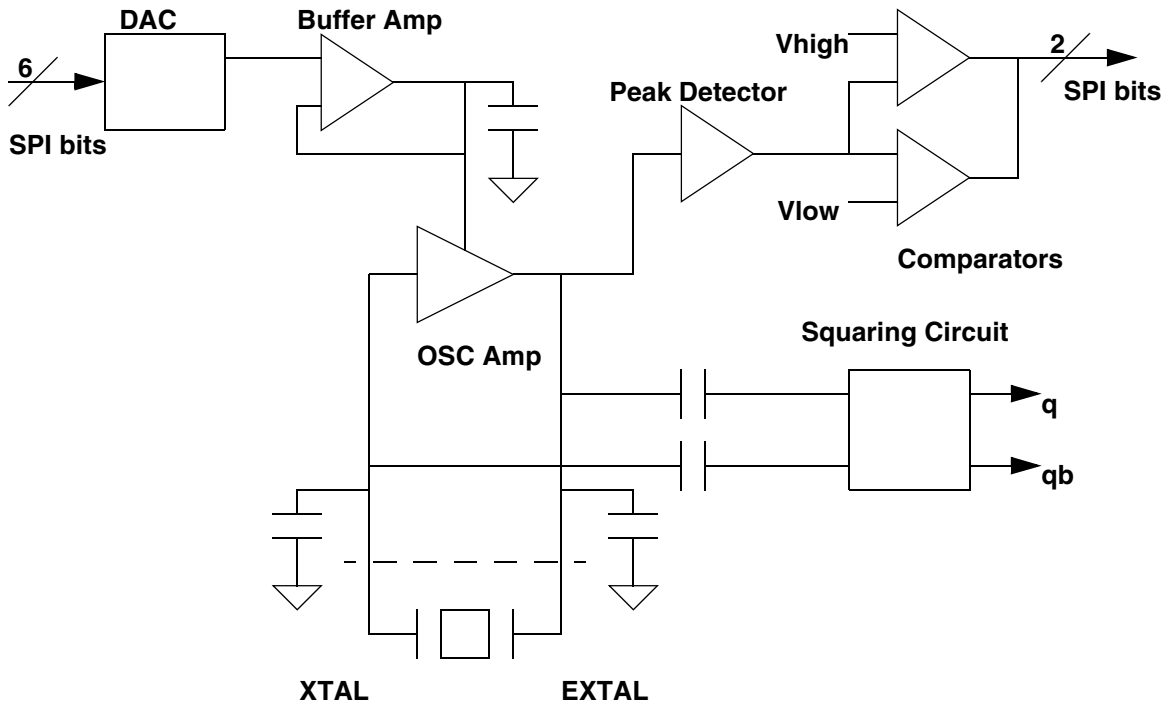


Figure 30-14. OSC26M Block Diagram

The power level for the OSC amp is controlled by a simple 6 bit DAC. The 6 bits are under microprocessor control through 6 SPI bits. A buffer amp is used to support the current load of the OSC amp. The DAC runs directly off of the vddxtal supply. Alternately, with a metal 5 options, the DAC will run off a buffered version of the 1.575 RTC supply, isolating the DAC from the supply variation of the 2.65V supply used by the osc amp. The bandwidth of the buffer amp should be a minimum of 2 decades down from the 26 MHz operation frequency of the osc amp. The output of the OSC amp is fed into a peak detector; this in turn is fed into two comparators. This allows the microprocessor to determine the peak to peak signal level of the oscillator (2 SPI bits). If not in the desired range, the microprocessor can then tune the oscillation level using the 6 bit DAC.

A squaring circuit is AC coupled to the input and output of the OSC amp. This differential signal is gained / squared up to CMOS levels and level shifted to a 1.575 V supply. Differential and single ended outputs are available, and will be buffered and delivered to other blocks.

Capacitors are applied to fine tune the oscillation frequency; this is a value typically spec'd by the crystal manufacturer. For better matching, this capacitance is on-chip; no external capacitors should be added.

An external clock may be fed into the oscillator through the xtal pin, extal must be left floating when driving in an external clock. This will not have the desired spectral purity, only a crystal will provide the necessary signal quality. The osc amp can only be disabled with a metal option to permit usage of an external oscillator in the case where an external oscillator is chosen for normal operation.

30.7.3 Start-Up

Startup of the oscillator introduces some system level issues that must be addressed. The 6 bit DAC control signals must be reset on powerup or after the OSC amp has been powered down. Otherwise the oscillator may not start. The 6 DAC SPI bits should be reset to a logic 1; this will apply the maximum supply to the OSC amp, resulting in the highest possible gain. After the oscillator has started up such that a valid clock is being generated, the oscillator peak level must be trimmed. This requires intervention from the

microprocessor. The microprocessor will have to incrementally trim down the supply level for the oscillator using the 6 bits for the DAC. The 2 outputs from the comparators are provided to indicate the current oscillation peak level to the microprocessor. MSB, when asserted, indicates the oscillation amplitude is too high. LSB, when asserted, indicates the oscillation amplitude is too low. If both bits are asserted; then an error has occurred. These bits will be updated into memory space by the a2dgl block at a frequency of 32kHz. The microprocessor should not try to update the DAC at a rate faster than this.

Table 30-15. osc26m_peak[1:0] Decoded Values

osc26m_peak[1:0]	Action
10	Amplitude too high; trim lower.
00	Amplitude in desired operating range.
01	Amplitude too low; trim higher.
11	Invalid state.

Internal to the oscillator are several circuits that have a time constant associated with them; the microprocessor cannot trim the amplitude of the oscillator faster than these time constants or the oscillator may stop running. The output SPI bits will be latched by the a2dgl at a frequency of 32 kHz. This will dictate the maximum update rate from the microprocessor. Once the trim level is set; it can be stored in memory and reload after a powerdown. Additional start-up circuitry resides in the synthesizer (running off the core 1.575 supply) to aide in startup of the oscillator. At powerup, a counter in the synthesizer will count 256 valid clocks from the output of the OSC amp squaring circuit before any trim values will be applied to the 6 bit DAC. During the 256 count, the DAC will be forced to its maximum output (equivalent to an all 1 input). After 256 valid clocks, the value of the SPI bits will be applied to the DAC input, and the counter is disabled.

30.7.3.1 Summary of Programming Algorithm

Trimming of the oscillator is necessary only on initial powerup; the trim may be stored in FLASH for future reference.

1. At powerup or system reset, osc26m_agc[5:0] bits are reset to logic 1 (done in hardware, no software interaction required).
2. Read the peak amplitude value in A2DIGL; bits osc26m_peak[1:0].
3. If amplitude is not in desired range, adjust by writing osc26m_agc[5:0] Decrement by 1 count.
4. Wait at least 30.5us (1 cycle of 32kHz clock) for system up update osc26m_peak bits.
5. Repeat steps 2 to 4 until trimmed in desired range.
6. Decrement 4 additional counts to provide a margin of error for temperature drift.
7. Store trim value in FLASH for future use.

30.7.4 Power down Modes

Two power down mode senerios are supported. When the signal **stby** is asserted, the OSC26M module will power down. Alternately the 2.65V power supply can be removed.The start up sequence as noted in 30.7.3 must be followed when **stby** is de-asserted. The trim level for the 6 bit DAC can be loaded from memory after a valid clock is achieved if it was saved before entering sleep mode. The microprocessor is responsible for this; the SPI bit values are not stored locally. If the oscillator is left running while other circuits are in deep sleep, the SPI bits should NOT be reset. They should only be reset when the oscillator is powered down.

30.7.5 Test Modes

For all test modes, a clock must be applied to the system (either with a crystal attached and using the internal oscillator or using the external oscillator option.) to allow the loading of the test settings into the anatest module. The majority of testing can be done in the system using the crystal oscillator as the clock source for the system

Key internal nodes are available to measure through the anatest module. Characterization of the DACA, the 1.575V internally generated supply, output of the peak detectors and reference levels to the peak detector are all measurable.

Table 30-16. syn_osc26m Test Modes

anatest_modes<5:0>	Description
6'b xx 0000	No test selected.
6'b 11 0001	xtal supply
6'b 11 0010	squaring supply
6'b 11 1000	peak hi reference
6'b 11 0100	peak low reference
6'b 11 1100	peak detector output

30.7.6 Specifications

Table 30-17. Reference Oscillator Specifications

Specification	Min	Typ	Max	Unit
Supply Voltage	2.68	2.775	2.875	V
Center Frequency	-10 ppm	26	+10 ppm	MHz
[1]Additional capacitance of 19pF is located on-chip.				

Table 30-17. Reference Oscillator Specifications (Continued)

Specification	Min	Typ	Max	Unit
Startup Time			5	ms
Capacitive Load (off-chip)[1]		0		pF
Phase Noise Δ F=10kHz Phase Noise Δ F=100kHz			-140 -150	dBc/Hz
[1]Additional capacitance of 19pF is located on-chip.				

Table 30-18. Spurious Emission Requirement for the 26 MHz Oscillator

Offset Frequency (KHz)	Output RF Spectrum Requirement (dBref/Hz)	Output Sideband Noise Requirement (dBC/Hz)	Loop Filter Attenuation (dB)	Ref Oscillator Sideband Noise Requirement (dBC/Hz)
200 to 400 400 to 1800 1800 to 3000 3000 to 10000	-30 -60 -65 -73	-93.2 -123.2 -128.2 -136.2	-3.0 to 12.0 12.0 to 40.0 40.0 to 61.0 61.0 to 83.0	-133.9 to -118.9 -148.9 to -120.9 -125.9 to -104.9 -112.9 to -90.9

Table 30-19. Typical Crystal Specifications

Specification	Min	Typ	Max	Unit
Motional Capacitance[1]		3.5		fF
Motional Inductance[1]		10.7		mH
Parallel Capacitance (including PC board)			4	pF
Series Resistance Operating: Startup:	 5 10	 15 40	 40 100	Ω
Startup Time			5	ms
[1]Values correspond to a Q of 50,000. (26MHz +- 10ppm)				

30.7.7 Input and Output Description

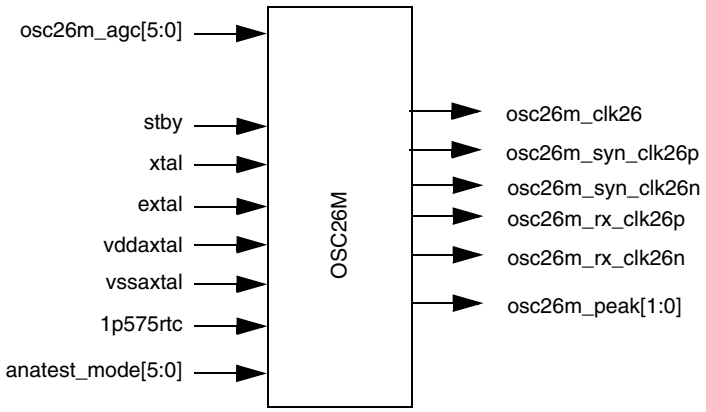


Figure 30-15. OSC26M Pin Diagram

Table 30-20. Inputs and Outputs Description

Name	Direction	Reset State	Description
stby	input	low	Powers down the oscillator when asserted. Reset state is low.
osc26m_agc[5:0]	input	high	6 bit input for AGC of osc amplitude
anatest_mode[5:0]	input	low	Test mode control.
osc26m_peak[1:0]	output	NA	2 bits to indicate current osc amplitude level
osc26m_clk26	output	NA	Output of the 26 MHz clock to clock monitor, USB PLL, reference PLL, and the gated 26MHz clock block.
osc26m_syn_clk26p	output	NA	Positive 26 MHz clock output to synthesizer.
osc26m_syn_clk26n	output	NA	Negative 26 MHz clock output to synthesizer.
osc26m_rx_clk26p	output	NA	Positive 26 MHz clock differential output to Neptune receiver.
osc26m_rx_clk26n	output	NA	Negative 26 MHz clock differential output to Neptune receiver.
xtal	analog	NA	Connecting to one side of the crystal oscillator outside of the chip.
extal	analog	NA	Connecting to the other side of the crystal oscillator outside of the chip.

Table 30-20. Inputs and Outputs Description (Continued)

Name	Direction	Reset State	Description
vddaxtal	power	NA	Power supply to the OSC26M block.
vssaxtal	power	NA	Ground to the OSC26M block.
1p575rtc	power	NA	1.575 RTC supply

Chapter 31
Power Amplifier Control (PAC)

Revision History

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	05/22/02	David Redmond, Shannon Osgood	Updated per DDTS DSPhl14017, replaced all occurrences of highv_offset or a2di_highv_offset with a2di_soft_sat.
0.2	07/12/02	Hugh O Brien, Dermot O'Keefe	Updated per DDTS DSPHl14415 DDTS DSPHl13266, TLSbo24331
0.3	10/16/02	Shannon Osgood	Updated per DDTS# DSPhl15332 and DSPhl15550. Change the PAC analog activity detector trip threshold to +5.0mV/- 3.5mV. Applies to LTE and ULS as well.
0.4	11/12/02	Alan Casey, Shannon Osgood	Updated per DDTS# DSPhl15473/DSPhl15744.
0.5	05/07/03		Updated for LTE specification release.

31.1 Overview

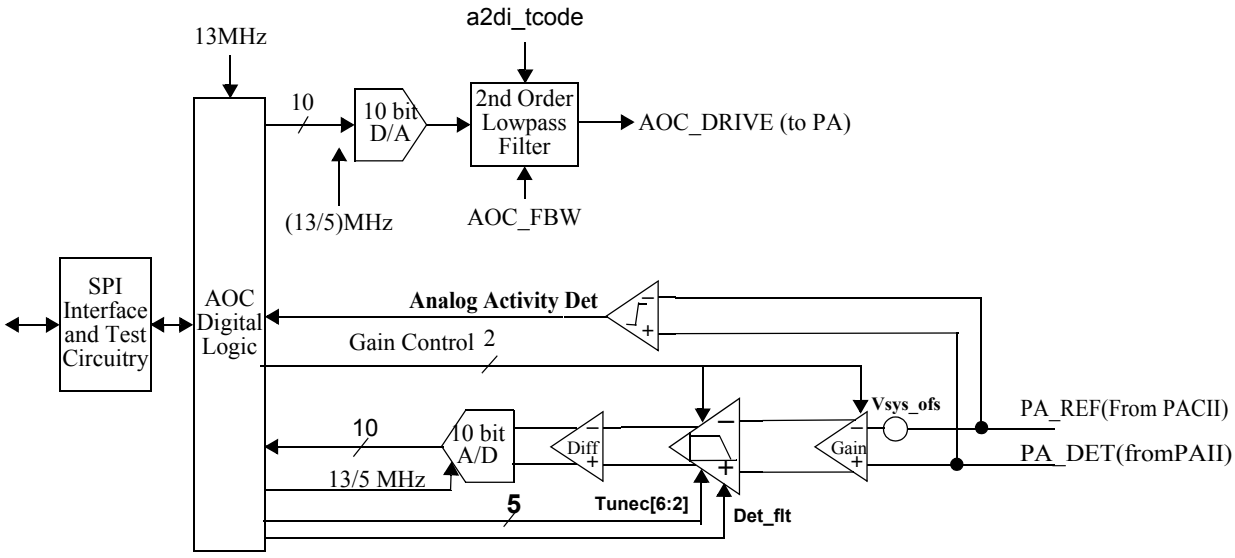


Figure 31-1. Power Control Top-Level View

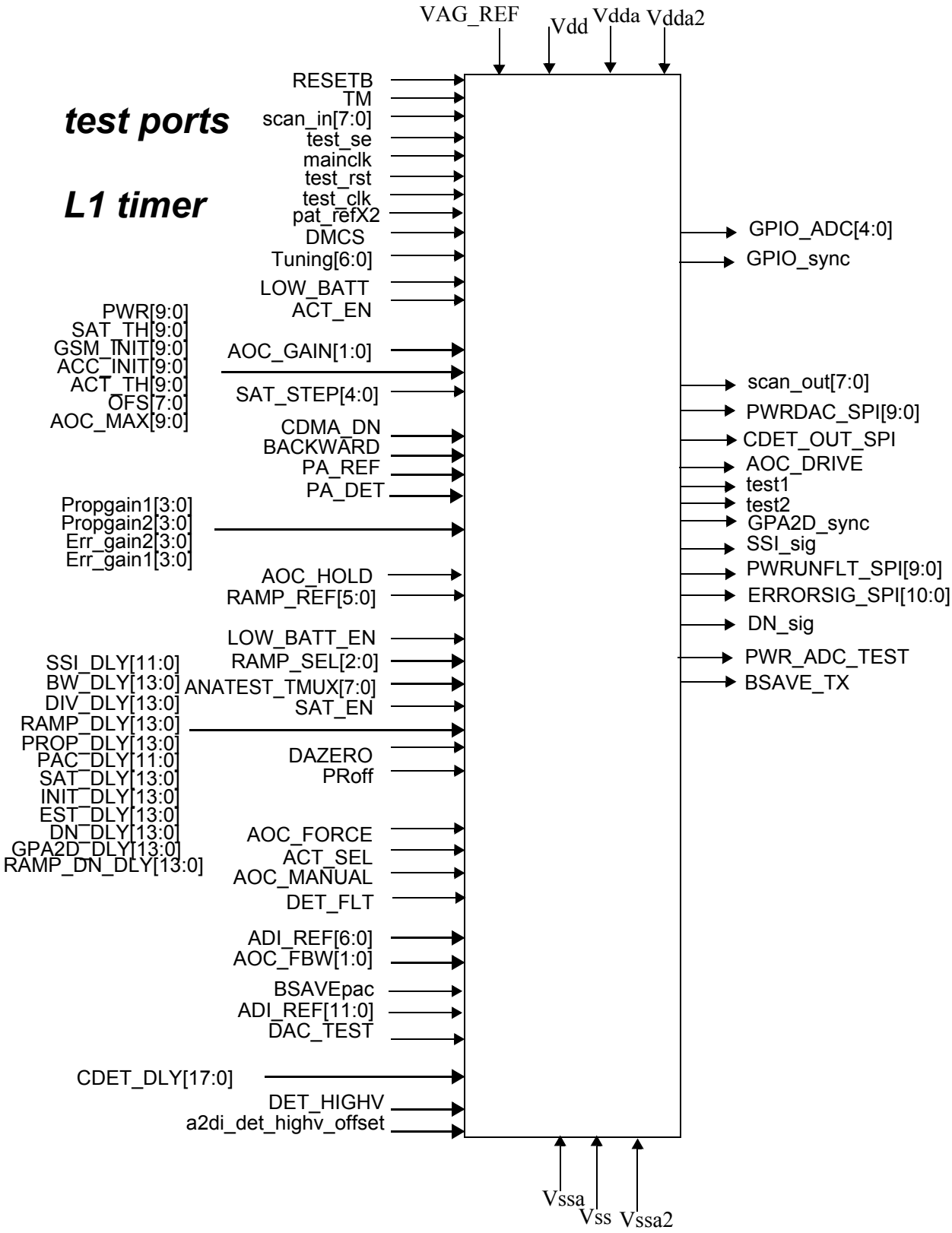


Figure 31-2. Black Box Pin Diagram

The basic operation of the digital AOC control system is described as follows:

1. Three gain settings are provided at the input of the A/D converter. A input offset is introduced to compensate for the PACII negative differential voltage at zero RF power detection.
2. At the beginning of the transmit burst, the signal DMCS will rise. Rising DMCS will clear the accumulator which drives the D/A and insures that the starting control signal is zero.
3. The rise of DMCS is also delayed by a time represented by the digital word PAC_DLY. This delay time is sufficient to insure activation of the PAC II detector DC biasing. (1 µsec max)
4. An A/D digital word corresponding to the DC offset of the PACII detector is stored at the end of this delay time.
5. This offset is subtracted from all detector words throughout the burst to result in only the detected signal level. If the detector subtraction results in less than zero, then zero is forced at this point (a fail-safe for the difference of small numbers).
6. The detected word is subtracted from the desired level of ramp. This results in the error signal between the desired ramp step and the PA output voltage.
7. The signed result of this subtraction is then multiplied by a programmable gain of less then or equal to unity. This gain is realized as a simple shifting operation.
8. The multiplied signal is then applied to the input of an accumulator which is clocked with a programmable clock rate.
9. The 10 MSBs of the output of the accumulator feeds a 10 bit D/A converter.
10. The converters' output is passed through a low pass filter to remove quantization noise and is then applied to the PA or VCA control port(s).

The above steps will form a closed loop controller. The bandwidth of the controller can be varied by changing the multiplication factor applied to the error signal. As the gain can be controlled by shifting as many as 12 steps, the bandwidth of this system can be varied by 4096 to 1. The system can also be placed in an ideal sample and hold mode. The ratio of the bandwidth settings will be very precise due to the digital nature of the system.

As the ramp clock can be much slower than the accumulator clock, it is possible to cause the D/A converter to make intermediate steps between ramp clock pulses. This will aid in smoothing the ramp shape before the output filter. The shaping during an individual ramp step will be influenced by the setting of the less than unity gain operation. Again closer to one will cause the system to attempt to track the ramps steps. A lower gain will cause a smoother response between steps.

In addition a "proportional" gain term is added. This term bypasses the accumulator to feed the error correction directly to the D/A. The gain of the proportional term can be selected from 1 to 1/4096 with control bits PROPGain1 and PROPGain2. The control bit PROP_DLY selects the proportional gain term from PROPGain1 to PROPGain2 and control bit Proff is included to disable this feature.

The following PAC Control variables will maintain their state when programmed from the SPI controller until the next rising edge of DMCS. This allows the next burst parameters to be loaded without interfering with the present burst. Note that some parameters can be loaded instantly from SPI.

ADI_REF PAC_DLY SAT_DLY EST_DLY INIT_DLY

RAMP_DLY BW_DLY DIV_DLY PROP_DLY DN_DLY

AOC_GAIN ADI_REF ACT_SEL ACT_CL GPA2D_DLY

AVG_DIV ACT_TH ACT_EN SAT_TH RAMP_DN_DLY

SAT_STEP SAT_EN PWR RAMP_SEL OFS RAMP_REF

ERR_Gain1 ERR_Gain2 PROPGain1 PROPGain2

Proff ADI_REF GSM_INIT ACC_INIT AOC_MAX
AOC_FBW DAZERO SSI_DLY
DET_HIGHV a2di_soft_seta2di_soft_set

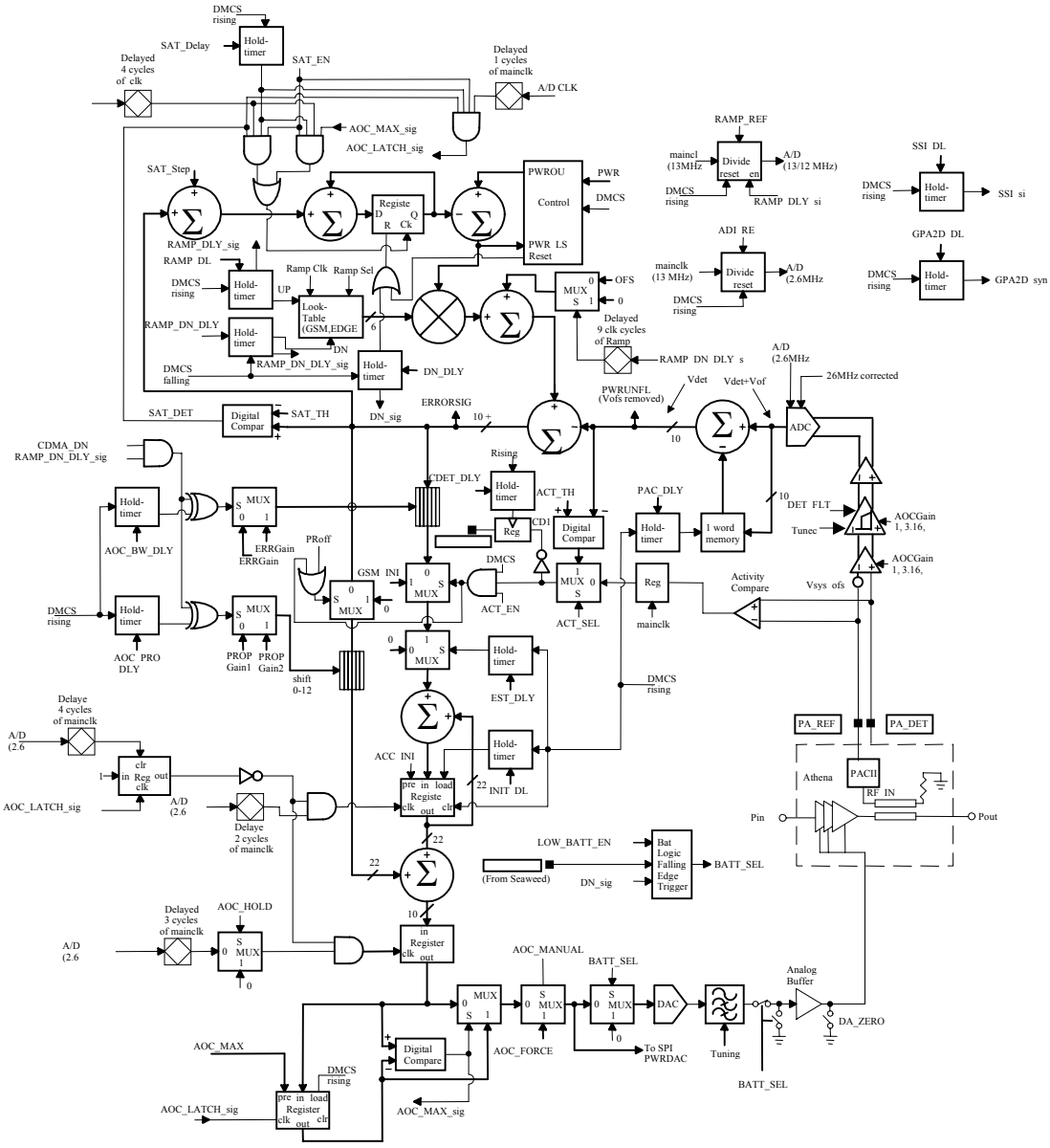


Figure 31-3. PAC System Diagram

31.2 PAC Module Pin List

Table 31-1. PAC Module Pin List

Port Name	Function	to/from	Comments
INPUTS			

Table 31-1. PAC Module Pin List (Continued)

Port Name	Function	to/from	Comments
test_rst	Test reset	TCM	Active High
test_clk	Test Scan clock	TCM	
TM	Test mode	PAD	Active High
scan_in	Scan chain input	TCM	active during TM 8-bit
test_se	Test_Scan Enable	TCM	Active during TM
mainclk	13MHz input main clock. Corrected Digital Clock.	Input from TX module	13 MHz
pat_refX2	26MHz input clock.	Input from TX module	26 MHz
DMCS	Signals beginning of transmit burst	From DSP	Active high, occurs on negative edge of mainclk.
RESETB	Reset signal	Dedicated i/p pin	Put circuit in known state Active LOW
SSI_DLY	Sets the number of clocks from the rising edge of DMCS to the point at which the first TX_CLK occurs. SSI interface variable.	A2DIGL	14-bit
BW_DLY	Sets the number of clocks from the rising edge of DMCS to the point at which the loop gain is switched from ERR_GAIN1 to ERR_GAIN2.	A2DIGL	14-bit
DIV_DLY	Sets the number of clocks from the rising edge of DMCS to the point at which the AOC controller may be programmed to enter a hold mode	A2DIGL	14-bit
PROP_DLY	Sets the number of clocks from the rising edge of DMCS to the point at which the proportional gain is changed from PROP_GAIN1 to PROP_GAIN2	A2DIGL	14-bit
RAMP_DLY	Sets the number of clocks from the rising edge of DMCS to the start of the ramp up in lookup mode.	A2DIGL	14-bit
RAMP_DN_DLY	Sets the number of clocks from the falling edge of DMCS to the start of ramp down	A2DIGL	14-bit
RAMP_SEL	Selects the look up ramp shape	A2DIGL	000 Half Raised Cosine 001 Quarter Raised Cosine Others Not Used Reset Value 000H

Table 31-1. PAC Module Pin List (Continued)

Port Name	Function	to/from	Comments
ADI_REF	Sets the division ratio from the main clock to the accumulator clock	A2DIGL	12 bit
GPA2D_DLY	Sets the number of clocks from the rising edge of DMCS to the start of the General Purpose A2D.	A2DIGL	14-bit
PAC_DLY	Sets the number of clocks from the rising edge of DMCS to triggering the reading the PACII DC offset. Do not program to multiples of 5.	A2DIGL	12-bit
BSAVEpac	If programmed high the PAC system is placed in battery save.	A2DIGL	1-bit
PWR	Power level desired for systems which use the internal look up table for PA ramp shaping	A2DIGL	10-bit
PA_REF	PACII Reference to PAC	From PACII	Analog Input, 60mVtyp
PA_DET	PACII Feedback to PAC. It can handle up to 2.5V input.	From PACII	Analog Input 0 - 2.5V
ACT_TH	Threshold of detected power to insure closed loop control	A2DIGL	10-bit
ACT_EN	Logic high activates activity detection and initial linear ramping. Logic low deactivates this feature	A2DIGL	For modes other than GSM FN the ACT_EN bit is set low
SAT_TH	Threshold of error between detected power and ramp shape waveform which activates saturation detection	A2DIGL	10-bit
SAT_DLY	Number of clocks from rising edge of DMCS to enabling of saturation detection	A2DIGL	14-bit
SAT_EN	Logic high activates saturation detection. Logic low disables this feature.	A2DIGL	When saturation is enabled, PWR is decremented by SAT_STEP on each occurrence of RAMP_clk
SAT_STEP	Saturation decremented step. Negative slope when saturated.	A2DIGL	5-bit Reset Value 00H
A2DIGL_SOFT_SAT	A2DIGL_SOFT_SAT cannot occur without having hard sate working also.	A2DIGL	Default is 0.
GSM_INIT	Slope of the initial linear AOC drive ramp in GSM mode	A2DIGL	10-bit

Table 31-1. PAC Module Pin List (Continued)

Port Name	Function	to/from	Comments
ERR_GAIN1	Selects the number of bit shifts right of the error signal fed to the AOC accumulator input before BW_DLY expires. The error gain has 12 steps. The error gain can be selected from 1 to 1/2048 for LTE.	A2DIGL	4-bit 0000 gain 1 0001 gain 1/2 0010 gain 1/4 ..... 1100 gain 1/4096(Highest Gain) ..... 1111 gain 0 (LTE)
ERR_GAIN2	Selects the number of bit shifts right of the error signal fed to the AOC accumulator input after BW_DLY expires. (Gain selection is the same as above.)	A2DIGL	4-bit gain as above
PROP_GAIN1	Selects the number of bit shifts right of the error signal fed to the AOC accumulator output before PROP_DLY expires. The error gain has 12 steps. The error gain can be selected from 1 to 1/2048 for LTE.	A2DIGL	4-bit 0000 gain 1 0001 gain 1/2 0010 gain 1/4 ..... 1100 gain 1/4096(Highest Gain) ..... 1111 gain 0 (LTE)
PROP_GAIN2	Same as PROP_GAIN1 above.	A2DIGL	4-bit
OFS	Offset power level for systems which uses the internal look up table for PA ramp shaping	A2DIGL	8-bit
AOC_GAIN	Selects the gain of the differential amplifier in front of the AOC A/D converter	A2DIGL	2-bits 00=>gain 1 01=>gain 3.16 1X=>gain 10
EST_DLY	Sets the number of clocks from the rising edge of DMCS to the end of the estimation stage	A2DIGL	14-bit
INIT_DLY	Selects the number of clocks from the rising edge of DMCS to the start of the estimation stage	A2DIGL	14-bit
ACC_INIT	Estimation state value for the D/A	A2DIGL	10-bit
DN_DLY	Sets the number of clocks from the falling edge of DMCS to power down of the TX section. None of the hold off timers change or reset until after DN_DLY expires.	A2DIGL	14-bit
DAZERO	If set to high the output of the smoothing filter is shorted out and the AOC output ports pulled low after the DN_DLY timer has expired	A2DIGL	1-bit

Table 31-1. PAC Module Pin List (Continued)

Port Name	Function	to/from	Comments
PRoff	If programmed high then the proportional gain branch is disabled	A2DIGL	1-bit
AOC_FORCE	Value which AOC is forced if AOC_MANUAL is programmed high	A2DIGL	10-bit
ACT_SEL	Logic high selects the digital activity comparator, logic low selects the analog activity comparator.	A2DIGL	1-bit
AOC_MANUAL	If set high then the AOC D/A is driven by AOC_FORCE, and the AOC_FORCE values are loaded as soon as they are written by the SPI. If set low the AOC D/A is driven by the accumulator path.	A2DIGL	1-bit
ADI_REF	Sets the division ratio from the main clock to the ADC clock.	A2DIGL	7-bit
AOC_FBW	Selects the bandwidth for analog filter following DAC path	A2DIGL	Not Implemented!
CDET_DLY	CDET_DLY is also used as a SPI programmable delay timer. CDET_DLY is the delay from DMCS to the time when the SPI readback parameters are latched into their respective registers. It also sets the number of clock cycles from the rising edge of DMCS to triggering the latching of cross-over detection.	A2DIGL	18-bit
AOC_MAX	Maximum permitted value of the AOC D/A. If the value from the accumulator attempts to exceed this value then AOC_MAX is held at the D/A input and the clock to the accumulator is removed and held low until a negative ERRORSIG occurs.	A2DIGL	10-bit
RAMP_REF	Sets the division ratio from the main clock to the LUT ramp clock	A2DIGL	6-bit
LOW_BATT_EN	Enables LOW_BAT detection. When LOW_BATT_EN is programmed high and at the falling edge of LOW_BAT the accumulator clock shall be suspended till expiration of DN_DLY timer even if LOW_BAT signal goes high again.The PAC analog output shall be driven to zero volts. If the LOW_BATT_EN bit is programmed low, the low battery detection is disabled.	A2DIGL	1-bit
LOW_BAT	Battery power low detected from Seaweed. Disables driver for the PA, active low	from Pin	1-bit
DET_HIGHV	If DETECT_IN voltage is <2.5V, this bit should be set high.	A2DIGL	No longer utilized with 2.5V PA_DET range.

Table 31-1. PAC Module Pin List (Continued)

Port Name	Function	to/from	Comments
a2di_soft_set	Activates current source on REF pin. Used to maintain 10mV DC offset at activity detector input when DET_HIGHV is active.	A2DIGL	No longer utilized with 2.5V PA_DET range.
AOC_HOLD	If programmed high then the clock to the AOC_HOLD register (Output register) is removed thus forcing a hold state when DIV_DLY expires.	A2DIGL	1-bit
BACKWARD	Applies PWR_RAMP word directly to AOC DAC input	A2DIGL	1-bit
CDMA_DN	If programmed high then when RAMP_DN_DLY expires the feedback loop will revert to ERR_GAIN1 and PROP_GAIN1.	A2DIGL	1-bit
a2di_tcode[6:0]	Precision tuning of integrated filters	TUNEC	7-bit
anatest_tmux	Test Mux, Only used in function a mode for Analog Test evaluation	ANATEST	8-bit
ABF_SEL	If programmed high then BSAVE_TX is set true after DN_DLY expires.	A2DIGL	1-bit
ABR_SEL	If programmed high then BSAVE_TX is set false after DN_DLY expires.	A2DIGL	1-bit
DAC_TEST	Enables clocking of DAC test counter	Test	DAC test counter run off of mainclk, will be enabled with DAC_TEST. This counter will input to the DAC in test mode.
ADC_IBIAS	10 uA current bias signal	REGUL	1-bit
DET_FLT	Sets the 3dB cutoff frequency of the analog LPF in front of the ADC	A2DIGL	1- bit: 0 ==> 600KHz 1 ==>1.2 MHz
vdd	supply for digital circuits	from Regulator	1.58V +/- 6% approx 5mA
vdda_pac	supply for analog circuits	from regulator	2.5V +/- 5% approx 30mA
vdda_trsynth	supply for analog DAC circuits at 1.58V	from regulator	1.58V +/- 5% approx 10mA
vss	supply for digital circuits	supply	
vssa_pac	supply for analog circuits	supply	
vssa_trsynth	supply for analog DAV circuits	supply	
vag_ref	accurate reference voltage	from PIN	1.575V reference
OUTPUTS			

Table 31-1. PAC Module Pin List (Continued)

Port Name	Function	to/from	Comments
scan_out	Scan chain output	Dedicated o/p pin	Active during TM 8-bit
PWRDAC_SPI	SPI read back value of the AOC D/A input before DLPF	A2DIGL readable	10-bit
PWRUNFLT_SPI	SPI read back value of the AOC ACC input minus DC_OFFSET	A2DIGL readable	10-bit
ERRORSIG_SPI	SPI read back value of the difference between the Power Ramp and ADC input values	A2DIGL readable	11-bit
CDET_OUT_SPI	SPI read back value of crossover detect value	A2DIGL readable	1-bit
GPA2D_sync	Output activity signal for General Purpose A2D	GP ADC	1 bit rising edge signal
DN_sig	Output to TX block to turn off the modulator in the synthesizer.	TX block	DN_DLY after falling DMCS.
SSI_sig	Hold off timer output.	TX block	SSI_DLY after falling DMCS.
BSAVE_TX	If programmed high the TX block is placed in standby mode.	TX block	1-bit
AOC_DRIVE	Power Amp Control	PA	Output to pad PA_OUT
Test1	Analog Bi-directional test port	Anatest	1-bit
Test2	Analog Bi-directional test port	Anatest	1-bit
VPA	VSS sense to ground	star connected vssa pin	1-bit
gpio_adc	PAC ADC data to GPIO	ADC controller to GPIO	5-bit
coarse_fine	PAC sync tp GPIO	ADC controller to GPIO	1-bit
pac_dig_edge_mode	Edge mode indicator	txdigital	Set when ramp_sel = 3'h2

31.3 Digital PAC

This section describes the Power Control Module to generate the correct digital wave shape for the PA.

The digital section consists of the following main blocks, ramp look up table (LUT), a saturation detection block, a proportional path, an integral path, level / error generation, activity detector, synchronization section and a filtering section.

The saturation detection and ramp LUT are only used in GSM. The detection block is a digital comparator monitoring the difference between the power control loop reference and feedback from the PACII IC as read by the AOC A/D converter. If the transmit chain is incapable of driving the power out to its target value, the PACII feedback will not rise enough to cause the AOC A/D output to match the digital word corresponding to the desired output power level (PWR). The threshold of the error which will cause the comparator to trip is set by the SPI word SAT_TH[9:0]. If the comparator trips then PWR will be decremented by the SPI word SAT_STEP[4:0] on each occurrence of the Ramp Clock until the saturation condition is cleared. The start of the saturation detection is delayed relative to DMCS by the SPI word SAT_DLY[11:0].

The proportional path is formed by a variable gain block connected from the error detected between the detected signal and the desired PWR setting to the input of the AOC D/A. The gain of this block is set by the SPI fields PROPGain1[3:0] and PROPGain2[3:0]. The proportional gain is switched between these two fields at the expiration of the timer set by the SPI field PROP_DLY[11:0]. The gain of the proportional path is useful in aiding the stability of the closed loop system. The proportional path can be disabled if the PRoff SPI bit is programmed high.

The integral path consists of a digital accumulator with a variable clock rate. The clock rate for the accumulator is selected by the field ADI_REF[11:0].

Digital level / error generation subtracts a DC offset from the detected signal, if the subtraction results in a negative number a zero is forced. The detected word is subtracted from the desired level of the ramp. This results in an error signal between the ramp step and the PA output voltage.

When ACT_SEL and ACT_EN are programmed high, the digital activity detector is used as a comparator to determine when the power detector becomes active. As soon as the input detected voltage minus the DC_OFFSET rises above the activity threshold voltage the comparator trips.

The outputs of the proportional path and the integral path are summed and fed to the input of the AOC D/A. A maximum value of the D/A may be set with the AOC_MAX[9:0] word. If the control system attempts to exceed this word then the D/A input is held at the AOC_MAX word and the clock to the accumulator and output register is removed and held low. The clocks are held low until the error detected between the detected signal and the desired PWR settings is negative or until DN_DLY expires.

RAMP_DN_DLY is a hold timer required for specifying ramp down timing.

31.3.1 PAC Reset

RESETB is a active low pulse. All the SPI PAC bits will be reset to 0.

31.3.2 Battery Save

The A2DIGL will use DN_sig from the PAC to create the BSAVE for the PAC and TX blocks. TX and PAC will both use the BSAVEpac signal from the A2DIGL. A2DIGL will use the spi bits ABF_SEL and ABR_SEL to determine when the TX and PAC will be in battery save.

31.3.3 Low Battery Detection

The LOW_BATT_EN bit shall enable LOW_BAT detection. When LOW_BATT_EN is programmed high and at the falling edge of LOW_BAT the accumulator clock and output register clock are suspended until expiration of DN_DLY timer. The AOC analog output shall be driven to zero volts (output shall be shorted to ground). To reiterate this point, it is important that once LOW_BAT goes low, the accumulator clock and output register clock will remain suspended till the expiration of DN_DLY even if the LOW_BAT signal goes high again. This will avoid an unstable condition. If the LOW_BATT_EN bit is programmed low, the low battery detection is disabled.

31.3.4 PAC Clocks And Timing Diagrams

The main clock in the PAC is mainclk (corrected 13MHz).

For Neptune the AOC_REF will be assigned the same value ADI_REF which defines the ADC conversion rate, and accumulator clocks.

Every clock process is synchronous to the positive edge of mainclk. The following clock enables are generated inside the PAC and used with mainclk: AOCOUT_clk (Accumulator enable), RAMP_clk (Look Up Table enable), AOCADC_clk (ADC enable) and OUTREG_clk (Output register enable). In order for the PAC to work correctly the AOCADC_clk and AOCOUT_clk must be at the same frequency, this is controlled by the SPI bits ADI_REF[6:0] and ADI_REF[11:0].

All the clock counters are reset on the rising edge of DMCS. All the clock enables are synchronized by this reset including the ADC clock enable (AOCADC_clk) and accumulator clock enable (AOCOUT_clk).

The clock applied to the accumulator register is phased delayed by 2 mainclk periods from AOCOUT_clk, delayed by 3 mainclk periods to the output register. Therefore there is two mainclk periods from the rising edge of AOCADC_clk to the rising edge enable to the accumulator register.

The RAMP_clk counter is reset on the rising edge of DMCS. The RAMP_CLK is held low. When RAMP_DLY expires the internal signal RAMP_sig goes high. On the next rising edge of mainclk the RAMP_CLK signal goes high and remains high for one mainclk period. The next rising edge of RAMP_clk will depend on the RAMP_SEL value. This sequence of events can be seen in the “RAMP_clk And Saturation Timing Diagram”. RAMP_clk is used to access the lookup table and RAMP_clk_D1 which is a delayed version is used to clock the saturation detect circuit and apply the result to the ERRORSIG mux. RAMP_DLY is programmed by the SPI bits RAMP_REF[5:0].

The PAC expects DMCS to occur on the negative edge of mainclk. DMCS_P is generated from a rising edge detect register. DMCS is registered with mainclk (13MHz) inside the PAC.

Table 31-2. Clock Ratios

CLK	Ratio	Ratio Indicator
RAMP_clk	1 / 12	RAMP_REF
AOCOUT_clk	1 / 5	ADI_REF
AOCADC_clk	1 / 5	ADI_REF

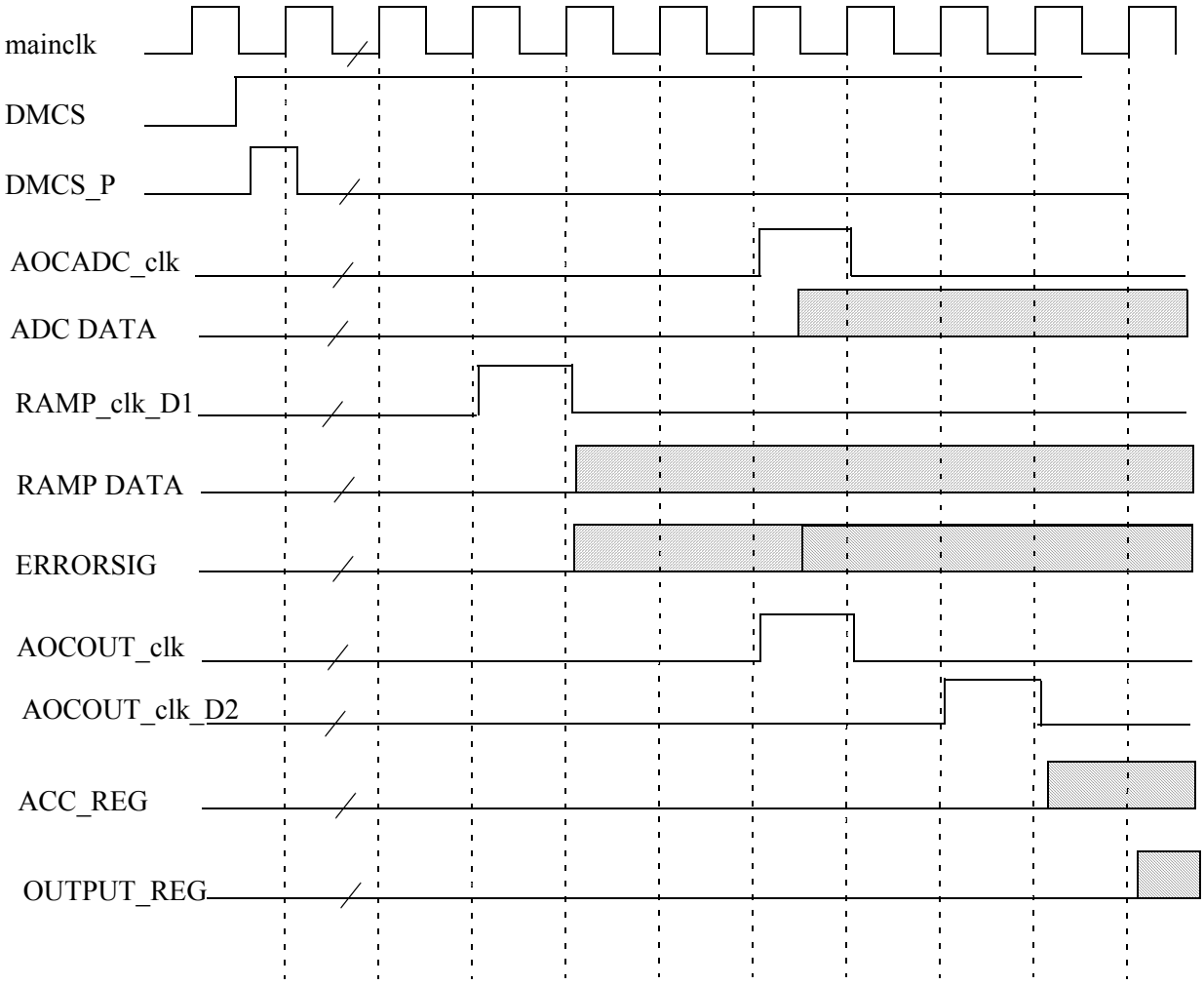


Figure 31-4. PAC Clocking Example

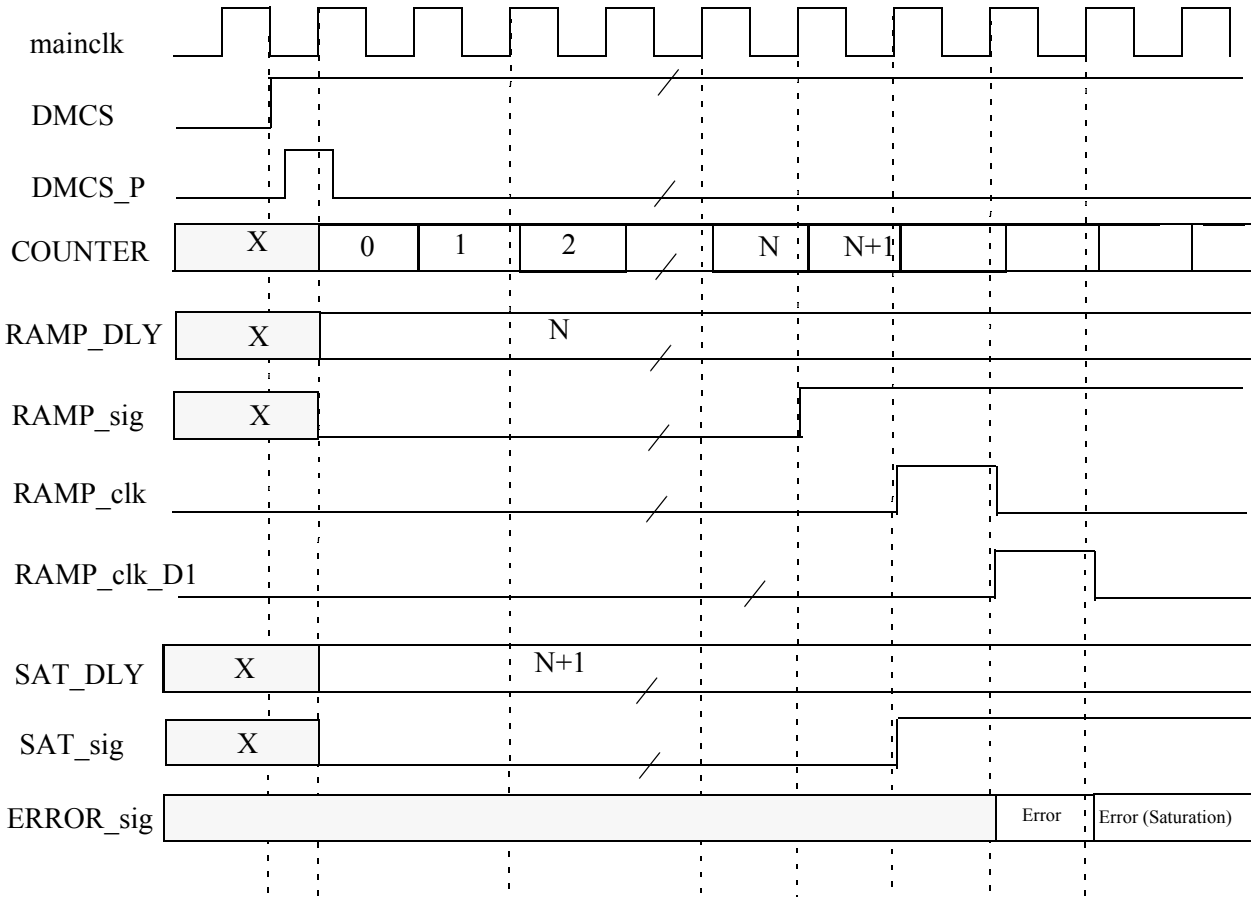


Figure 31-5. RAMP_clk And Saturation Timing Diagram

The first ERROR_sig value is valid on the next mainclk rising edge when RAMP_clk is high. This ERROR_sig value is compared with the SAT_TH value which determines if a SAT_STEP decrement to the power level is required. If required the new decremented power value is valid on ERROR_sig on the next mainclk rising edge when RAMP_clk_D1 is high. There is a phase difference of 1 mainclk cycle between RAMP_clk and RAMP_clk_D1. This allows saturation detection to work on the current error value without waiting a whole RAMP_CLK period.

31.3.5 Ramp Down

RAMP_DN_DLY is a delay for specifying ramp down timing. When RAMP_DN_DLY expires the ramp LUT table is accessed and the ramp value is decremented in 8 steps indicated by the internal signal DN_RAMPsig being high. DN_RAMPsig goes high after RAMP_DLY expires and goes low after 8 mainclk cycles. It indicates the ramping down of the required power signal. After DN_RAMPsig goes low the OFS value is removed from the required power value. This allows the PAC system to ramp down to zero in a number of smooth transitions. When DN_DLY expires the input to the DAC is pulled low and the accumulator and output registers are cleared.

DN_DLY clears the SAT detect register and is output to the TX block to turn off the modulator in the synthesizer. If DAZERO is set to high, the output of the smoothing filter is shorted out and the AOC output ports are pulled low, after the DN_DLY timer has expired.

When low batt detection is enabled and when a falling edge of LOW_BATT is detected after the rising edge of DMCS, the PAC output shall be shorted to GND until DN_DLY expires. The response time shall be 1us as measured from 50% of falling edge of LOW_BATT to 50% PAC output voltage. Also, if low batt detection is enabled and LOW_BAT (from seaweed) is low at the rising edge of DMCS, the PAC output shall be shorted to ground until the expiration of DN_DLY.

31.3.6 GSM Look Up Tables

There are 2 GSM look up tables: Half raised and quarter half raised cosine selected by the SPI bits RAMP_SEL[2:0] These are both eight step ramps, see table below. Ramp clock is 12 times slower than the mainclk, which is 13 MHz. The gives an approximate 8 micro second ramp time.

Table 31-3. Ramp Values

Step	Half Raised Cosine	Quarter Raised Cosine
0	00 H	00 H
1	02 H	0D H
2	09 H	18 H
3	13 H	23 H
4	1F H	2D H
5	2C H	34 H
6	36 H	3A H
7	3D H	3E H
8	3F H	3F H

31.3.7 Activity Detector

The activity detector is a comparator used in the GSM mode to determine when the power detector becomes active. As soon as the RF Detect voltage rises above the DC reference voltage the comparator trips. The comparator will act to terminate the loop precharge. The precharge must respond to changes at the input of the Activity Detector within 250 nS.

A choice of an analog or a digital activity comparator is provided via ACT_SEL bit. If ACT_SEL is programmed low then an analog activity comparator is used. The analog activity comparator input ANA_ACT_IN is registered with the 13MHz clock in order to align it with the output of the ADC. This shall insure the proper GSM_INIT mux timing also. If ACT_SEL is programmed high then the digital comparator is used. The advantage of the digital comparator is that the threshold may be programmed.

Normally the activity detection scheme is used in the GSM transmit mode. Activity Detection may be disabled if ACT_EN is programmed low.

An analog averaging filter may be activated on the PA_DET input. AOC_GAIN is registered on the rising edge of DMCS and holds its value for the duration of the transmit burst.

CDET_DLY is also used as a SPI programmable delay timer. CDET_DLY is the delay from DMCS to the time when the SPI readback parameters are latched into their respective registers. There are 4 read back spi values: CDET_OUT_SPI, ERRORSIG_SPI, PWRUNFLT_SPI and PWRDAC_SPI.

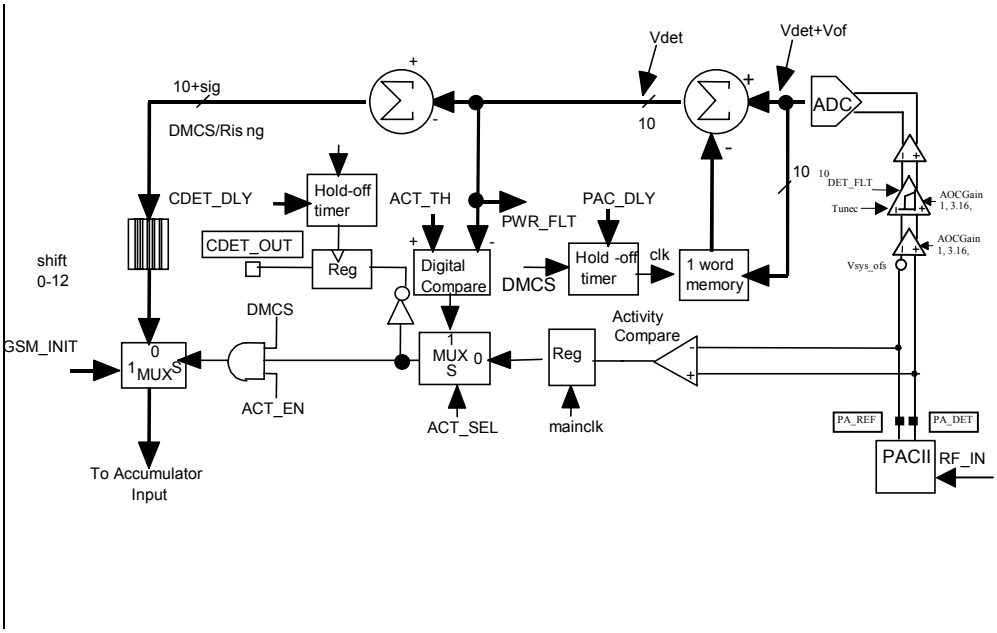


Figure 31-6. The Activity Detection Section of the AOC Block Diagram

Note that for the analog comparator the PACII outputs are connected such that the comparator output is false until the detected output level results in an approximately 10mV increase in the PA_DET voltage. See analog section of this chapter for activity detect comparator specifications

31.3.8 SAT Detector And Waveform Generator

The SAT Detector is only used in GSM mode. It is a digital comparator monitoring the difference between the power control loop reference and feedback from the PACII IC as read by the AOC A/D converter. If the transmit chain is incapable of driving the power out to its target value, the PACII feedback will not rise enough to cause the AOC A/D output to match the digital word corresponding to the desired output power level (PWR). The threshold of the error which will cause the comparator to trip is set by the SAT_TH[9:0]. If the comparator trips then PWR will be decremented by the word SAT_STEP[4:0] on each occurrence of the Ramp Clock until the saturation condition is cleared. The start of the saturation detection is delayed relative to DMCS by the word SAT_DLY[11:0].

PWR is registered on the rising edge of DMCS. In the current implementation SAT_TH, SAT_DLY, SAT_step and SAT_EN are registered on the rising edge of DMCS and remain static for the duration of the DMCS slot.

Two ramp shapes are stored in a look up table and clocked out of it via RAMP_CLK. The ramp shape is selected by the SPI bits RAMP_SEL. The RAMP_CLK rate is set by the variable RAMP_REF[5:0]. The first rising edge of RAMP_CLK is held off to RAMP_DLY[13:0] reference clock cycles past the rising edge of DMCS. Once the first pulse of RAMP_CLK occurs, subsequent pulses appear at the rate set by RAMP_REF. For the falling ramp, the selected ramp shape is played out of the lookup table in reverse order after RAMP_DN_DLY reference clock cycles past the falling edge of DMCS.

Table 31-4. Ramp Look-up Table

Half-Wave Cosine	Quarter-Wave Cosine
Step(0)=00H	Step(0)=00H
Step(1)=02H	Step(1)=0DH
Step(2)=09H	Step(2)=18H
Step(3)=13H	Step(3)=23H
Step(4)=1FH	Step(4)=2DH
Step(5)=2CH	Step(5)=34H
Step(6)=36H	Step(6)=3AH
Step(7)=3DH	Step(7)=3EH
Step(8)=3FH	Step(8)=3FH

The LUT shall remain at Step(0) until the expiration of RAMP_DLY. At the expiration of RAMP_DLY timer, the LUT shall clock through Steps 1 to 8 for every clock cycle of RAMP_CLK and hold at Step(8). At Step(8) the multiplication of the LUT value and PWR setting value is forced to the PWR setting, therefore achieving the multiplication factor of 1.

The LUT shall cycle back down at the expiration of RAMP_DN_DLY timer. The LUT shall step down from Steps(8) to Step(0) in eight clock cycles RAMP_CLK (13/12 MHz).

31.3.9 Proportional and Integral Control

The feedback from the error signal to the AOC D/A consists of a proportional path and an integral path.

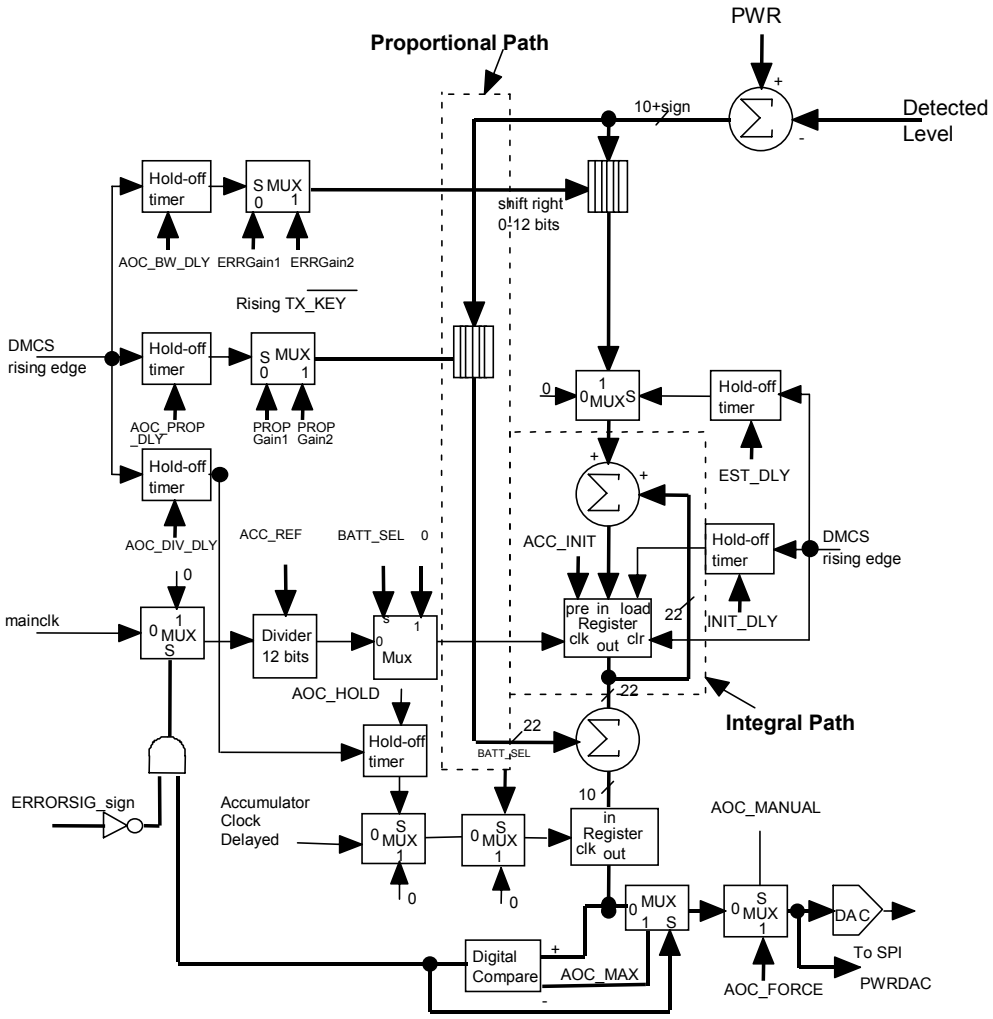


Figure 31-7. The Proportional and Integral Control Section of the AOC Block Diagram

The proportional path is formed by a variable gain block connected from the error detected between the detected signal and the desired PWR setting to the input of the AOC D/A. The gain of this block is set by the PROPGain1[3:0] and PROPGain2[3:0] fields. The proportional gain is switched between these two fields at the expiration of the timer set by the field PROP_DLY[11:0]. The gain of the proportional path is useful in aiding the stability of the closed loop system. The proportional path can be disabled if the PRoff bit is programmed high.

The integral path consists of a digital accumulator with a variable clock rate. The clock rate of the accumulator is selected by the field ADI_REF[11:0].

The input to the accumulator is the error between the detected signal and the desired PWR setting. This error is passed through a variable gain set by the fields ERR_GAIN1[3:0] and ERR_GAIN2[3:0]. The gain is switched between these two fields at the expiration of the timer set by the field BW_DLY[11:0]. This gain setting will determine the closed loop bandwidth of the controller.

If a value of 4’hf is set for either err_gain or prop_gain, then they have a gain of zero.

If CDMA is programmed high then when RAMP_DN_DLY expires the feedback loop will revert to ERR_GAIN1 and PROP_GAIN1.

The outputs of the proportional path and the integral path are summed and fed to the input of the AOC D/A. A maximum value of the D/A may be set with the AOC_MAX[9:0] word. If the control system attempts to exceed this word then the D/A input is held at the AOC_MAX word.

Finally, AOC values may be forced into the D/A by the word AOC_FORCE[9:0] if AOC_MANUAL is high. This will allow open loop control of the PA at low power levels. If ACT_EN is programmed high then the loop will leave the forced mode (GSM_INIT) once the detected power exceeds ACT_TH.

Table 31-5. Path Gain of the Proportional Path and the Integral Path

Parameter	Min	Typ	Max	Unit
Integral Path Gain	1/4096		1	
Proportional Path Gain	1/4096		1	

31.4 DMCS Counters And Hold-Off Timers

There are two 14 bit DMCS counters used to synchronize the ramping stages.
The rising DMCS counter is reset with the rising edge of DMCS. It will count to its max value less one, but will not wrap around and count again, i.e once it reaches its max value less one. It holds it until it is reset again by the next rising DMCS edge. If a signal is not to be set, the delay time is set to the max value of the counter.

The falling DMCS counter is reset on the rising edge of DMCS. The counter will remain static until the falling edge of DMCS. The counter starts from the falling edge of DMCS and increments on each mainclk rising edge. It counts to its max value less one i.e. “3FFE” and remains there until it is reset again. If a delay time is set to “3FFF” it will never match the counter value and so will not be set.

Example: DMCS falling edge count is reset to zero on the rising edge of DMCS but does not start counting until the falling edge of DMCS. When DN_DLY expires the signal DN_sig goes high and remains high until the next rising edge of DMCS when it is set low.

The counters are clocked with mainclk (13 MHz) and reset on the rising edge of DMCS.

31.5 PAC Analog Blocks

31.5.1 ADC_CTL

LTE uses a RSD ADC which is clocked from the 26MHz clock. To observe the ADC data in real time the output signals GPIO_ADC[4:0] and COARSE_FINE are made observable via the GPIO port.

31.5.2 PAC_ANA Analog Blocks

The analog portion of the Power Control system is responsible for getting the most accurate 10-bit representation of the detected output power, and delivering the appropriate analog voltage to the PA.

. The differentially detected output power level (PA_DET - PA_REF) is passed through a programmable gain stage with a single pole filter and then through a sample and hold stage before entering the PAC_ADC. A 10-bit A/D is performed on the differential signal at 2.6Msamples/sec rate That 10-bit word is passed to the digital processing section of the PAC.

There is an analog comparison of the PA_DET signal and a reference signal PA_REF. This indicates detected RF output power and is passed to the digital processing block. This signal is not synchronized or latched with the 13MHz clock in the analog section.

The 10-bit current DAC takes the data from the digital section and generates a representative voltage, which is filtered appropriately before being buffered and passed to the output pad.

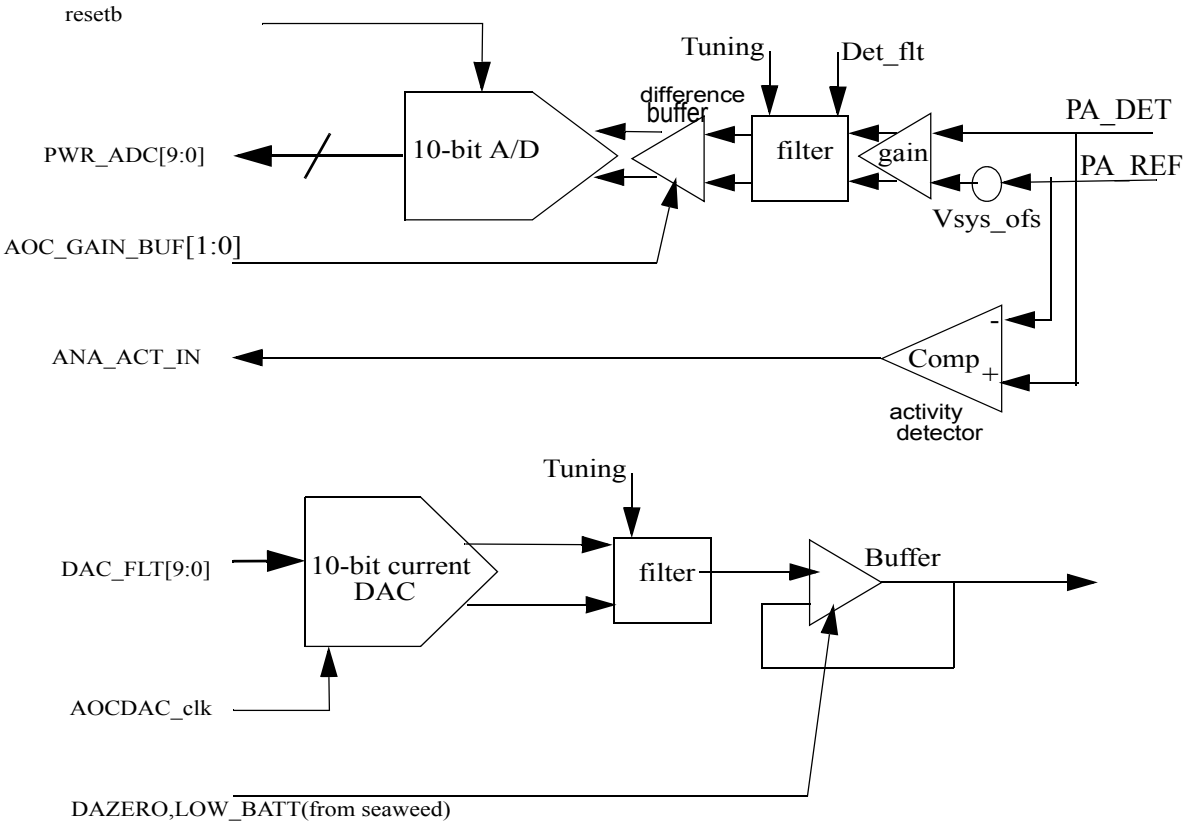


Figure 31-8. Analog Modules

31.5.3 PACII Power Detector

The power detector will be a PACII type design. It will be necessary to design the detector output voltage so that it's output at peak power is near the maximum input voltage of the AOC Gain buffer. This will maximize the dynamic range of the power control system.

DET_IN and REF of PACII (PA_DET and PA_REF on Neptune), should be DC loaded as little as possible. Significant DC loading will degrade the temperature stability of the PACII detector and reference. The PA_DET must be able to handle at least up to 2.3V input within ADC Dynamic range and to 2.5V which may be outside the ADC dynamiv range.

Table 31-6. PACII Power Detect Specifications

Parameter	Min	Typ	Max	Unit
PA_DET @ 16 dBm Pin, 50 ohm source, relative to zero applied signal	1.1	1.3	1.5	V
PA_DET @ -17 dBm Pin, 50 ohm source, relative to zero applied signal	5		15	mV
PA_DET @ 0 mW RF input	48	50	52	mV
PA_REF @ mW RF input	58	60	62	mV
DC load at PA_DET and PA_REF	100			KOhm
Det. Offset; PA_DET - PA_REF@0mW RF input	-5	-10	-15	mV

A simulated plot of the PACII detector output versus the input power is shown below. This is derived with a 50 Ohm source impedance

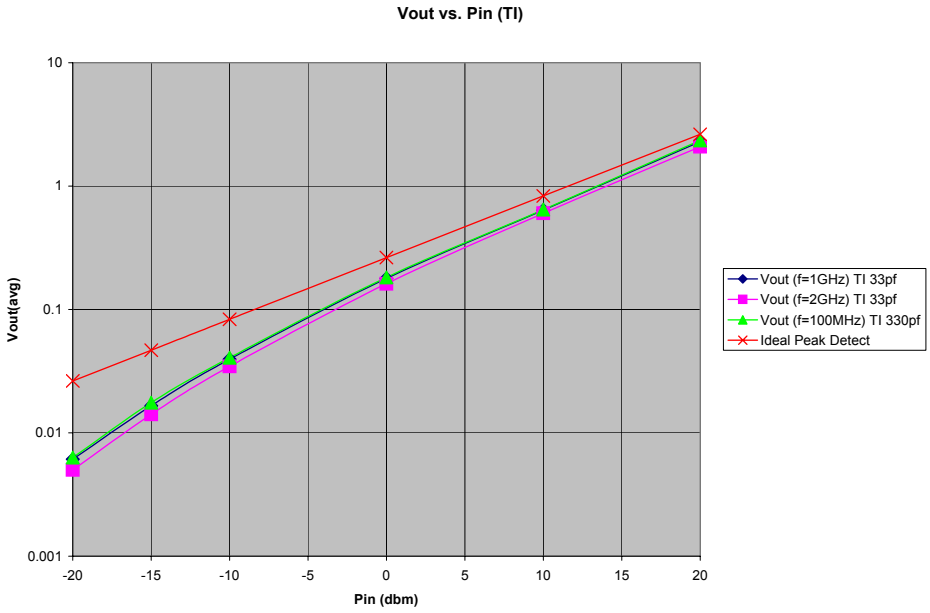


Figure 31-9. PACII Power Detector Response

31.5.4 PAC Input AMP, Comparator, and ADC

Table 31-7. Operating Conditions

Parameter	Min	Nom	Max	Unit
Analog Supply Voltage	2.35	2.5	2.65	V
Operating Temp, Simulation	-40		110	°C

31.5.5 Difference Buffer

The PAC_INP_COND stage is connected between the PACII outputs and the PAC_ADC input. Since PACII will have PA_REF 10mV greater than PA_DET with zero input power, the Input Conditioning adds an offset to produce a positive differential signal to the ADC for all power levels. The Nominal gains are scaled by 0.6 as the ADC input range is narrower than the PA_DET signal range. This will produce a output an output = [G1*Vi +(1+G1)Vsys_ofs] * G2, where Vi is the differential signal PA_DET - PA_REF, G1, G2 are gain factors dependent on the AOC_GAINsetting and Vsys_ofs is a offset introduced to compensate for the negative differential form the PACII detector at low power. Vsys_ofs is set to 19mV. The effective offset for each gain setting is specified later in this chapter.

Table 31-8. PAC Input Conditioning Gain Settings

AOC_Gain	Nom Gain	G1	G2	Actual Gain	A/D IN*
00	1	0.6	1	0.6	16.4mV
01	3.16	1.89	1	1.89	21.56mV
10,11	10	1.89	1.89	6	40.7mV

* A/D input for zero power level, i.e. PA_DET - PA_REF = -10mV

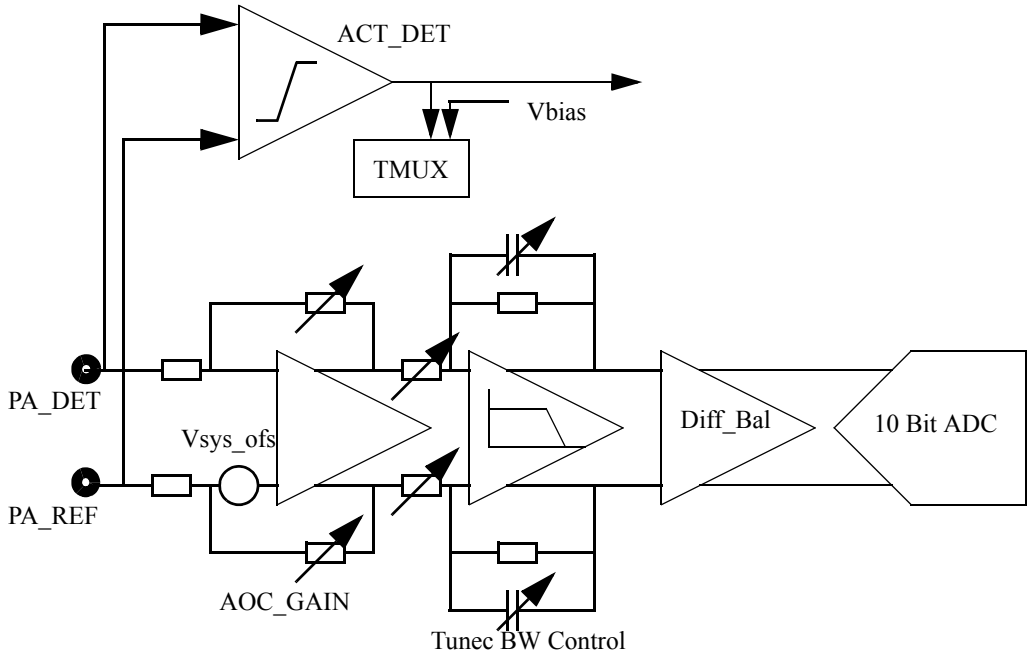


Figure 31-10. PAC INP COND

Table 31-9. PAC INPUT Specifications

Parameter	Min	Nom	Max	Unit
Input CM Range	0		2500	mV
Gain Settings		1,3.16,10		
Gain Tolerance	95		105	%
SlewRate	3			V/µS
CMRR			60	db
Supply Current (Active)		2000		µA
Supply Current (Disabled)			1	µA

31.5.6 Activity Detect

Table 31-10. Analog Activity Detect Specifications

Parameter	Min	Nom	Max	Unit
Trip Threshold	-3.5		5.0	mV
Comparator Delay			125	nS
PA_DET Input Range	0		2500	mV
PA_REF Input Range	0		250	mV

Table 31-10. Analog Activity Detect Specifications (Continued)

Parameter	Min	Nom	Max	Unit
Supply Current (Active)		400		µA
Supply Current (Disabled)			1.0	µA

31.5.7 Low Pass Filter

The anti-aliasing filter is a single pole filer programmable by the DET_FLT bit. When DET_FLT is 0 the 3dB cutoff frequency is 600 KHz nominal. When DET_FLT is 1 the 3dB frequency is 1.2 MHz nominal. The filter is tuned by the 5 MSB's of the Tunec Code from the Neptune Tunec circuit. This tuning code is updated before each transmit burst.

Table 31-11. PAC Input Filter Specifications

Parameter	Min	Nom	Max	Unit
Filter Corner (DET_FLT = 0)	0.5	0.6	0.7	MHz
Filter Corner (DET_FLT = 1)	0.9	1.2	1.5	MHz
Supply Current (Active)		1000		µA
Supply Current (Disabled)			1.0	µA

31.5.8 PAC 10-bit ADC

A 10-bit RSD architecture will be used to realise the A/D convertor. The clocking will be supplied by the digital circuits. The convertor will be initialized by resetb.

Table 31-12. ADC Specifications

Parameter	Min	Nom	Max	Unit
Simulated Operating Temp	-40	27	110	°C
resolution		10		bits
sample rate		2.6		MHz
INL	-3		3	LSB
DNL	-1		1	LSB
clock input		13		MHz
Ref. Voltage (from seaweed 2%)	1.545	1.575	1.605	V
Supply Current		5000		µA

31.5.9 Voltage Offset Requirements for Input Conditioner and PAC ADC

The compensating offset introduced at the PAC ADC input conditioning is dependent on the AOC_GAIN setting. This is specified for zero differential voltage (PA_DET = PA_REF) for each gain setting with the limits shown below. The limits allow for input conditioning and ADC error offsets which accumulate.

Table 31-13. PAC Input & ADC Offset and Dynamic Range

Parameter	AOC_GAIN	Min	Nom	Max	Unit
ADC Output @ PA_DET = PA_REF =60mV	00: x1	6	21	35	LSB
	01: x3.16	18	38	55	LSB
	10: x10	60	120	160	LSB
PA_DET Input Low for ADC dynamic Range ADC Code = 0 PA_REF =60mV	00: x1	-30	9	45	mV
	01: x3.16	15	31	45	mV
	10: x10	19	31	45	mV
PA_DET Input Range Hi for ADC dynamic Range ADC Code =1023 PA_REF =60mV	00: x1	2345	2509	2670	mV
	01: x3.16	767	822	877	mV
	10: x10	256	281	309	mV
PA_DET Range, (not guaranteed in ADC dynamic Range	All	-0.05		2.6	V

31.5.10 PAC D/A Chain

The D/A chain consists of a digital low pass filter, the D/A itself, a unity gain buffer, an anti-imaging filter, and a driver network. The D/A chain section of the AOC block diagram is shown below.

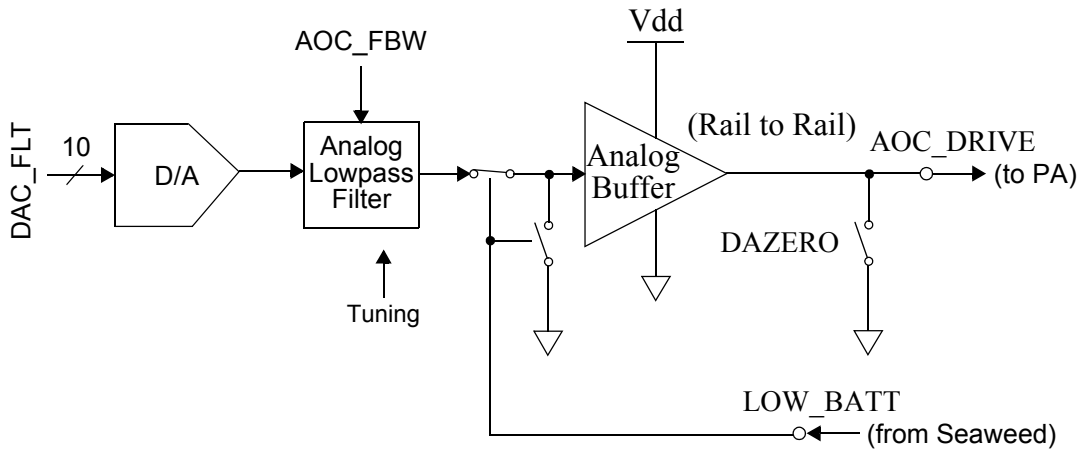


Figure 31-11. The D/A Chain Section of the AOC Block Diagram

The AOC_DRIVE shall be buffered with an analog buffer amplifier and have the capability to sink or source 1mA with a drop out voltage of less than 100mV. Thus, the output shall be capable of driving down a level of 100mV volts maximum at a zero D/A setting and up to a minimum level of 2.2 volts at max D/A setting, with 1mA loading (sink or source). The analog buffer shall have a gain such that these levels are obtained when the 10 bit D/A is programmed to all 1's (full scale) or all 0's. The analog buffer shall also have the capability to sink and source 3mA

An option shall be added to safely short the AOC_DRIVE pin to ground. This shall automatically happen at the end of the ramp down period if the control bit DAZERO is set high. This option is included to eliminate any offset which may cause the transmitter to be activated in off times.

Also, the AOC_DRIVE shall be controlled by a LOW_BATT signal generated by Seaweed IC. On a HIGH to LOW transition of the LOW_BATT signal input from Seaweed, the input of the analog buffer shall be disconnected from the analog lowpass filter and shorted to ground. Also, the accumulator clock and output register clock shall be suspended. This shall occur until the expiration of the DN_DLY timer. The feature may be enabled or disabled via a control bit LOW_BATT_EN.

The D/A is a 10 bit D/A. The conversion rate of the D/A is (13/5) MHz.

The PAC D/A Chain shall be bounded by the curve shown below. On a part to part basis the output voltage at code 256'd shall deviate by no more than +/-100mV. There shall be two regions defined in the transfer function of the DAC "Deadzone" and "Saturation Zone". The "Deadzone" region shall be the minimum code to produce an output voltage of 100mV. The "Saturation zone" shall be defined as the minimum code to produce an out put voltage of 2.2V. The PA D/A chain must be linear over the range 0.1 to 2.2V. The PA D/A chain can exhibit nonlinear behavior when the output voltage is less than 0.1V and greater than 2.2V

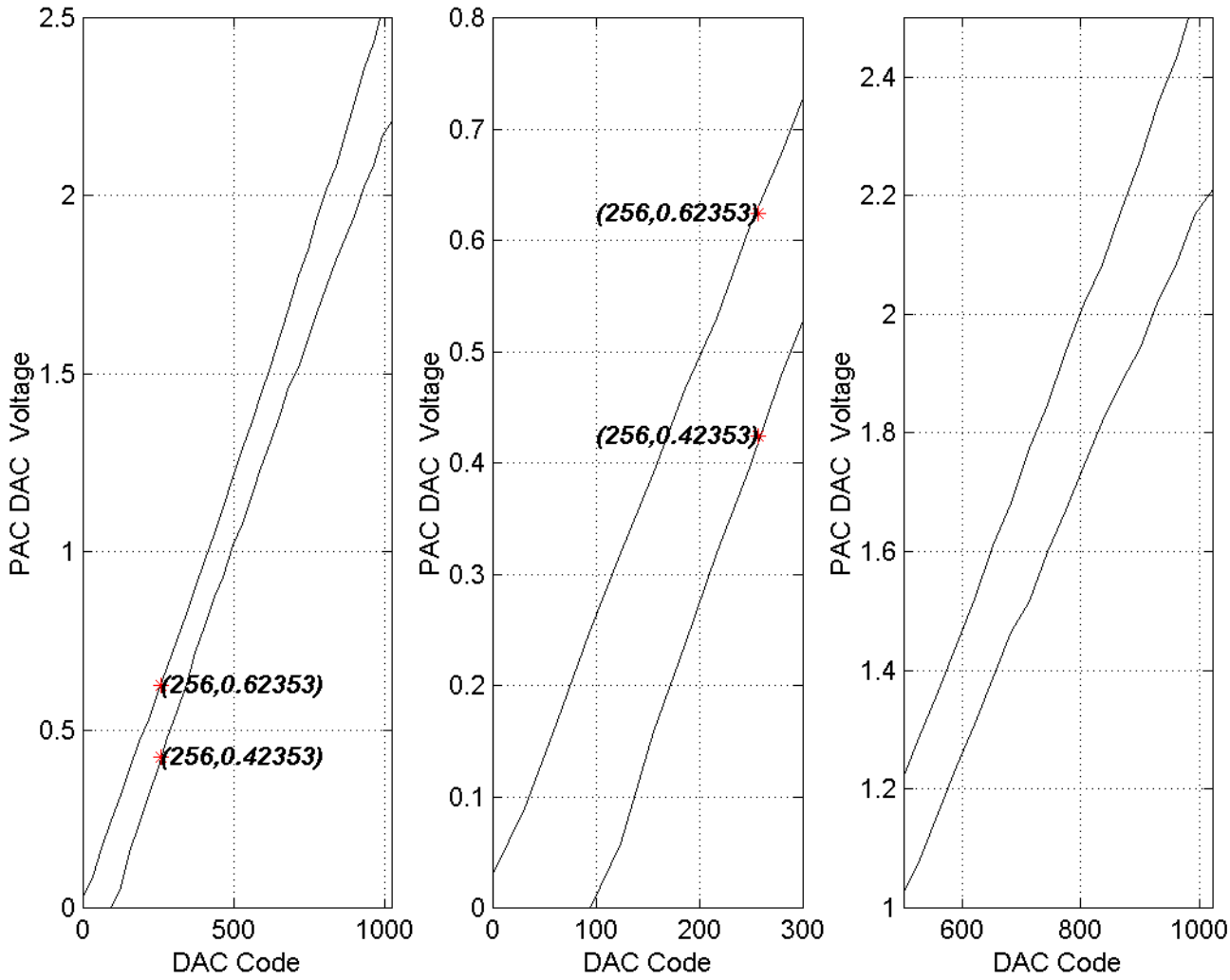


Figure 31-12. PA D/A Chain Limits

Table 31-14. Specifications for the D/A Chain

Parameter	Min	Typ	Max	Unit
D/A Resolution		10		bits
D/A LSB (for Vcc=2.5V)		2.4		mV
DAC DNL, over Range 0.1 to 2.2V (+/-1mA Max)	-1		+ 1.0	LSB
DAC INL, over Range 0.1 to 2.2V (+/-1mA Max)	-2		+ 2	LSB
D/A Conversion Rate		13/5		MHz
Analog Filter Bandwidth AOC_FBW=10	450	500	550	KHz
Deadzone	35	86	137	LSB's
Saturation Zone	876	941	1015	LSB's

Table 31-14. Specifications for the D/A Chain (Continued)

Parameter	Min	Typ	Max	Unit
Buffer Slew Rate	3			V/μS
Output Buffer Load: Current Peak Current DC Capacitance	 -3 -1		 3 1 200	 mA mA pF
Analog Filter Rejection AOC_FBW=10, f=500KHz f=1MHz f=2.2MHz	 -3 -10 -20			 dB dB dB
Supply current (Active)		5000		uA
Supply current (Disabled)		20		nA

The PAC D/A Chain shall be bounded by the curve shown below. On a part to part basis the output voltage at code 256'd shall deviate by no more than +/-100mV. There shall be two regions defined in the transter function of the DAC "Deadzone" and "Saturation Zone". The "Deadzone" region shall be the minimum code to produce an output voltage of 100mV. The "Saturation zone" shall be defined as the minimum code to produce an out put voltage of 2.2V. The PA D/A chain must be linear over the range 0.1 to 2.2V. The PA D/A chain can exhibit nonlinear behaviour when the output voltage is less than 0.1V and greater than 2.2V

The analogue low-pass filter is has a 2 pole order response with a 3dB cutoff frequency at 500 KHz.

The PA D/A chain forms the analog interface to the PA. As such it consists of a 10-bit DAC, a reconstruction filter along with an output buffer capable of driving large capacitive loads. A differential architecture shall be used to minimise common-mode noise effects on the desired signal.

As the PA D/A chain is guaranteed to be linear only from 0.1V to 2.2V. INL and DNL shall only be calculated over this range.The LSB size is also to be calculated in the range 0.1 to 2.2V.

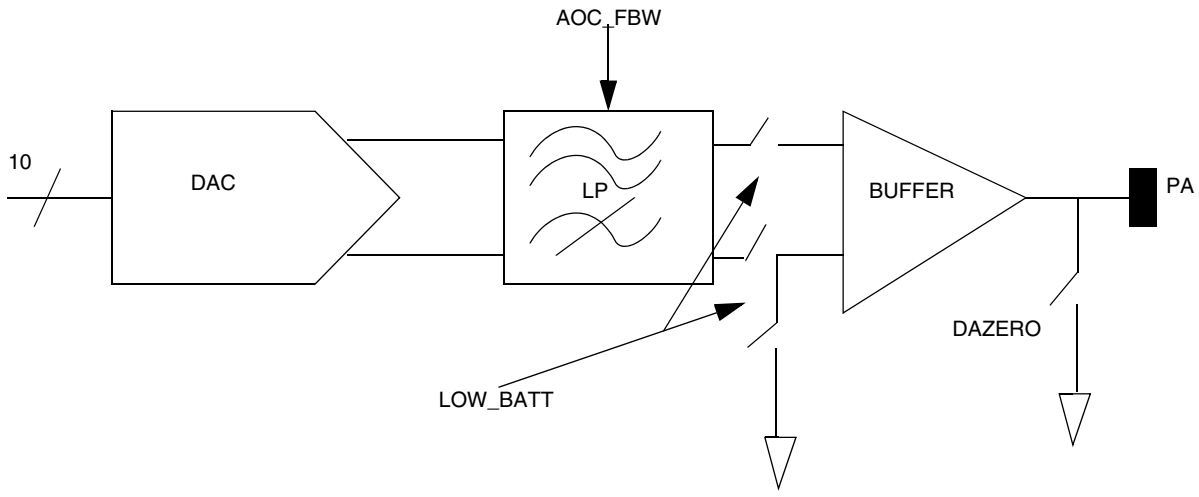


Figure 31-13. PAC DAC Chain showing DAZERO and LOW_BATT switches

31.6 Analog Test

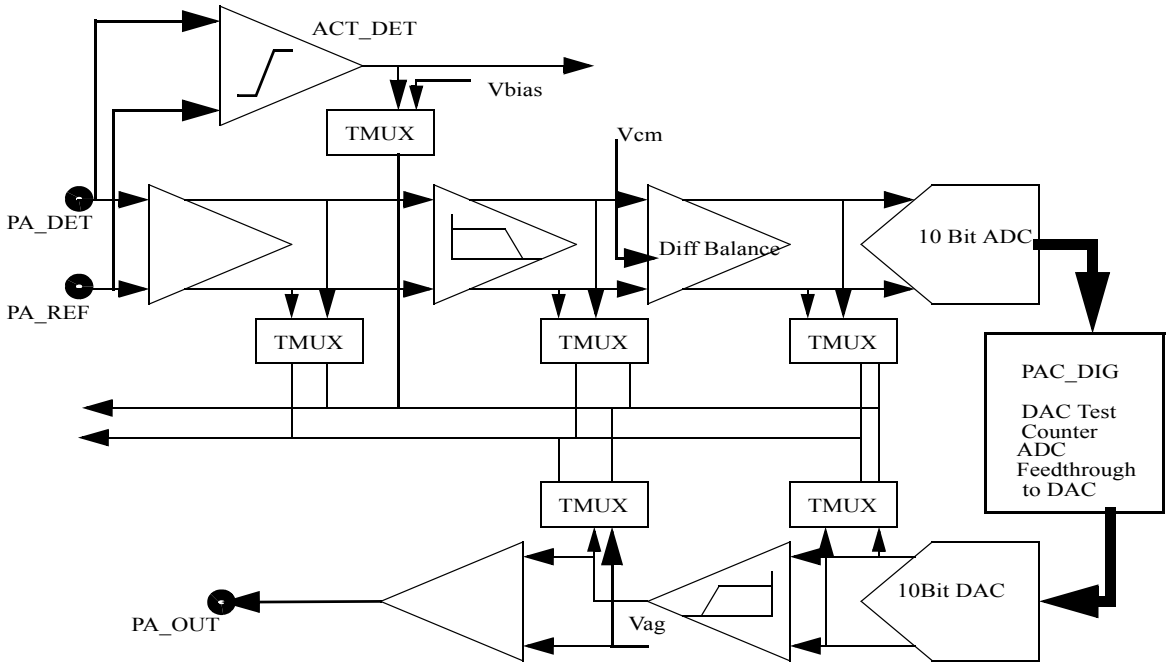


Figure 31-14. Analog Test For ADC

Table 31-15. PAC-anatest TMUX selection

#	PAC[4:0]	Test 1	Test 2	Comments
0	00000	Unused	Unused	
1	00001	Vinp Force	Vinn force	Drive ADC i/p externally
2	00010	OP3_in_p	V0p4	
3	00011	Vref_p	Vref_n	
4	00100	Vref_cmp_p	Vref_cmp_n	
5	00101	V_cmp_p	V_cmp_n	
6	00110	Vref_cm_buf	Vref_cm	
7	00111	V1p9	V0p0	
8	01000	I_rsd_I_gen1	I_rsd_I_gen2	Currents
9	01001	Out-p	Out_n	
10	01010	Cm_in_p	Vag	
11	01011	Cmp_hi_h	Cmp_lo_h	
12	01100	Input_Amp_p	Input_amp_n	
13	01101	Aaf_outp	Aaf_outn	
14	01110	Vdda	Vssa	
15	01111	Act_det	Ibias	
16	10000	Preamp Outp	Preamp Outn	I-V Outputs
17	10001	Filter Outp	Filter CMFB Signal	
18	10010	CMFB from Preamp	Vag	
19	10011	Filter Outn	Buffered LVDD	
20	10100	Gnd	Gnd	Digital Filter test. Drive DAC input with word from the ADC. Maximum filter BW setting for the ADC
21	10101	FilterInp	FilterInn	Analog Filter test. Drive DAC filter with a signal from Anatest

Table 31-15. PAC-anatest TMUX selection (Continued)

#	PAC[4:0]	Test 1	Test 2	Comments
22	10110	Gnd	Vdda	Power and Gnd for Pac Dac Output buffer
23	10111	Unused C	Unused D	
24	11000	Cmp_hi_l	Cmp_lo_l	
25	11001	Swvin_l	Swfb_l	
26	11010	Clkout_h	Swvin_and_c 1_l	
27	11011	P1_h	P1d_l	
28	11100	P2_h	P2d_l	
29	11101	Vdda_ref_gen	Vdda-2stg_op amp	
30	11110	Vssa_ref_gen	Vssa_2stg_o pamp	
31	11111	Vinp	Vinn	Measure ADC input

31.7 System Operation

The system is intended to be operational for GSM FN mode.

31.7.1 GSM FN Operation

The internal GSM mapping performed by the two port modulation is a constant envelope signal with no ramp-up due to the I and Q modulation. Therefore, the power up ramping will be controlled by a predefined envelope shaping applied to the PA bias control.

Requirements

In the GSM TDMA system, symbol times and bit times are equal and are 3.69 μs, the allowed power ramp time is 7.58 symbols, and there are 3 prescribed symbols at the start and end of each slot (0,0,0). System timing is given relative to a GSM definition which places symbol time 0 as the second local maximum/minimum after the constant slope on a modulator phase plot. The corresponding zero reference in the digital domain is defined as the beginning of the bit time at which the first tail symbol is sent. This symbol timing reference is shown graphically below:

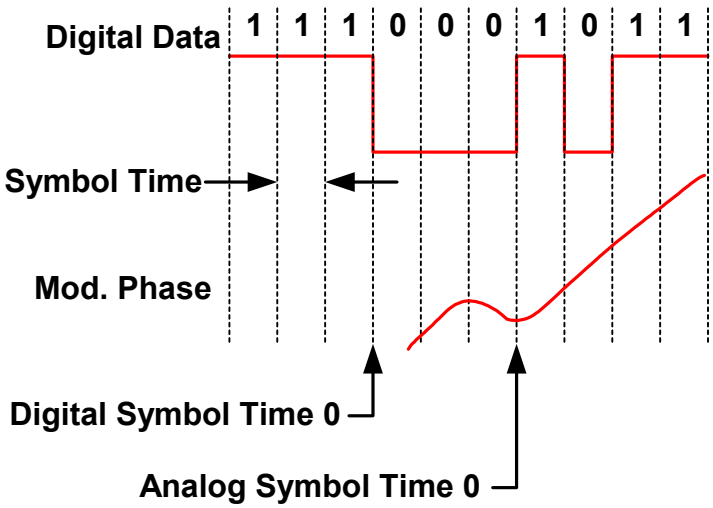


Figure 31-15. GSM Symbol - Digital /Analog Zero Reference Timing

This timing places the allowed ramp up time 28 us before zero, so the ramp up may occur from symbol time -7.583 to symbol time 0. There are additional restrictions on power levels at various times during the ramp up period which led to the timing described below. System times are given relative to the RF input signal. Any delays between the edges of DMCS (which are the timing references for the AOC system) and the symbol times of the RF input signal must be accounted for in the programming of the delays. This can only be done with the knowledge of the delay between the rising edge of DMCS and the phase appearing at the RF input.

The following discussion is for a standard burst of 148 symbols. For bursts of other lengths, the timing of DMCS falling edge would need to be adjusted according to the burst length.

The peak/average and peak/null ratios of the signal are both 0 dB (constant envelope). There is a defined time mask in the system spec, as well as a requirement for transient alternate channel power. For transient alternate channel power, the output of the transmitter must be viewed through 30 kHz filters placed on the alternate channels. The power versed time response of each filter output is measured, and the peak power is determined. The peak powers of each filter must be less than -23 dBm for radio power out levels of 37 dBm or lower.

Due to the constant envelope nature of GSM, the signal amplitude is not controlled by I/Q amplitudes, but is constant coming out of the modulator, regardless of whether data is being transmitted of not. This means that the signal coming into the AOC system is at a high level even before ramping begins. The system will have four states of operation as described below:

State 1: Estimation

For the loop precharge function the digital word GSM_INIT is applied to the accumulator input. The application of this word can be held off with the EST_DLY timer. INIT_DLY and ACC_INIT can be used if it is desirable to start the AOC control line from a point other than zero. If they are used, INIT_DLY would be set for the beginning of this activity detection period, and ACC_INIT would set the start value for the AOC control line. If the control line is to start at zero, then INIT_DLY and ACC_INIT are set to zero in this mode.

Once the EST_DLY timer completes then GSM_INIT will cause a linear rise in the D/A output causing the PA output power to rise nonlinearly, until the detected word exceeds the ACT_TH level. At this time the GSM_INIT word is removed and the loop configuration changes to closed loop mode awaiting power

ramping. For modes other than GSM the ACT_EN bit is set low to disable the function. The purpose of this step is to force the loop into a low power closed loop state so that a ramping function may be applied to the control voltage port of the PA. Ramping the loop directly from the zero starting state would fail due to poor response of the power detector at very low power levels.

The activity detector is selectable from the ACT_SEL SPI field to be a digital system, as described above, or an analog system as existed in previous GSM ICs. The analog activity detector is not programmable, but uses a comparator that reacts to any difference in voltage between the detect and detect reference inputs.

During this time interval the ramping waveform is programmed to a factory phased value which corresponds to the minimum power output of the power amplifier necessary to achieve correct closed loop operation. This is set with the SPI field OFS[7:0]. Thus when the constant input to the integrator (GSM_INIT) is terminated, the comparison of the detected RF to the ramping waveform will result in an output of almost zero so that the loop does not attempt to shift from the minimum power closed loop state.

Two symbol times are allocated to the initialization state. It is expected that the time for activity detection to complete will be variable, but for correct operation activity detection must complete within the two symbol times allocated to initialization. Once activity detection completes, the loop idles in a low power out state awaiting the start of ramping.

A plot of the timing events in the GSM initialization state is presented below.

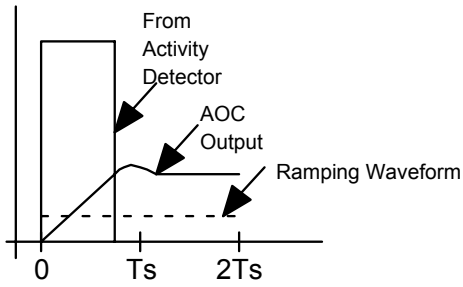


Figure 31-16. The Timing Events in the GSM Initialization State

State 2: Ramp Up

In the Ramping state, the ramping waveform is generated from a lookup table. The output values will be equal to [(Magnitude)*PWR]+OFS. Magnitude is the value of the quarter raised cosine lookup table that is normalized to a value of 1 at the end of the ramp. Two predefined ramp shapes are provided for the FN GSM AOC controller. They are the Raised Cosine and Half Raised Cosine shapes. OFS[7:0] is the offset needed to close the loop due to the detector offset, and PWR[9:0] is the desired power of the burst. The ramping state lasts 2 symbol times. The loop bandwidth is the same as in the Estimation state, and is sufficient to meet the time mask and transient ACP specs.

Note that in the GSM case the offset power and final power values are programmed into the AOC controller. Figure 31-17 shows the timing events in the GSM initialization and ramping states.

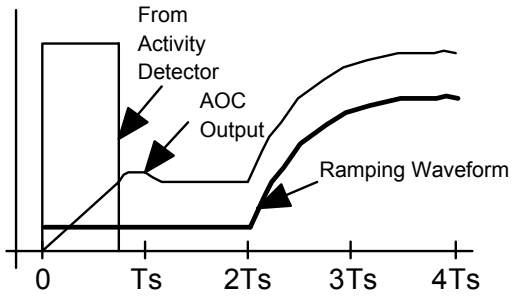


Figure 31-17. The Timing Events in the GSM Initialization and Ramping States

State 3: Modulation

In this state, the AOC reference output is nominally fixed to hold the RF amplitude constant.The AOC reference value may be modified by the Saturation Detection function described below.

Saturation detection is accomplished with a digital comparator which trips when the error signal becomes too large positive. The function of the comparator is held off by a programmable delay relative to DMCS. This delay is set by the digital word SAT_DLY[13:0]. If SAT_DET is true then the digital word SAT_SEPT[4:0] is subtracted from the PWR[9:0] word on each clock interval until SAT_DET becomes false. (If the value of the ramping waveform reaches zero, then any further decrements result in a value of zero.

State 4: Ramp Down

Ramp down is triggered by RAMP_DN_DLY expiring after the falling edge of DMCS. This will cause the look up table to output the ramp down shape. After the ramp down has completed, 1/4 bit time later the OFS value will be zeroed. At the end of the ramp down time the DN_DLY timer will expire which will clear any residual saturation offset, and force the TX and AOC control to an off state. The timing of the shut off is relative to the falling edge of DMCS, and is set by the SPI field DN_DLY[13:0]

Proper setting of the falling edge of DMCS depends on both the modulator and AOC design. The current SSI/modulator design for bitstream GSM mode will set the input o f the waveform generator to 1 on the falling edge of DMCS. This implies that the falling edge of DMCS will set the effective end of the GSM burst, as it will mark the transition to processing of dummy bits through the modulator. The modulator will continue to process these 1 inputs to the waveform generator until the DN_DLY expires.

The PAC system ramp down is triggered by the falling edge of DMCS, but the actual timing is set by the SPI variable RAMP_DN_DLY.

Phasing

No special phasing considerations beyond the setting of OFS and PWR for each desired output level.

Example Timing (assuming zero delays around the system)

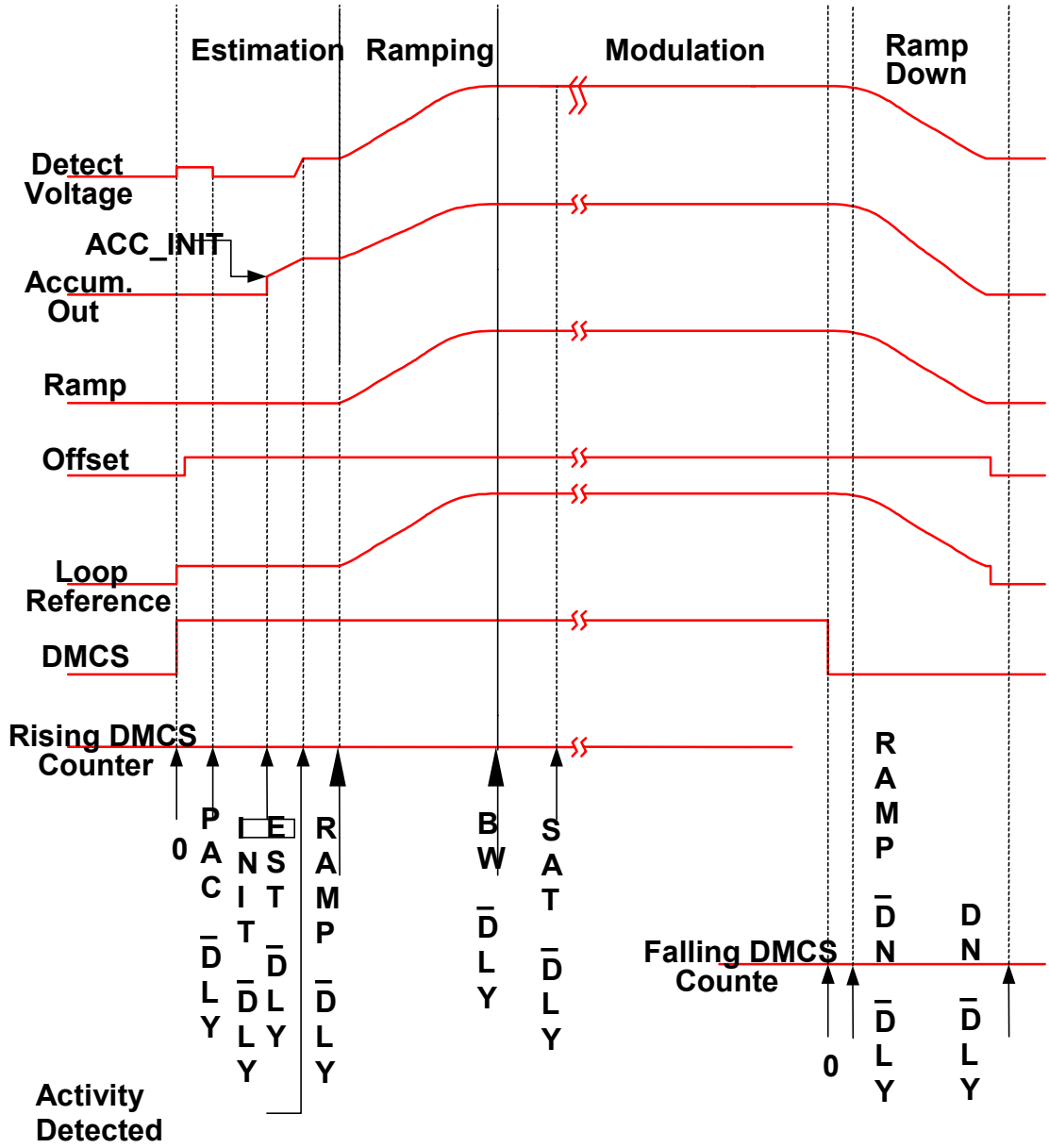


Figure 31-18. PAC Transmit Burst Timing

This example starts with the rising edge of DMCS. After a delay set by PAC_DLY the detected voltage is reduced by the residual offset. At this point there is no output from the PA. If used, a delay defined by INIT_DLY would set the point at which the accumulator and thus the AOC D/A would be forced to the initial value of ACC_INIT. Note that the control output to the PA would be smoothed by the analog filter on the AOC D/A output and would not contain the discontinuities shown for the accumulator output. At the same time, a delay defined by EST_DLY sets the point at which the accumulator begins to accumulate GSM_INIT values. This forces the AOC D/A to ramp up linearly from its start point (ACC_INIT or 0), and is the start of the activity detection period. As the AOC D/A and control line rises, the actual PA output will also rise. At some point voltage will be detected on the AOC detect line that is greater than the zero power reference voltage of the detector. This is the point of activity detection. After this point, the loop

will close and lock to the power represented by OFFSET, which is typically a low value of power, near the activity detection threshold power. The loop will idle here waiting for the next state to begin.The actual PA output will be essentially the same as the detected voltage.

After a time defined by RAMP_DLY the predefined ramp generator begins to output values at 1/4 bit time intervals. This is the start of the ramping state. As the ramp waveform rises, the radio power out follows it in a closed loop fashion. During this time the loop is closed and has a bandwidth determined by the setting of ERRGain1. The end of the ramping state corresponds to the start of the active part of the burst, whereupon the loop enters the modulation state.

In the modulation state, the loop can be set to a lower bandwidth if desired through the use of ERRGain2 and BW_DLY at the time shown above, but this is not mandatory. In earlier descriptions the bandwidth was not changed. If this is desired, then ERRGain2 may be set to the same value as ERRGain1. Shortly after the modulation state begins, SAT_DLY expires to begin the SAT detection function if it is enabled.

Finally, as the end of the burst arrives, DMCS goes low and after RAMP_DN_DLY expires ramp down starts. The offset value drops to zero 1/4 bit time after the ramp down completes, as if it were another ramp step. After a delay set by DN_DLY the TX AOC section is powered down, pulling the AOC control line to zero.

To guarantee correct AOC circuit operation, the AOC delays cannot be set to identical values. This is required from a system and design view point.

Delay Timing

Determining the proper settings for the various delays in the system is critical to its correct operation. The diagram below is a simplified block diagram of a complete system showing the various delays that may occur. It will be used in the following discussion

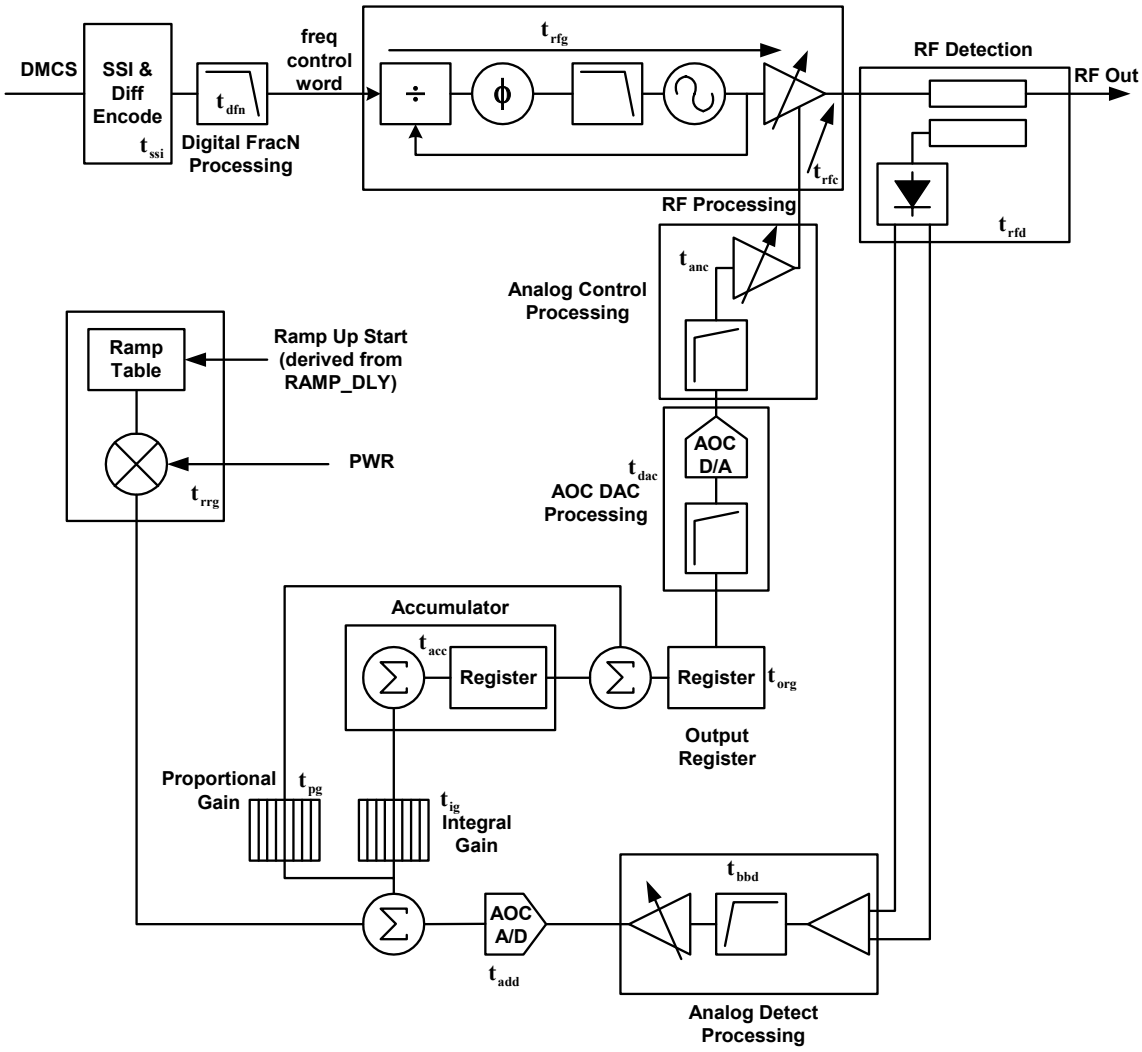


Figure 31-19. PAC Loop Including Element Timing Delays

The various delays in the diagram are summarized below. In some cases they may be zero or negligible, but they are included for completeness.

Table 31-16. PAC Loop Element Timing Delays

Sym	Delay
tssi	Delay from DMCS rising to first bit appearing at differential encoder output
tdfn	Delay through digital filtering to produce the FracN frequency control word
trfg	Baseband delay through RF processing such as synthesizer, VCO, and PA
trfc	Control line delay through power control elements such as VCA or PA
trfd	Delay through RF detection such as power coupler and RF to DC detector
tbbd	Delay through analog detect processing such as difference buffer, detect filter, and PRA

Table 31-16. PAC Loop Element Timing Delays (Continued)

Sym	Delay
tadd	Delay through AOC A/D
tig	Delay through integral gain shift register
tpg	Delay through proportional gain shift register
tacc	Delay through AOC accumulator
torg	Delay through AOC output register
tdac	Delay through AOC DAC chain such as digital LPF and DAC
tanc	Delay through analog control line processing such as AOC reconstruction filter and scaling circuits
trrg	Delay from RAMP_DLY timer expiring to first ramp values appearing at error summer

The settings of the various SPI delay parameters depend on nominal system timing as well as the delay values shown above.

For good transient spectral performance, the logic unit must send dummy 1 bits prior to the three tail bits the system spec calls for. The AOC ramp occurs during these dummy bits, so a minimum of two dummy bits must be sent to cover an AOC ramp of 2 symbols duration. There is no penalty in sending more that two bits as long as the AOC timing takes the number of dummy bits into account. The DSP design for bitstream GSM mode applies a 1 to the waveform generator input during the training sequence. This results in a series of 4 1 bits being applied to the waveform generator input prior to the first SSI data bit. That is sufficient to meet requirements, so the baseband processor may begin its transition with the first system defined tail bit.

The parameter representing the symbol time at which the logic unit sends the first serial bit is called symref.

Similarly, on the falling edge the logic unit must send dummy 1 bits following the three tail bits the system spec calls for. The AOC ramp occurs during these dummy bits, so a minimum of two dummy bits must be sent to cover an AOC ramp of 2 symbols duration.

The DSP design for bitstream GSM mode applies a 1 to the waveform generator input on the falling edge of DMCS. If the falling edge of DMCS is nominally timed to follow the transmission of the last tail bit, this results in a series of 1 bits being applied to the waveform generator input following the last tail bit. That is sufficient to meet requirements, so the baseband process need not send any dummy bits following the last tail bit.

The parameter representing the symbol time at which the logic unit sends the last serial bit is called symfall. DMCS should be set to fall once the last bit has been sent. Since the timing reference will place symfall at the start of the last symbol, but DMCS is at the end, calculations concerning the falling edge of DMCS will be offset by the SSI symbol length. In this case that offset is 1 TX_CLK period.

Also, the parameter tssi has both a fixed hardware component as well as a SPI variable software component set by the variable SSI_DLY. This is taken into account in the calculations below. Calculation of the parameters is shown in the spreadsheet below using the GSM standard reference time of symbol 0 being the second local maximum/minimum after the constant slope on a modulator phase plot. Some parameters depend on the exact radio hardware implemented with the AOC system, these have been set to zero in the spreadsheet below.

Calculated System Parameters

Parameter	Nominal System Timing (in symbols)	Delay Value	Comment
PAC_DLY	- - -	22	Assuming PAC comes up with DMCS, the delay should be 4 ADC conversion cycles + 2 source clock cycles from DMCS.
INIT_DLY	-4	184	Total delay from DMCS to the RF output minus the delay from accumulator output to RF output. (Symbol -4 reaches RF output when control line does.)
EST_DLY	-4	182	Total delay from DMCS to the RF output minus the delay from accumulator input to RF output. (Symbol -4 reaches RF output when control line does.)
BW_DLY	0	389	Total delay from DMCS to the A/D output. (Symbol 0 reaches shifter input when the BW changes.)
DIV_DLY	0	392	Total delay from DMCS to the accumulator output. (Symbol 0 reaches accumulator output when clock changes.)
PROP_DLY	0	389	Total delay from DMCS to the A/D output. (Symbol 0 reaches shifter input when the BW changes.)
RAMP_DLY	-2	293	Total delay from DMCS to the adder input minus delay from ramp start signal to adder input. (Symbol -2 from both paths reaches the error summer at the same time.)
RAMP_DN_DLY	147	101	Total delay from falling DMCS to the adder input minus delay from ramp start signal to adder input. (Symbol 147 from both paths reaches the error summer at the same time.)
DN_DLY	149.125	198	Total delay from falling DMCS to the RF output. (Symbol 149.125 reaches RF output when analog control lines pulling to zero do.)
SAT_DLY	1	437	Total delay from DMCS to the A/D output. (Symbol 1 reaches SAT detector input when SAT detection begins.)

Calculated System Parameters

Parameter	Nominal System Timing (in symbols)	Delay Value	Comment
PAC_DLY	- - -	22	Assuming PAC comes up with DMCS, the delay should be 4 ADC conversion cycles + 2 source clock cycles from DMCS.
INIT_DLY	-4	184	Total delay from DMCS to the RF output minus the delay from accumulator output to RF output. (Symbol -4 reaches RF output when control line does.)
EST_DLY	-4	182	Total delay from DMCS to the RF output minus the delay from accumulator input to RF output. (Symbol -4 reaches RF output when control line does.)
BW_DLY	0	389	Total delay from DMCS to the A/D output. (Symbol 0 reaches shifter input when the BW changes.)
DIV_DLY	0	392	Total delay from DMCS to the accumulator output. (Symbol 0 reaches accumulator output when clock changes.)
PROP_DLY	0	389	Total delay from DMCS to the A/D output. (Symbol 0 reaches shifter input when the BW changes.)
RAMP_DLY	-2	293	Total delay from DMCS to the adder input minus delay from ramp start signal to adder input. (Symbol -2 from both paths reaches the error summer at the same time.)
RAMP_DN_DLY	147	101	Total delay from falling DMCS to the adder input minus delay from ramp start signal to adder input. (Symbol 147 from both paths reaches the error summer at the same time.)
DN_DLY	149.125	198	Total delay from falling DMCS to the RF output. (Symbol 149.125 reaches RF output when analog control lines pulling to zero do.)
SAT_DLY	1	437	Total delay from DMCS to the A/D output. (Symbol 1 reaches SAT detector input when SAT detection begins.)

Calculated System Parameters

Parameter	Nominal System Timing (in symbols)	Delay Value	Comment
PAC_DLY	- - -	22	Assuming PAC comes up with DMCS, the delay should be 4 ADC conversion cycles + 2 source clock cycles from DMCS.
INIT_DLY	-4	184	Total delay from DMCS to the RF output minus the delay from accumulator output to RF output. (Symbol -4 reaches RF output when control line does.)
EST_DLY	-4	182	Total delay from DMCS to the RF output minus the delay from accumulator input to RF output. (Symbol -4 reaches RF output when control line does.)
BW_DLY	0	389	Total delay from DMCS to the A/D output. (Symbol 0 reaches shifter input when the BW changes.)
DIV_DLY	0	392	Total delay from DMCS to the accumulator output. (Symbol 0 reaches accumulator output when clock changes.)
PROP_DLY	0	389	Total delay from DMCS to the A/D output. (Symbol 0 reaches shifter input when the BW changes.)
RAMP_DLY	-2	293	Total delay from DMCS to the adder input minus delay from ramp start signal to adder input. (Symbol -2 from both paths reaches the error summer at the same time.)
RAMP_DN_DLY	147	101	Total delay from falling DMCS to the adder input minus delay from ramp start signal to adder input. (Symbol 147 from both paths reaches the error summer at the same time.)
DN_DLY	149.125	198	Total delay from falling DMCS to the RF output. (Symbol 149.125 reaches RF output when analog control lines pulling to zero do.)
SAT_DLY	1	437	Total delay from DMCS to the A/D output. (Symbol 1 reaches SAT detector input when SAT detection begins.)

Figure 31-20. PAC System Parameters

Example values for system settings using a reference clock of 13 MHz are shown in the table below (all values decimal). They are based on simulations of a particular set of RF hardware associated with the AOC loop. Some values marked with * likely will need to change for each hardware configuration, and are presented as starting points only. The delay values shown above are repeated below. Some values are power level dependent, and are marked P. Variables for functionality that is not enabled in this mode are not shown, and their programming will not matter.

Table 31-17. Reference Clock 13MHz System Settings Example Values

Parameter	Setting	Parameter	Setting	Parameter	Setting
PAC_DLY	*22	DN_DLY	*198	BACKWARD	0
CD_MODE	0	AOCG_DLY	0	ACT_SEL	1
ADI_REF	1	Proff	1	RAMP_DN_DLY	
GPA2D_DLY		AOC_MAX	1023	ACT_TH	P
SAT_TH	P	SAT_STEP	P	OFS	P
GSM_INIT	P	SAT_DLY	*437		
AOC_MANUAL	0	DAZERO	1	DET_FLT	3
SSI_DLY	*240	CD_FORCE	0		
AOC_FBW	2	ADI_REF	5		
AOC_GAIN	P	DET_HIGHV	P	a2di_soft_set	P
SAT_EN	1			ERR_Gain1	3
ERR_Gain2	*3	ACC_INIT	P	CDMA_DN	1
AOC_HOLD	0	RAMP_SEL	*0	EST_DLY	*182
INIT_DLY	*184	RAMP_DLY	*293	BW_DLY	*389
DIV_DLY	*392	ACT_EN	1	RAMP_REF	12
PWR	P			RAMP_DN_DLY	101

Chapter 32
Regulators (REGUL)

Revision History Table

Revision	Date	Author	Changes
0.0	03/25/2002	May Len	Initial Release: copied from LT rev2.4 For LTS: Remove core regulator startup overshoot and startup time spec because the core regulator is always on.
0.1	05/07/03		Updated for LTE specification release.

32.1 Linear Regulator Characteristics

All regulators are voltage “foldback” regulators having the topology as shown in Figure 32-1. The minimum current limit is defined as the output voltage not going below the specified voltage output limit with the specified current being drawn.

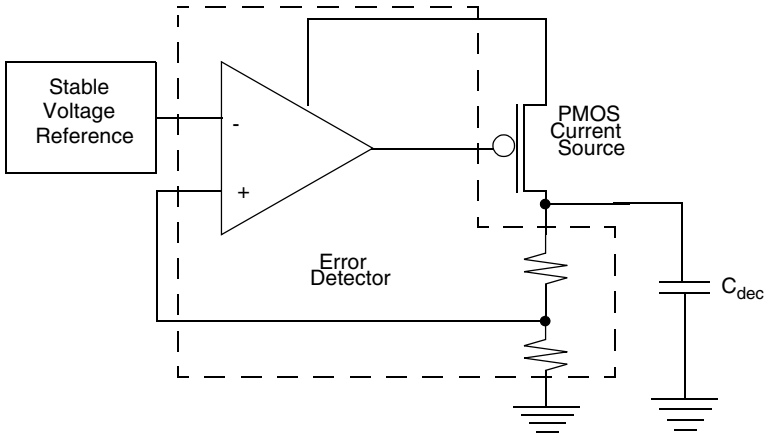


Figure 32-1. Generic Foldback Regulator Block Diagram

等效串联电阻值

All regulators will operate with low ESR ceramic type bypass capacitors. This choice is made to ensure that the impulsive current drains typically seen on digital radios will not cause excessive transient drops. In addition, the use of low ESR capacitors aids in the rejection of switching power supply noise at higher frequencies. The output voltage tables specified for each of the linear regulator sections indicate the allowed variation inclusive of all conditions, including process variation, temperature range, line regulation, and load regulation.

32.1.1 Test Signal Descriptions

Each regulator is allocated two test bits so that silicon debug can be simplified by porting out signals of interest to test ports.

Table 32-1. Core Regulator Test Bit Allocation

regul_core_test_inp	regul_core_test_inn	regul_core_test_outp	regul_core_test_outn
0	0	0	0
0	1	0	regul_core_byp (1.575 V)
1	0	0	0
1	1	regul_core_byp_sense (1.575 V)	vpgate (1.2 ~ 1.7)

Table 32-2. TXRX Regulator Test Bit Allocation

regul_txrx_test_inp	regul_txrx_test_inn	regul_txrx_test_outp	regul_txrx_test_outn
0	0	0	0
0	1	vinp (1.575 V)	ibias_mux (5 uA)
1	0	rtp_rx_vag (1.25 V)	vpgate (2.1 ~ 2.5 V)
1	1	rep_rx_ref_lo (0.85 V)	rtp_rx_ref_hi (1.65 V)

Table 32-3. Codec Regulator Test Bit Allocation

regul_codec_test_inp	regul_codec_test_inn	regul_codec_test_outp	regul_codec_test_outn
0	0	0	0
0	1	vinp (1.575 V)	regul_codec_byp (2.5 V)
1	0	rtp_rx_vag (1.25 V)	vpgate (2.1 ~ 2.5V)
1	1	rep_rx_ref_lo (0.85 V)	rtp_rx_ref_hi (1.65 V)

Table 32-4. Synthesizer Regulator Test Bit Allocation

regul_synth_test_inp	regul_synth_test_inn	regul_synth_test_outp	regul_synth_test_outn
0	0	0	0
0	1	0	regul_synth_byp (1.575 V)
1	0	0	0
1	1	regul_synth_byp_sense (1.575 V)	vpgate (2 ~ 2.4 V)

The regulator analog test mode is controlled by accessing to ANA_CNTL and ANA_DATA2 registers. The detail description of the registers is in Test Control Module (TCM) chapter. The description of analog test control and data register for regulators is listed in this spec for reference.

ANA_CNTL Control Register **Addr $2484_4010**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
		ANA_EN1	ANA_EN0								tcm_TST_DC[5:4]		tcm_TST_DC[3:0]			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 32-5. Analog Control Mux1 Description for Regulators

TST_DC3	TST_DC2	TST_DC1	TST_DC0	Test_output select
1	0	0	0	REGUL_RXTX
1	0	1	0	REGUL_VOC
1	0	1	1	REGUL_SYN
1	1	0	0	REGUL_CORE

Table 32-6. Analog Control Mux2 Descriptions

TST_DC5	TST_DC4	Output_type select
0	0	Unbuffered
0	1	Digital Buffer
1	0	Analog Buffer
1	1	LVDD

ANA_DATA2

Data Register2

Addr $2484_4018

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
					REG[7:0]								GPADC[3:0]			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 32-7. Analog Data Register 2 Description for Regulators

Name	Description	Settings
REG[7] Bits 11	**regul_core_test_in_p -** input	0 = disable 1 = asserted
REG[6] Bits 10	**regul_core_test_in_n -** input	0 = disable 1 = asserted
REG[5] Bits 9	**regul_codec_test_in_p -** input	0 = disable 1 = asserted
REG[4] Bits 8	**regul_codec_test_in_n -** input	0 = disable 1 = asserted
REG[3] Bits 7	**regul_synth_test_in_p -** input	0 = disable 1 = asserted
REG[2] Bits 6	**regul_synth_test_in_n -** input	0 = disable 1 = asserted
REG[1] Bits 5	**regul_txrx_test_in_p -** input	0 = disable 1 = asserted
REG[0] Bits 4	**regul_txrx_test_in_n -** input	0 = disable 1 = asserted

32.1.2 1.575 V Regulator-regul_core

This regulator provides power to the digital cores in the Neptune IC. It receives a 1.875 V (+/-4 tolerance) input and generates a regulated 1.575 V (+/-5% tolerance) output. An external bypass capacitor of 4.7uF is required. This regulator should always be powered on during operation. Its enable bit will therefore have to be hardwired to JVDD in order to ascertain that this is always the case. Table 32-8 shows characteristics of regul_core regulator. The core regulator will use two power supplies, JVDD (2.775V nom) and QVDD(1.875V nom) for its circuitry. The amplifier and feedback circuit will use JVDD as its power supply while that of the PMOS pass device will be connected to QVDD.

Table 32-8. regul_core - Neptune 1.575V Regulator Specs

Parameter	Condition	Min	Typ	Max	Units
Output Voltage	1.8 V < qvdd< 1.95 V 2.67 V < jvdd < 2.88 V 100 μA < IL < 200 mA	1.496	1.575	1.654	V
Load Regulation	100 μA < IL < 200 mA			50	mV_{pp}
Line Regulation	IL = 100 μA, 1.8 V < qvdd< 1.95 V			10	mV
PSRR	C = 4.7 μF IL = 100mA, 20-104MHz		40		dB
Transient Response	IL = 100 μA to 200 mA C = 4.7 μF		1		%
Bypass Cap ESR		> 0		0.5	Ω
Bypass Cap			4.7		uF

Table 32-8. regul_core - Neptune 1.575V Regulator Specs (Continued)

Parameter	Condition	Min	Typ	Max	Units
$I_{quiescent}$	IL = 0 mA			100	uA

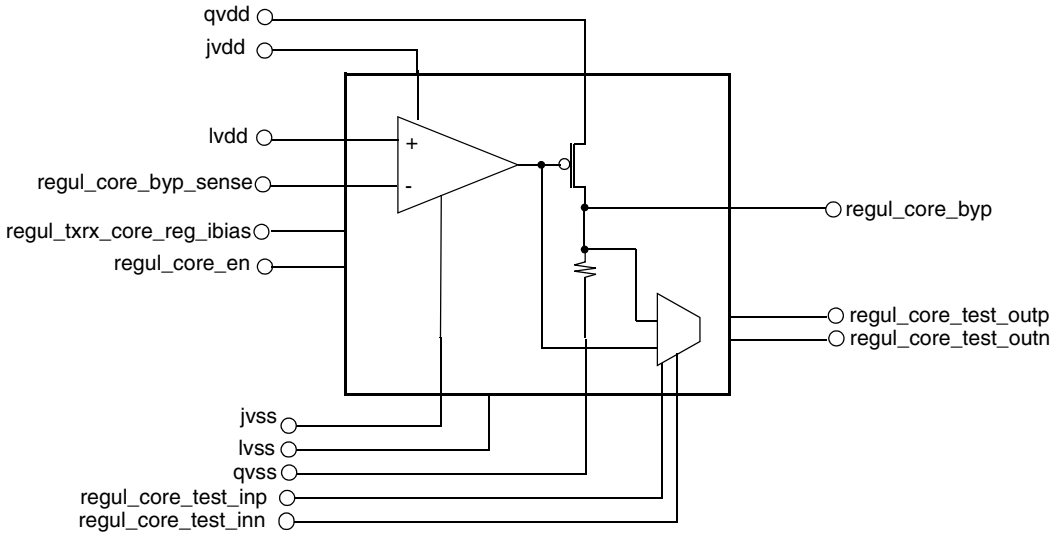


Figure 32-2. Block Diagram of Core Regulator Showing Signal Pin

Table 32-9. Core Regulator Signal Descriptions

Pin Name	Direction	Description
qvdd	In	1.875V supply voltage for PMOS pass device
qvss	In	gnd pair for qvdd
jvdd	In	2.775V supply voltage for regulator amplifier
jvss	In	gnd pair for jvdd
lvdd	In	1.575V regulator reference voltage[1]
lvss	In	gnd pair for lvdd
regul_txrx_core_reg_ibias	In	regulator 1μA bias current
regul_core_en	In	regulator enable signal, hardwired to jvdd at top
regul_core_test_inp	In	regulator test bit
regul_core_test_inn	In	regulator test bit
regul_core_byp_sense	In	1.575V regulated voltage sensed at I/O pad
regul_core_byp	Out	1.575V regulated digital supply

Table 32-9. Core Regulator Signal Descriptions

Pin Name	Direction	Description
regul_core_test_outp	Out	test bit output to anatest
regul_ore_test_outn	Out	test bit output to anatest

1. 1.575V band gap voltage will be used as reference

32.1.3 PSRR Improvement Regulators

Two PSRR improvement regulators are designed to generate the regulated supply for the RX, TX, and PAC blocks and for the Audio Codec block. This is done to improve the supply rejection characteristics of the circuits. These power supply rejection ratio improvement regulators need to be able to power down independent of receiver, transmitter, or codec enable function. The regulators receive a 2.775 V (+/-3.6% tolerance) input and generate a 2.5 V (+/-5% tolerance) regulated output. The following tables and block diagrams detail the specifications of each PSRR improvement regulator and the signal names for each.

Table 32-10. regul_txrx - Neptune TX,RX, and PAC 2.5V Regulator

Parameter	Condition	Min	Typ	Max	Units
Output Voltage	2.67V < vdd_txrx< 2.88 V, 50 μA < IL < 30 mA	2.375	2.5	2.625	V
Load Regulation	50 μA < IL < 30 mA			20	mV_{pp}
Line Regulation	IL = 50 μA, 2.67V < vdd_txrx< 2.88 V			10	mV
PSRR	C = 1 μF IL = 15mA, 20-104MHz		40		dB
Startup Overshoot	IL = 0 mA C = 1 μF			0.20	V
Transient Response	IL = 50 μA to 30 mA C = 1 μF		1		%
Turn-On Time	a2di_rt_reg_en to 90% of regul_txrx_byp (IL=30 mA)			1	ms
Bypass Cap ESR		>0		0.5	Ω
Bypass Cap			1		uF
$I_{quiescent}$	IL = 0			50	uA
$I_{leakage}$	a2di_rt_reg_en = 0		1		uA

As the following block diagram shows, the master bias block (section 31.1.5) is embedded with the regul_txrx. Its signal pin descriptions are described in section 31.1.5 of the spec.

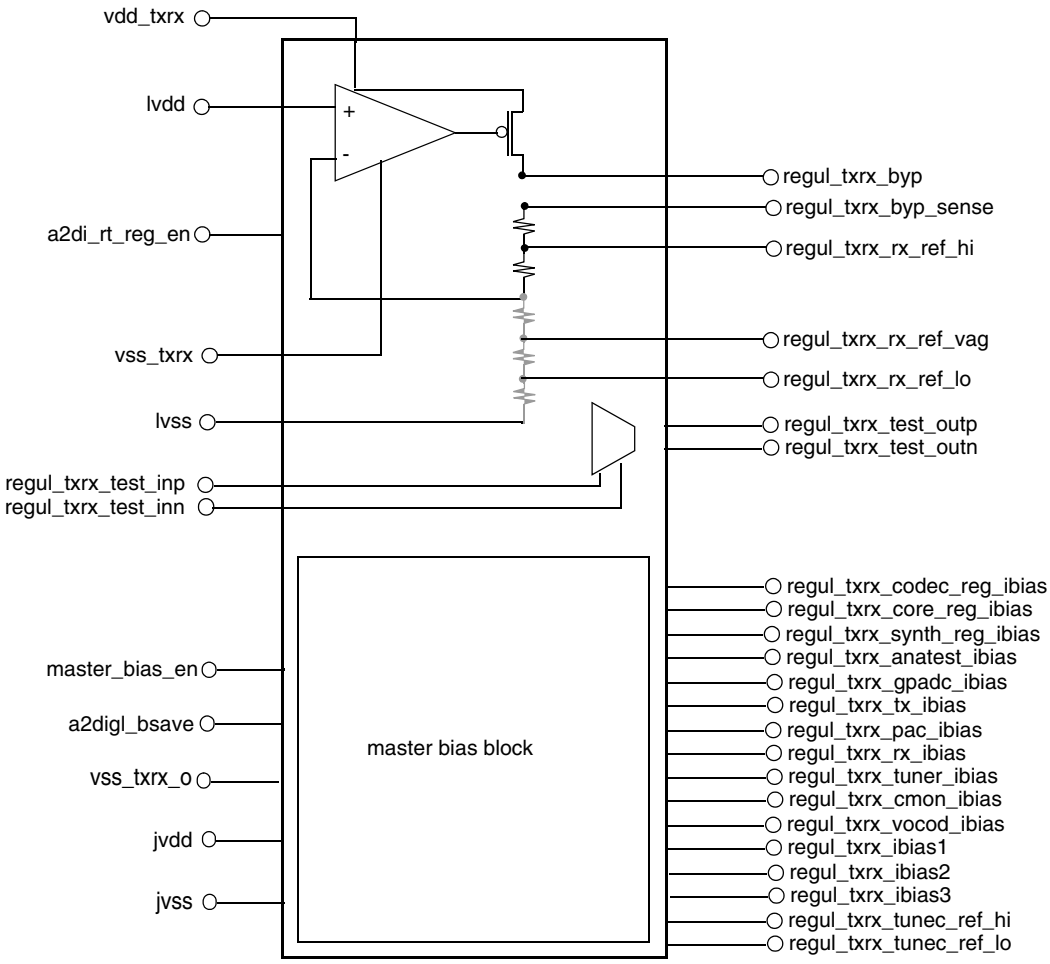


Figure 32-3. Block Diagram of TX,RX,PAC Regulator Showing Signal Pins

Table 32-11. TX,RX, PAC Regulator Signal Descriptions

Signal Name		Description
vdd_txrx	In	2.775V supply voltage for regulator amplifier
vss_txrx	In	gnd pair for vdd_txrx
vss_txrx_o	In	gnd for nwell resistor substrate
lvdd	In	1.575V regulator reference voltage[1]
lvss	In	gnd pair for lvdd
a2di_rt_reg_en	In	regulator enable signal
regul_txrx_test_inp	In	regulator test bit
regul_txrx_test_inn	In	regulator test bit
regul_txrx_byp_sense	In	2.5V regulated voltage sensed at I/O pad

Table 32-11. TX,RX, PAC Regulator Signal Descriptions

Signal Name		Description
regul_txrx_byp	Out	2.5V regulated supply
regul_txrx_test_outp	Out	test bit output to anatest
regul_ore_test_outn	Out	test bit output to anatest
regul_txrx_rx_ref_hi	Out	1.65V reference voltage for RX path
regul_txrx_rx_ref_vag	Out	1.25V reference voltage for RX path
regul_txrx_rx_ref_lo	Out	0.85V reference voltage for RX path

1. 1.575V band gap voltage will be used as reference

Table 32-12. regul_codec - Neptune Audio Codec 2.5V Regulator

Parameter	Condition	Min	Typ	Max	Units
Output Voltage	2.67 V < vdd_codec< 2.88V, 50 μA < IL < 20 mA	2.375	2.5	2.625	V
Load Regulation	50 μA < IL < 20 mA			20	mV_{pp}
Line Regulation	IL = 50 μA, 2.67 V < vdd_codec< 2.88 V			10	mV
PSRR	C = 1 μF IL = 10 mA, 20-104MHz		40		dB
Startup Overshoot	IL = 0 mA C = 1 μF			0.20	V
Transient Response	IL = 50 μA to 20 mA, C = 1μF		1		%
Turn-On Time	a2di_aud_reg_en to 90% of regul_codec_byp (IL=20mA)			1	ms
Bypass Cap ESR		> 0		0.5	Ω
Bypass Cap			1		uF
$I_{quiescent}$	IL = 0			50	uA
$I_{leakage}$	a2di_aud_reg_en = 0		1		uA

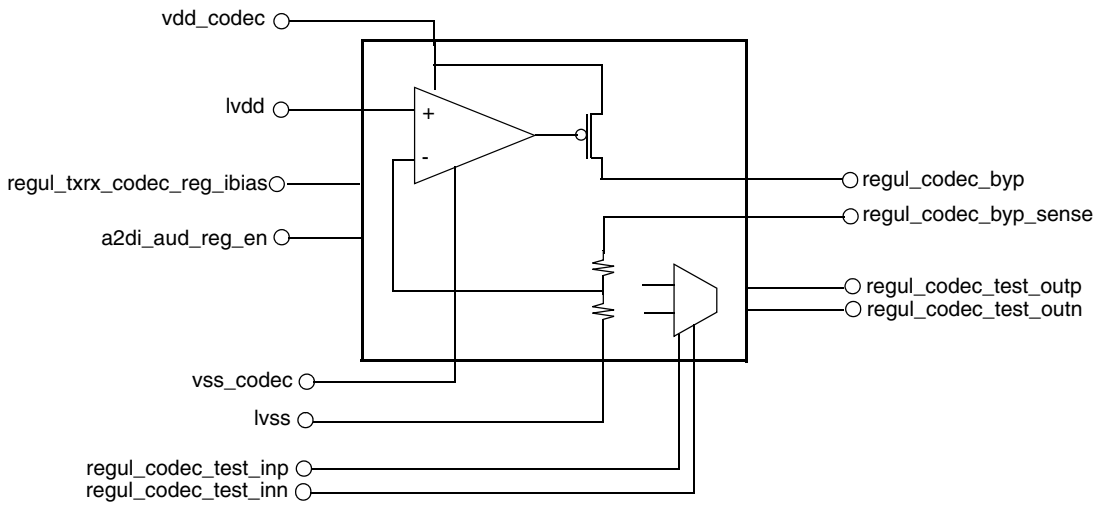


Figure 32-4. Block Diagram of Codec Regulator Showing Signal Pins

Table 32-13. Codec Regulator Signal Descriptions

Signal Name		Description
vdd_codec	In	2.775V supply voltage for regulator amplifier
vss_codec	In	gnd pair for vdd_codec
lvdd	In	1.575V regulator reference voltage[1]
lvss	In	gnd pair for lvdd
regul_txrx_codec_reg_ibias	In	regulator 1μA bias current
a2di_aud_reg_en	In	regulator enable signal
regul_codec_test_inp	In	regulator test bit
regul_codec_test_inn	In	regulator test bit
regul_codec_byp_sense	In	2.5V regulated voltage sensed at I/O pad
regul_codec_byp	Out	2.5 V regulated supply
regul_codec_test_outp	Out	test bit output to anatest
regul_codec_test_outn	Out	test bit output to anatest

1. 1.575V band gap voltage will be used as reference

32.1.4 1.575 V Synthesizer Regulator

The digital part of the Tx and Rx synthesizers uses the same power supply from a local regulator. This helps the prescalers in the synthesizers to receive 1 GHz incoming signals. The synthesizer regulator needs to be able to power down independent of either the receiver or transmitter enable function. The regulator receives a 2.775 V (+/-3.6% tolerance) input and generates a 1.575 V (+/-5% tolerance) output. The Table 32-14 shows characteristics of the regulator.

Table 32-14. Triton Synthesizer Regulator Specs

Parameter	Condition	Min	Typ	Max	Units
Output Voltage	2.67 V <vdd_txrx< 2.88 V, 50 μA < IL < 20 mA	1.496	1.575	1.654	V
Load Regulation	50 μA < IL < 20 mA			20	mV
Line Regulation	IL = 50 μA, 2.67V < vdd_txrx< 2.88 V			10	mV
PSRR	C = 1 μF, IL = 10 mA, 20-104MHz		40		dB
Startup Overshoot	IL = 0 mA			.075	V
Transient Response	IL = 50 μA to 20 mA C = 1 μF		1		%
Turn-On Time	a2di_s_reg_en to 90% of regul_synth_byp, (IL = 20 mA)			1	ms
Bypass Cap ESR		> 0		0.5	Ω
Bypass Cap			1		uF
$I_{quiescent}$	IL = 0 mA			50	uA
$I_{leakage}$	a2di_s_reg_en = 0		1		uA

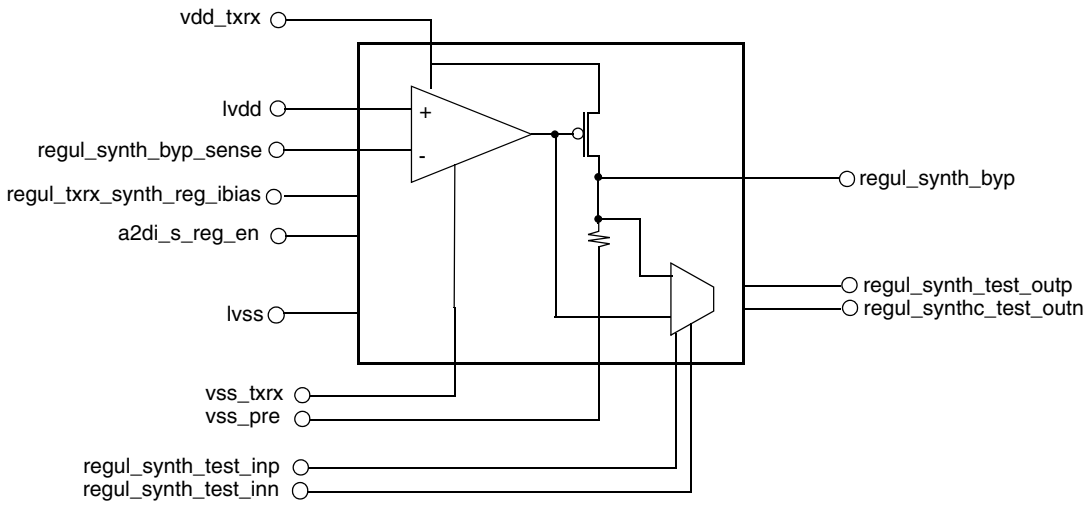


Figure 32-5. Block Diagram of Synthesizer Regulator Showing Signal Pins

Table 32-15. Synth Regulator Signal Descriptions

Signal Name		Description
vdd_txrx	In	2.775V supply voltage for regulator amplifier
vss_txrx	In	gnd pair for vdd_txrx
vss_pre	In	gnd pair for regul_synth_byp

Table 32-15. Synth Regulator Signal Descriptions (Continued)

Signal Name		Description
lvdd	In	1.575V regulator reference voltage[1]
lvss	In	gnd pair for lvdd
regul_txrx_synth_reg_ibias	In	regulator 1μA bias current
a2di_s_reg_en	In	regulator enable signal
regul_synth_test_inp	In	regulator test bit
regul_synth_test_inn	In	regulator test bit
regul_synth_byp_sense	In	1.575V regulated voltage sensed at I/O pad
regul_synth_byp	Out	1.575V regulated digital supply
regul_synth_test_outp	Out	test bit output to anatest
regul_synth_test_outn	Out	test bit output to anatest

1. 1.575V band gap voltage will be used as reference

32.1.5 Master Bias Generator

To generate the necessary bias currents for analog blocks on Neptune, a master bias block will be included within the TXRX regulator module. The master bias block will provide 1 uA (except for PAC, codec, and TX module), P-type current sources for Neptune with a +/-30% bias current tolerance. The enable pin to the master bias block will be hardwired to JVDD such that it is always on.

Table 32-16. Master Bias Signal Descriptions

Signal Name	Type	Description
jvdd	In	2.775 V bias block power supply
jvss	In	ground pair for jvdd
regul_txrx_rx_ibias	Out	1uA bias for RX path circuitry
regul_txrx_tx_ibias	Out	5uA bias for TX path circuitry
regul_txrx_pac_ibias	Out	10uA bias for PAC circuitry
regul_txrx_core_reg_ibias	Out	1uA bias for regul_core
regul_txrx_txrxp_reg_ibias	Out	1uA bias for regul_txrx
regul_txrx_synth_reg_ibias	Out	1uA bias for regul_synth
regul_txrx_codec_reg_ibias	Out	1uA bias for regul_codec
regul_txrx_anatest_ibias	Out	1uA bias for analog test block
regul_txrx_gpadc_ibias	Out	1uA bias for gpadc block
regul_txrx_tuner_ibias	Out	1uA bias for tuner block

Table 32-16. Master Bias Signal Descriptions (Continued)

Signal Name	Type	Description
regul_txrx_cmon_ibias	Out	1uA bias for cmon block
regul_txrx_vocod_ibias	Out	5uA bias for vocod block
regul_txrx_ibias1	Out	1uA bias line
regul_txrx_ibias2	Out	1uA bias line
regul_txrx_ibias3	Out	1uA bias line
regul_txrx_tunec_ref_hi	Out	1.575 V reference for tuner comparator
regul_txrx_tunec_ref_lo	Out	0.8 V reference for tuner amplifier
master_bias_en	In	master bias enable, hardwired to jvdd
a2digl_bsave	In	tuner block enable bit, used to enable reference voltages

Chapter 33
Tuning Circuit (TUNEC)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	05/07/03		Updated for LTE specification release.

33.1 Introduction

Neptune will be in the base line HiP7 digital process, with no precision resistor or capacitor. The RC constant tuning circuit (TUNEC) is to ensure the precision of all integrated filters for accurate cut off frequencies on chip. The tuning scheme will require a precision clock and trying to calibrate the RC time constant within a given period by the precision clock. Tuning scheme is designed such that it reduces the cutoff frequency variations to less than +/-10%.

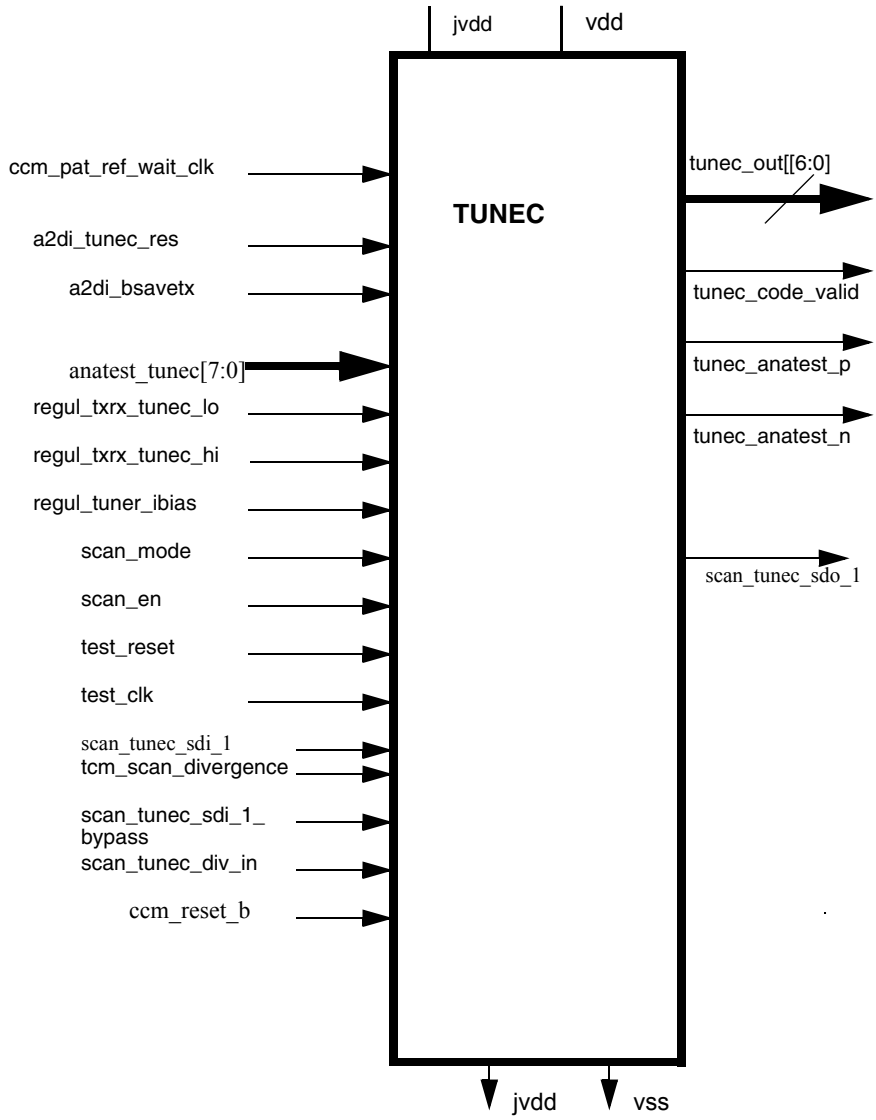


Figure 33-1. TUNEC Black Box Diagram

33.2 TUNEC Pin Description

There are twenty two pins associated with the RC tuning circuit. The block is powered by jvdd and jvss. The typical supply voltage is 2.775 V. It has provisions for test as will be described in the following sections.

Table 33-1. Pin Electrical Characteristics

Specification	Min	Typ	Max
Power Supply voltage (jvdd)	2.68 V	2.775 V	2.875 V
Digital Power supply (vdd)	1.5V	1.575V	1.65V
ccm_pat_ref_wait_clk	13 MHz		

Table 33-1. Pin Electrical Characteristics (Continued)

Specification	Min	Typ	Max
a2di_bsavetx (Connected to BSAVE)		Power down when high	
ref_tuner_ibias	0.3 µA (Power down mode)	1 µA (Normal operation)	
regul_txrx_tunec_ref_lo		0.8V	
regul_txrx_tunec_ref_hi		1.58V	
ccm_reset_b	ACTIVE LOW RESET		

33.2.1 TUNEC Module Pin List

Table 33-2 is a list of all the pins in the TUNEC module.

Table 33-2. TUNEC Module Pin List

Pin Name	Direction	Description
Input Pins		
anatest_tunec_x_[7:0]	input	RC tuning block (TUNEC) test mode select
a2di_bsavetx	input	RC tuning block power down (power save mode) signal
ccm_pat_ref_wait_clk	input	RC Tuning block clock input
a2di_tunec_res	input	RC Tuning block Reset signal
regul_txrx_tuner_ibias	input	RC tuning block bias input
regul_txrx_tunec_ref_hi	input	RC tuning block voltage reference (1.58 V)
jvdd	power input	RC tuning block power 2.775 V power supply
jvss	power input	RC tuning block 0 V analog ground
vdd	power input	digital power supply (1.57v)
vss	power input	digital ground (0V)
ccm_reset_b	input	from ccm to reset tuner
regul_txrx_tunec_ref_lo	input	from regul_txrx, reference voltage
output pins		
tunec_code_valid	output	RC tuning block valid code indicator
tunec_out_x_[6:0]	output	7 bit RC tuning block code output

Table 33-2. TUNEC Module Pin List (Continued)

Pin Name	Direction	Description
test pins		
anatest_tunec_x_[7:0]	input	RC tuning block test enable & test pins
tunec_anatest_p	output	RC tuning block -comparator probe pin
tunec_anatest_n	output	RC tuning block test pin
anatest_tunec_x_[6:0]	inputs	RC tuning block programmable test pins
scan pins		
scan_mode	input	Scan mode
scan_en	input	Scan enable
test_reset	input	RC tuning block scan test reset
test_clk	input	RC tuning block scan test clock
scan_tunec_sdi_1	input	RC tuning block scan data in
scan_tunec_sdo_1	output	RC tuning block scan data out
tcm_scan_divergence	input	Scan Divergence
scan_tunec_sdi_1_bypass	input	Scan Bypass
scan_tunec_div_in	input	Scan divergence data

33.2.2 Input Pins

As shown in Fig. 45-1, the input pins to the block comprise of the following signals.

a2di_bsavetx: This signal when active high puts the block in the power-down mode. This comes from the battery save pin (BSAVE).

ccm_pat_ref_wait_clk: System clock of 13 MHz is provided to this input. It is divided down by N, by a divider inside the block.

a2di_tunec_res: Reset pin for the tuning circuit, asserted high for reset to ensure proper operation.

reg_tuner_ibias: NMOS bias current inputs for the tuning circuit sub-blocks (1 µA).

reg_txrx-tunec_ref_hi: Input reference voltage for the comparator (1.575V) This comes from the regulator block.

regul_txrx_tunec_ref_lo: Input reference voltage for the integrator from the regulator, 0.8V.

anatest_tunec[6]: In the normal functional mode when this pin is enabled, the user will be able to observe the anatest_tunec_p, anatest_tunec_n signals.

Table 33-3. TUNEC Input Signal Description

Pin Name	Direction	Description
anatest_tunec[6]	input	RC tuning block (TUNEC) test mode select
a2di_bsavetx	input	RC tuning block power down (power save mode) signal
ccm_pat_ref_wait_clk	input	RC Tuning block clock input
a2di_tunec_res	input	RC Tuning block Reset signal
reg_tuner_ibias	input	RC tuning block bias input
regul_txrx_tunec_ref_hi	input	RC tuning block voltage reference (1.58 V)
jvdd	power input	RC tuning block power 2.775 V power supply
jvss	power input	RC tuning block 0 V analog ground

33.2.3 Output Pins

tunec_out [6:0] is the 7 bit output from the TUNEC block. Depending on the RC time constant they give an bit output which is less than +/-10% within the cutoff frequency of the filter.

tunec_code_valid bit is asserted when the TUNEC is ready with the output code.

Table 33-4. TUNEC Output Pin Description

Pin Name	Direction	Description
tunec_code_valid	output	RC tuning block valid code indicator
tunec_out[6:0]	output	7 bit RC tuning block code output

33.2.4 Test Pins

Table 33-5. TUNEC Test Pin Description

Pin Name	Direction	Description
anatest_tunec[7]	input	RC tuning block test enable (BYPASS mode)
anatest_tunec[6]	input	RC tuning block test enable in normal functional mode for anatest pins[1]
tunec_anatest_p	output	RC tuning block -comparator probe pin
tunec_anatest_n	output	RC tuning block test pin
anatest_tunec[6:0]	inputs	RC tuning block programmable test pins

anatest_tunec[7]: Bit'7 sets the TUNEC in BYPASS mode which enable urser to bypass tunec module and force a code onto the register. In normal functional mode this bit is set to 0.

DO NOT USE anatest_TUNEC[7] for normal functional tunec analog test.

anatest_tunec_n, anatest_tunec_p: To view the output of the comparator in the test mode and integrator in the test mode or functional mode.

anatest_tunec[6:0]: Are the 7 user programmable test inputs (corresponding to the 7 tunec_out pins). When TUNEC is selected in the test mode by setting anatest_tunec[7] high, anatest_tunec the these inputs can be used to force a desired code. The circuit is in the free running mode when in the test mode.

33.2.5 Scan Pins

Table 33-6. TUNEC SCAN PINS

Pin Name	Direction	Description
scan_mode	input	Scan mode
scan_en	input	Scan enable
test_reset	input	RC tuning block scan test reset
test_clk	input	RC tuning block scan test clock
scan_tunec_sdi_1	input	RC tuning block scan data in
scan_tunec_sdo_1	output	RC tuning block scan data out
tcm_scan_divergence	input	Scan Divergence
scan_tunec_sdi_1_bypass	input	Scan Bypass
scan_tunec_div_in	input	Scan divergence data

SCAN_MODE: In functional mode this signal is low.

SCAN_EN: Scan enable signal enables the block in the scan mode. In functional mode this signal is low.

TEST_RESET: Only used in scan mode, functional mode this signal is low.

TEST_CLK: Scan test clock. Only used in scan test mode.

SCAN_TUNEC_SDI_1: Scan data input to the module.

SCAN_TUNEC_SDO_1: Scan data output.

Note: Please refer to Neptune Scan methodology document for a detailed description of the scan pins.

33.3 Functional Description

The TUNEC block is intended to compensate for the variation of R or C or both, due to process or temperature changes. It does so by comparing the RC value to the reference voltages inside the TUNEC circuit, Vref2 (regul_txrx_tunec_ref_hi) tied to a tuner comparator with Vref1(regul_txrx_tunec_ref_lo), from an integrator inside the TUNEC block. By means of continuous comparisons, capacitors are switched out or in depending on the outcome of the comparison. Fig 32-2 gives the functional block diagram of the TUNEC.

When the comparator output is 1, then the C value in too low, therefore the Capacitor/bit under comparison is retained and the next capacitor is switched in. If the comparator output is 0, the Capacitor/bit under comparison is switched out and the next significant bit capacitor/bit is switched in. This is done for 7 clock cycles. The value of tune code at the end of comparison is stored in a register and output to the a2digl register at the 8th clock cycle, once the tunec_code_valid goes high indicating the availability of a valid code. The unit capacitor value is 0.2pf. The unit resistor value is 700 ohms, with a width of 1.6u for the unit resistor.

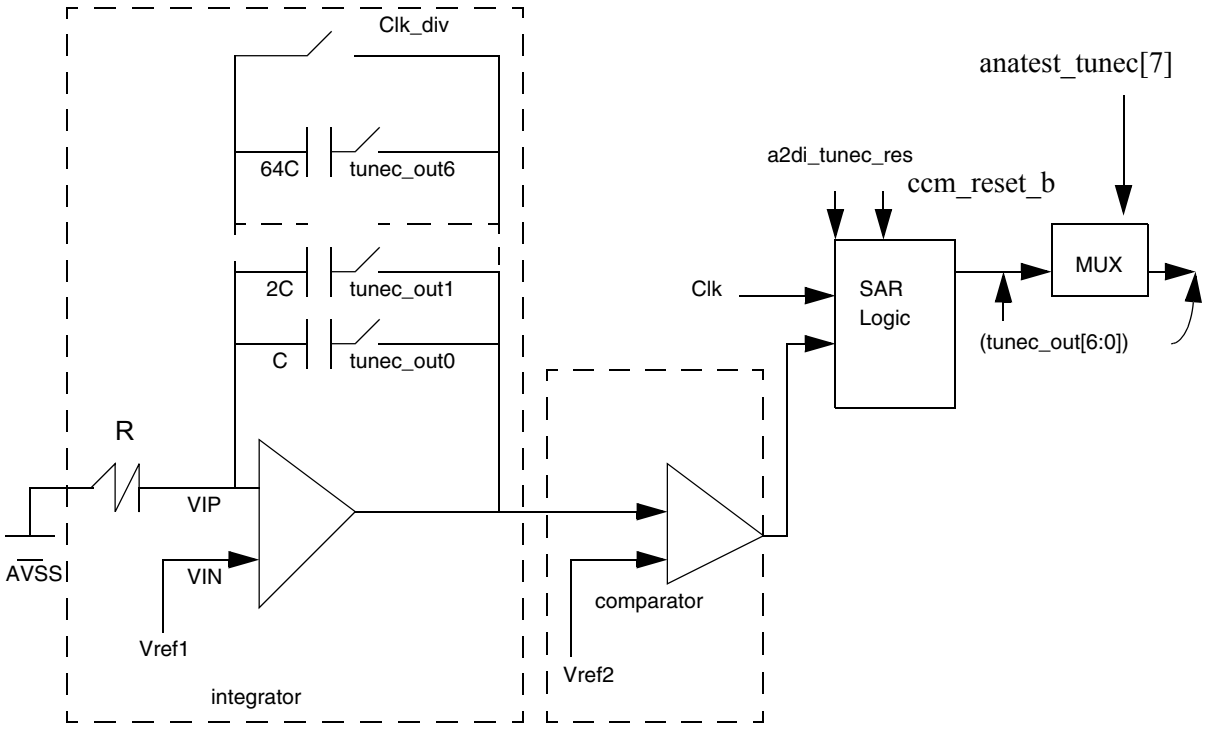


Figure 33-2. Functional Block Diagram

The comparison is done continuously for 7 cycles, corresponding to the 7 bit resolution of the TUNEC, from the largest capacitor down to the smallest. The TUNEC block when selected outputs the control code (tunec_out[6:0]) to the filter. The TUNEC circuit through its configuration outputs a code that is <+/-10% of the resistor, capacitor value.

The tuning block is selected by the a2di_tunec_res. a2di_tunec_res ensures the initial condition of the block. as shown in the timing diagrams in Figure 33-4 on page 33-9. It is also reset by the system reset signal ccm_reset_b. The power down signal is a2di_bsavetx.

The TUNEC block operates at a clock frequency of 1/N of the system clock (N = 16). An internal divider is used to divide the tunec clock of 13 MHz for the internal operation of the block. Once the block is selected and the block is brought out of reset, the tunec_code_valid goes low to disable any signals from the block. The tunec_code_valid goes high once the code becomes available after 7 clock cycles and stays high as long as the code is available and the block is not reset. The tunec_out[6:0] is then transmitted out to the appropriate locations as shown in the timing diagrams in Figure 33-4 on page 33-9. The tunec block is reset by a2digl_tunec_res or by the system reset signal reset_b. It is disabled during the power save mode.

If Vref1 and Vref2 are the inputs to the integrator and comparator respectively, and if fclk is the frequency of operation, then for an integrator, the ramp up time dt is expressed in Equation 33-1.

Eqn. 33-1

$$\left(dt = (RC)\frac{(Vref2 - Vref1)}{Vref1}\right)$$

If N =16, then the operating frequency of the block is expressed;

13 Mhz/16 = 812.5 kHz

dt = ramping time of the integrator as expressed in Equation 33-2.

$$\left(\left(\frac{1}{812.5KHz}\right) \div 2\right) = 0.6153us$$ **Eqn. 33-2**

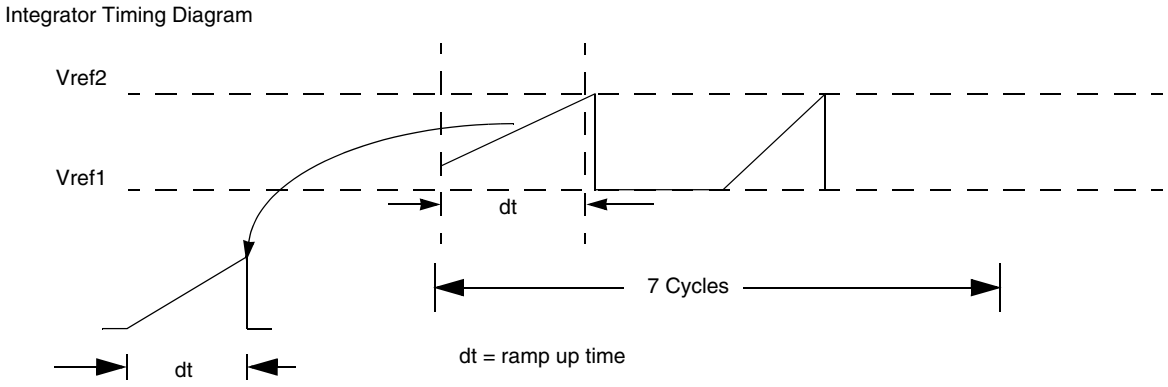


Figure 33-3. Integrator Timing Diagram

Therefore, by assuring accurate Vref2, Vref1 and a precision clock, RC at a particular temperature produces a unique code, which is made use of by the filter that requires precision tuning. When the test mode is enabled by anatest_tunec[7] is set to 1, then the test bits anatest_tunec[6:0] can be programmed by the user. However since during analog test mode the a2digl registers are not readable by the user, the forced code cannot be read out from the a2digl register. The comparator output can be observed through the anatest pin, tunec_anatest_p for 7 consecutive clock cycles (Please refer to tunec test document) and hence this code can be viewed.

The tunec block and the tune code can be completely bypassed through the a2digl block (Please refer to Chapter 36, “Mixed-Signal Control Interface (A2DIGL).”

33.4 Timing Diagrams

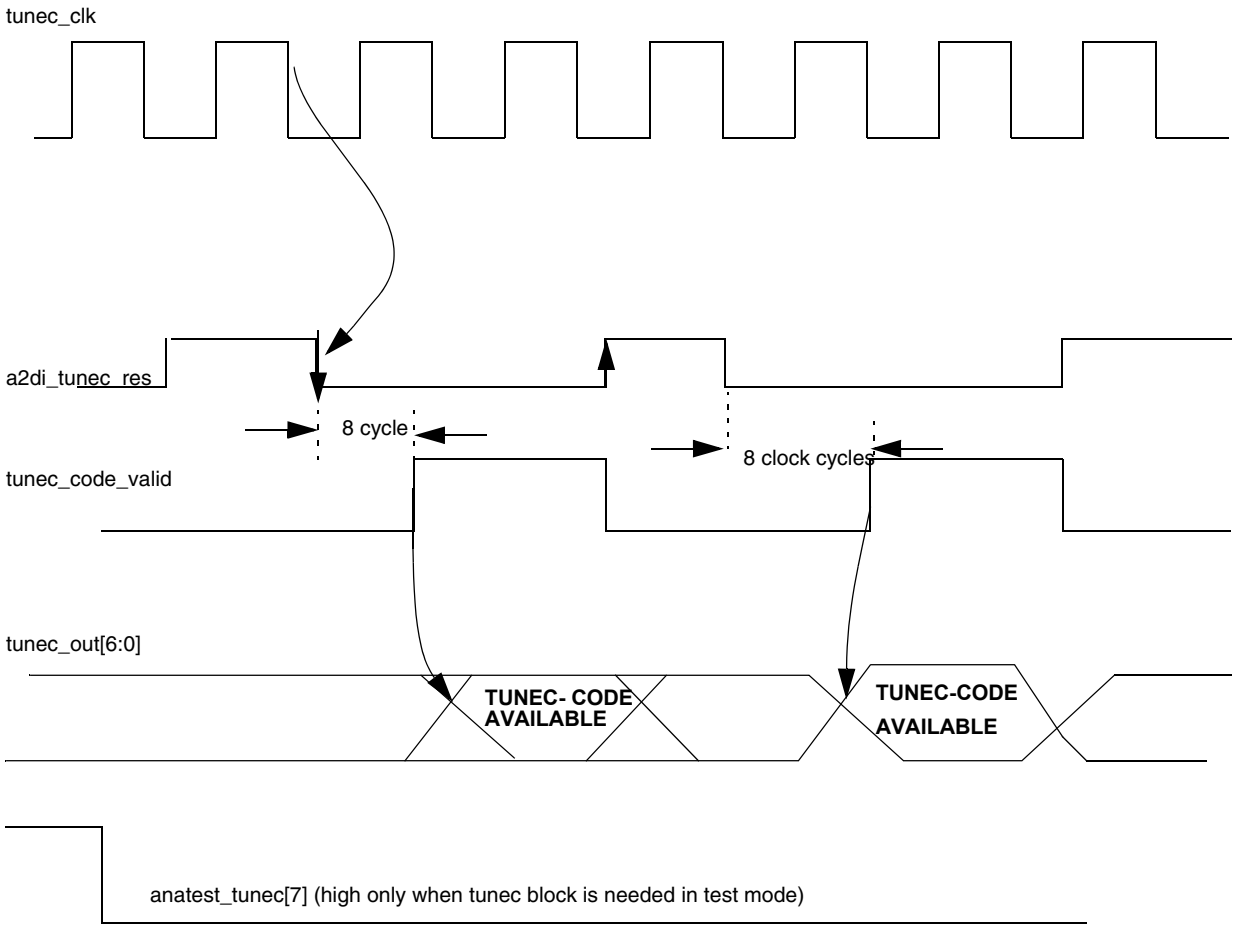


Figure 33-4. Timing Diagram

Chapter 34
Digital Phase Lock Loop (DPLL)

Revision History Table

Table 0-1.

Revision	Date	Author	Changes
1.0	10/8/99		Initial Release.
1.1	6/1/00	Rob Herring	Updated frequency-only lock time values.
1.2	09/21/00	Kathy Flories	Neptune Review Meeting
1.3	09/29/00	Yair Rosenbaum	remove 33.6.2.7 added more details on the numerator change procedure
1.4	11/16/00	Yair Rosenbaum	added 33.6.2.6 some corrections
1.5	02/05/01	Yair Rosenbaum	added some DFT and other tests ports
1.6	02/13/01	Yair Rosenbaum	Changed the ports names
1.7	03/09/01	Yair Rosenbaum	Changed the scan ports names 33.6.3.6
2.0	21/01/02	Sergey Sofer	Noise performance data update and more clear jitter definition
2.1	05/07/03		Updated for LTE specification release.

34.1 Introduction

An on-chip Digital Phase Lock Loop (DPLL) provides clock generation in digital and mixed analog/digital chips designed for wireless communication and other applications. The DPLL produces a high-frequency chip clock with a low frequency and phase jitter. The DPLL design basic concept is as follows:

- a new digital phase-frequency detection technique;
- fractional frequency multiplication method;

- digital control of an oscillator frequency.
- optional phase adjusting with digital phase loop filtering;
- a reduced multiplication factor range;
- compensation of parameter variations by means of self-calibration.

Digital implementation of frequency control and loop filtering functions allows the following new features:

- eliminating an on-board loop filter capacitor, minimization of internal capacitor value;
- selection of frequency and phase/frequency operation modes;
- an improved noise immunity, eliminating additional supply and ground pins;
- a high frequency resolution with a reduced lock time;
- reduced sensitivity to parameter variations caused by temperature and process.

The DPLL reference is an external system clock frequency. For wireless communication systems, the reference frequency is standardized. Most of the standard reference frequencies are located in the range of 10...30 MHz. The DPLL produces output clock (the double clock) at the frequency:

$$f_{dck} = 2 \cdot f_{ref} \cdot \frac{MF_I + MF_N / MF_D}{PDF}$$

— where f_{ref} - the reference frequency,

— MF_I - an integer part of a multiplication factor (MF),

— MF_N and MF_D - numerator and denominator of the MF fractional part;

— PDF- predivision factor.

The double clock has the 50% duty cycle and may be used for generation of a four-ticks chip clock.

Spectral purity of the DPLL output clock is characterized by phase and frequency jitter.

- cycle to cycle jitter is the difference between every two consecutive periods of an almost periodic waveform.
- phase (time interval) jitter is the phase difference between the almost periodic waveform and the periodic signal having the same average frequency. Along with phase jitter, the output clock may be skewed relative to the reference clock.
- frequency (period) jitter is the frequency (period) deviation of the almost periodic waveform from its average frequency (period).
- phase skew is the phase difference between two signals with the same or proportional frequency, has DC value and may vary around it due to phase jitter.

The DPLL jitter requirements are different according to a system configuration. For many stand-alone processors and asynchronous multiprocessor applications, only a frequency jitter value is important, and slow phase jitter and clock skew do not affect system performance. In such systems, it is not necessary to adjust an output clock phase with a phase of input clock. A clock generation mode, for which the slow phase fluctuations are permissible, is named in this document as the Frequency Only Lock (FOL) mode.

However, a phase error can be important for synchronous applications and sampling A/D and D/A precision converters. The DPLL mode providing minimal phase jitter and skew elimination is mentioned as the Frequency and Phase Lock (FPL) mode. The DPLL mode can be selected by a user.

The DPLL communicates with the Clock Module. This block contains a control register and provides an interface between the DPLL and the DSP Core.

This new PLL is not compatible with the previous one(HIP6W) with one main difference:

1. The Multiplication Factor Fractional Part Numerator (CMFN[23:0]) can be changed after the PLL was locked.

* The analog part is Gurgaon India Design Center responsibility.

34.2 DPLL Specifications

Parameters of the DPLL are given in Table 34-1. In this table, T^*_{ref} is a reference clock period after the predivider, T_{dck} is the output double clock period. In case the input clock and supply noise exceeds the specified limits, the PLL noise performance is not guaranteed.

The input clock phase noise at frequency < 50kHz will be transferred to the output one to one in case it exceeds the spec value

Table 34-1. DPLL Specifications

Parameter	Test conditions	Min	Typ	Max	Unit
Reference clock frequency range	Vcc=1.575 V	5	-	100	MHz
Double clock frequency range	Vcc=1.575 V	80	-	220	MHz
Predivision factor		1	-	16	
Multiplication factor integer part		5	-	15	
Multiplication factor numerator	should be less than denominator	-8,388,606	-	8,388,606	
Multiplication factor denominator		1	-	8,388,606	
Frequency lock time (After Partial reset)**	frequency jitter less than 4% of T_{dck}	150	225	300	T^*_{ref}
Phase lock time (After Partial reset)**		250	300	350	T^*_{ref}
Frequency jitter (defined in introduction, peak value)		-	0.03	0.04	$2*T_{dck}$
Phase jitter (defined in introduction, peak value)	FPL mode, integer and fractional MF	-	1.0	1.5	nsec
Phase shift for interval of 600ns (peak value) (USB requirement)	FPL mode, integer and fractional MF	-	-	1.0	nsec
Maximum allowed reference clock phase noise	$F_{modulation}$<50kHz 50kHz <$F_{modulation}$<300kHz $F_{modulation}$>300kHz	- - -	- - -	0.03 0.01 0.15	$2*T_{dck}$
Maximum allowed PLL supply ripple	$F_{modulation}$<50kHz 50kHz <$F_{modulation}$<300kHz $F_{modulation}$>300kHz	- - -	- - -	150 100 150	mV
Power supply voltage		1.48	1.575	1.67	V
Power dissipation	f_{dck} = 200 MHz Vcc=1.575 V	-	-	2	mW

** Partial Reset means restart the pll (means set ccm_crstrt to 1 and put 0 again)

34.3 Pins Description

The DPLL external signals are shown in Figure 34-1 and described by Table 34-2.

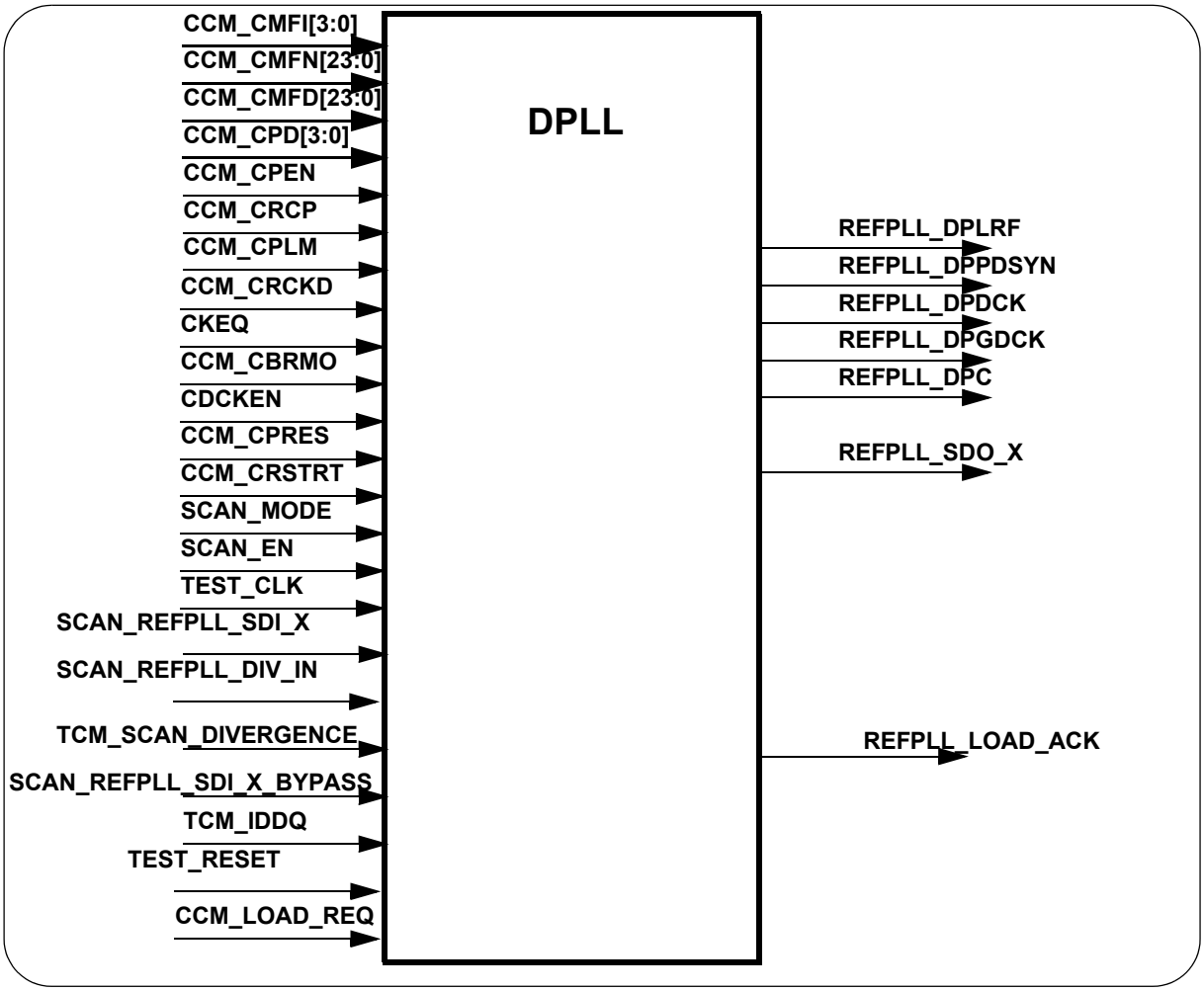


Figure 34-1. DPLL pins

Table 34-2. DPLL external signal description

Name	Type	Description
ccm_cmfi[3:0]	input	multiplication factor integer part
ccm_cmfn[23:0]	input	numerator of multiplication factor fractional part
ccm_cmfd[23:0]	input	denominator multiplication factor fractional part minus 1
ccm_cpd[3:0]	input	predivision factor minus 1
ccm_cpen	input	DPLL enable bit (‘1’ - DPLL enable)
ccm_crcp	input	reference clock polarity bit (‘0’ - non-inverted clock, ‘1’ - inverted clock)
ccm_cplm	input	phase lock mode bit (‘0’ - FOL, ‘1’ - FPL)
ccm_crckd	input	reference clock

Name	Type	Description
ckeq	input	clock **DPC0** delayed by equivalent delay
cdcken	input	double clock enable (‘1’ - double clock enable)
ccm_CBRMO	input	BRM order bit (‘0’ - first, ‘1’ - second)
ccm_cpres	input	power up reset (‘1’ - reset)
ccm_crstrt	input	DPLL restart (‘1’ - restart)
scan_mode	input	scan mode enable
scan_en	input	scan enable
test_clk	input	scan clock
scan_refpll_sdi_x	input	scan chain input
ccm_load_req	input	Notifies the DPLL that the numerator value has changed
refpll_dplrf	output	lock ready flag (‘1’ - lock)
refpll_dppdsyn	output	post-divider synchronization signal
refpll_dpdck	output	double clock
refpll_dpgdck	output	gated double clock
refpll_dpc	output	divided double clock C0 before equivalent delay
refpll_sdo_x	output	scan chain output
refpll_load_ack	output	indicates numerator has been successfully loaded into DPLL
scan_refpll_div_in	input	scan div
tcm_scan_divergence	input	scan tcm
scan_refpll_sdi_x_bypass	input	scan bypass
tcm_iddq	input	disable the VCO and the current reference (0 - enable, 1 - disable)
test_reset	input	test reset

34.4 Functional description

The DPLL functional block diagram is shown in Figure 34-2.

The reference frequency is divided by the Predivider with a division factor of 1...16. The Predivider introduces a constant delay independently on the predivision factor. The divided reference clock is fed to the Digital Phase-Frequency Detector (DPFD). The second DPFD input is connected to the binary rate modulator (BRM) output. The BRM gets the double clock divided by 2 (the chip clock). The BRM division factor is MF.

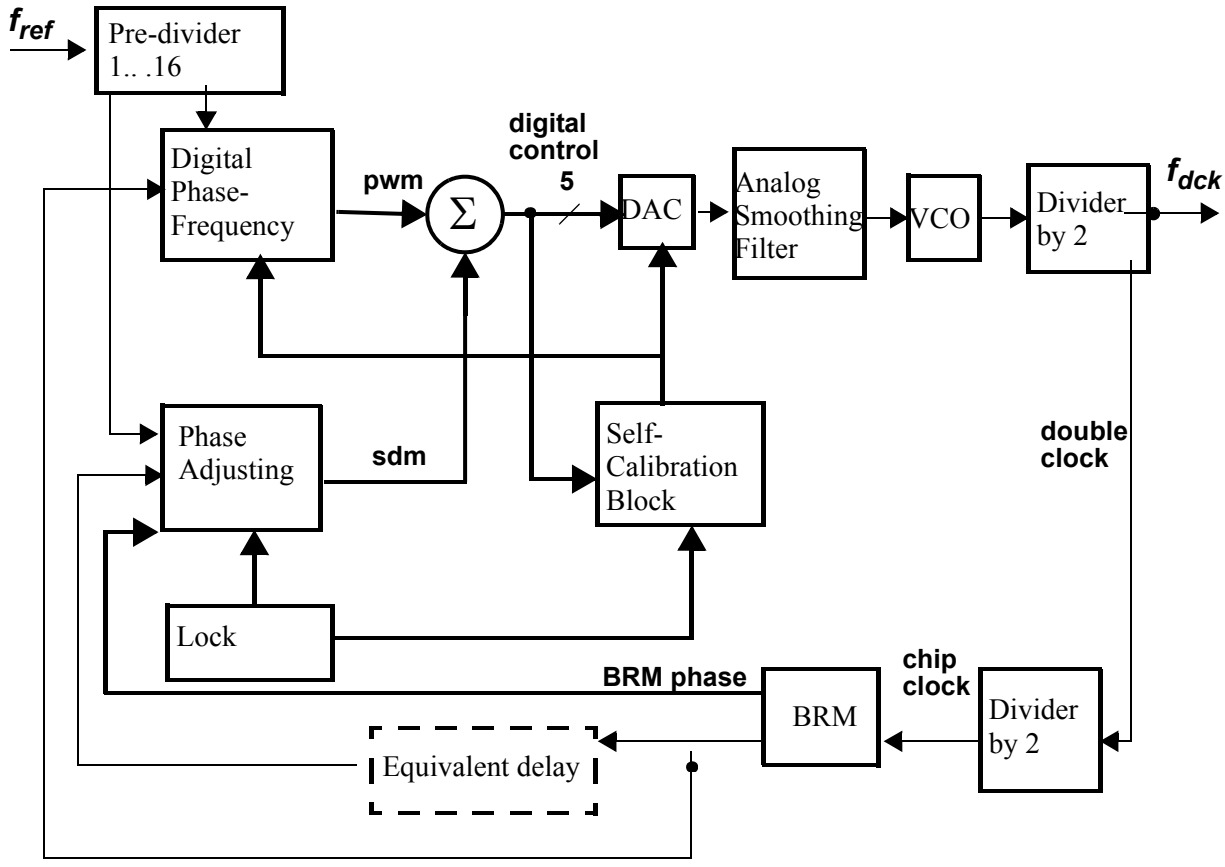


Figure 34-2. DPLL Functional Block Diagram

The DPFD is implemented as a 5-bit asynchronous up/down counter (see Figure 34-3). The counter content is incriminated by 1 at a positive edge of the reference clock and decremented by 1 at a positive edge of the divided double clock. Thus, the counter produces a multilevel PWM signal used for control of a frequency of the Voltage Controlled Oscillator (VCO).

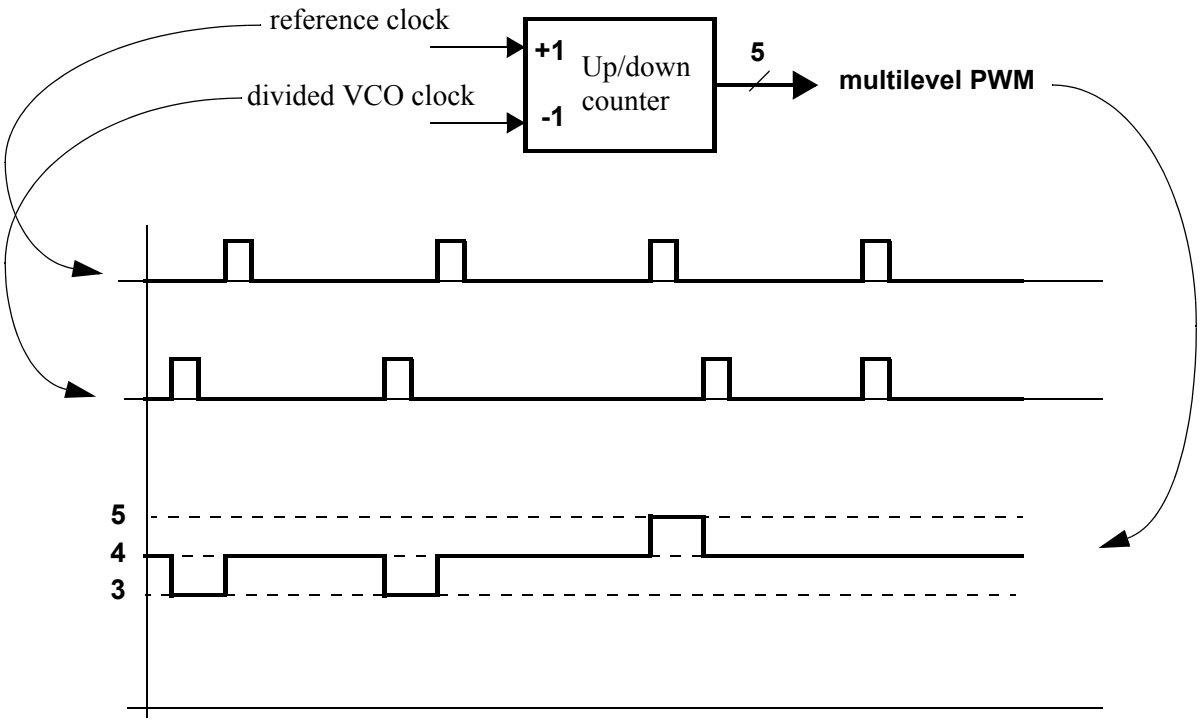


Figure 34-3. Digital Phase-Frequency Detector

Digital VCO control is performed by a 5-bit Digital-to-Analog Converter (DAC). A first order lowpass RC filter smooths the multilevel PWM signal from the DAC output. A frequency of the VCO controlled with the LPF output is proportional to a mean level of the PWM signal. The VCO frequency is two times higher than the double clock frequency. The Divider by 2 follows the VCO and generates the double clock.

During lock-in time the divided double clock frequency differs from the divided reference frequency. If the reference clock frequency is higher than the divided double clock frequency, the counter content increases that causes increased VCO frequency. If the feedback clock frequency is higher than the reference clock frequency, the counter content decreases lowering the VCO frequency.

The feedback loop closed via the BRM forces the system to go to a steady state, when both DPFD input signals have equal frequencies i.e. the multilevel PWM signal on the DPFD output has a constant duty cycle. The PWM duty cycle has such value that the mean DAC output tunes the VCO exactly to the proper frequency. The duty cycle may change slowly with variations of a VCO characteristic caused, for example, by temperature.

If the DPLL operates in the FOL mode, the Phase Adjusting Block is disabled and only the DPFD controls the VCO frequency. In this mode, the divided VCO frequency is exactly equal to the divided reference frequency, but there is an unrestricted phase offset between reference and output clocks. This phase offset can drift slowly with temperature and supply voltage changes.

In the FPL mode, the Phase Adjusting Block produces an additional control signal that minimizes the skew between reference and output clocks. Therefore, the skew is constant independent of temperature and supply voltage variations.

The Phase Adjusting unit block diagram is shown in Figure 34-4.

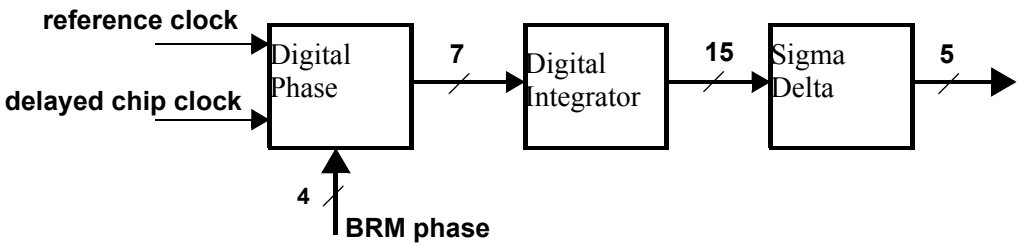


Figure 34-4. Phase Adjusting Block

The Phase Adjusting Block starts after frequency lock is achieved. The block contains a digital phase detector, a digital integrator and a sigma delta modulator. The divided VCO clock arrives to the digital phase detector via the Equivalent Delay unit. The Equivalent Delay unit compensates two delay components - an internal delay introduced by the DPLL and a delay of an external clock generator that produces chip clocks from the DPLL output clock. So the Equivalent Delay unit is composed of external and internal parts intended for compensation of both components.

A detected phase error is integrated by the digital integrator. The second-order sigma delta modulator converts a multibit integrator output to a 5-bit control signal to be added to the DPFD output. The sigma delta modulator operates at the chip clock rate. A quantization noise that appears due to conversion to the 5-bit signal is spectral shaped. Its power at low frequencies is negligible. The LPF effectively suppresses the high-frequency noise.

The BRM structure is shown in Figure 34-5. The BRM consists of a controlled divider with a division factor from 5 to 16 and a sigma delta modulator. The sigma delta modulator generates pulse series with a mean value of MF_N/MF_D . If the BRM output may be -1, 0, 1, 2. The total mean division factor is $MF=MF_I+MF_N/MF_D$.

Thus, acurrent period of the BRM output clock can be variable for a fractional MF but an average clock frequency is equal to $f_{dck}/(2*MF)$.

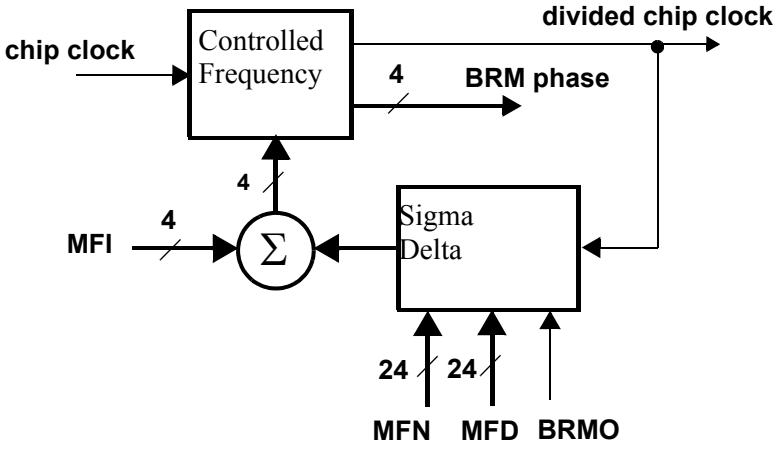
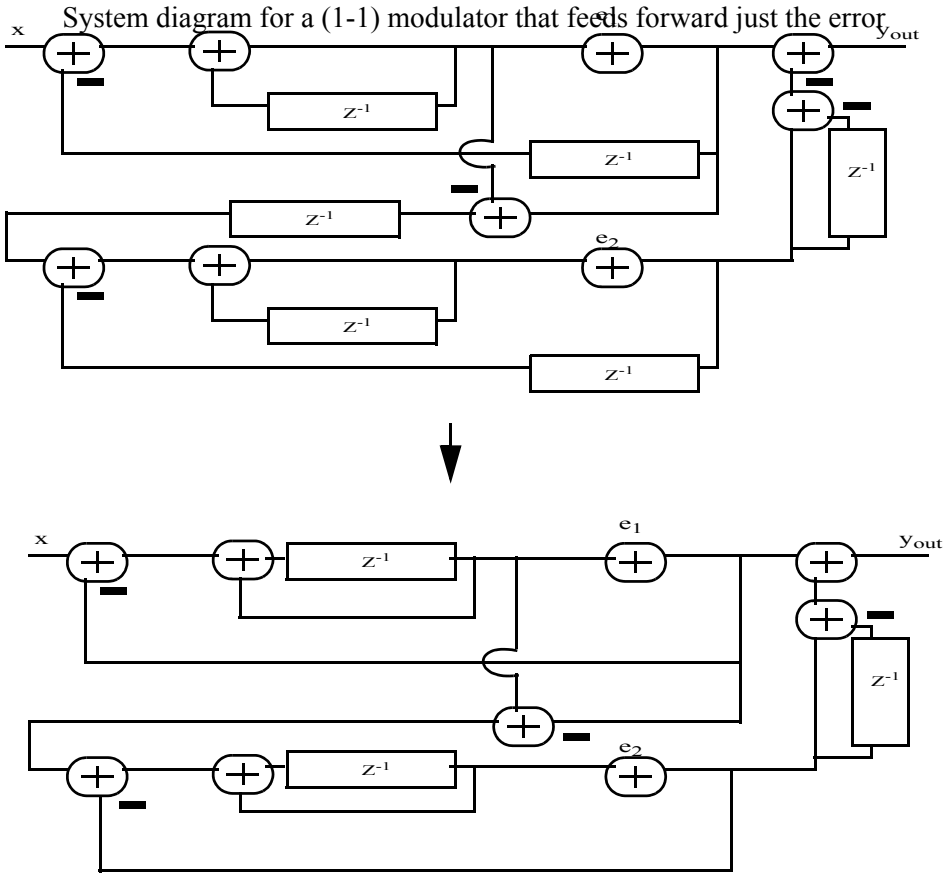


Figure 34-5. BRM structure

Design schematics of the Sigma-Delta Modulator which is in the BRM



The final design (lower) is equivalent to the (upper) design which can be found in the book Delta-Sigma Data Converters by S.R. Norsworthy p.202

The DPLL includes also the Self-Calibration Block. This block regulates the gain of the DAC at start-up in order that the total loop gain is proportional to the reference frequency and independent of the multiplication factor and process variations.

The self-calibration method is illustrated by Figure 34-6. Self-calibration is performed by two DACs: the reference current DAC and the main control loop multiplying DAC.

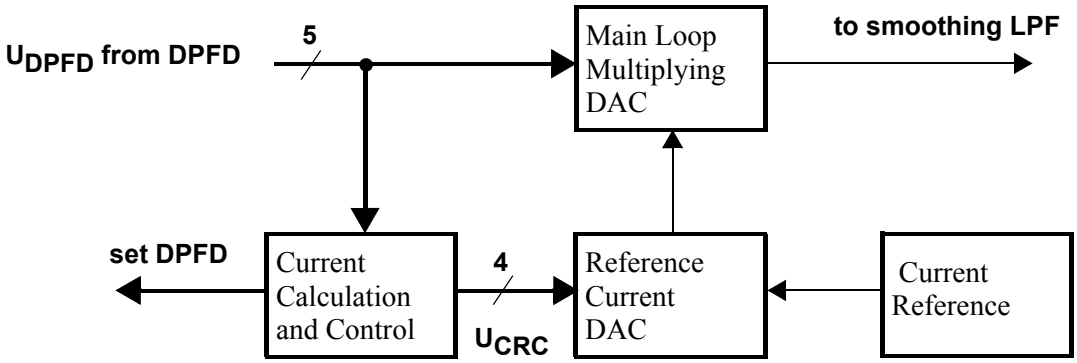


Figure 34-6. Self Calibration scheme

The main loop multiplying DAC is made of adjustable current sources with an input code U_{DPFD} from the DPFD output. The reference current DAC input is the code U_{CRC} from the Current Calculation and Control unit. The U_{CRC} signal is an unsigned 4-bit code. The combined arrangement yields an output current $U_{DPFD}*U_{CRC}*I_0$, where I_0 is a reference current. The output current controls the VCO frequency.

Self-calibration is performed in two steps:

a) The control code for reference current DAC is set to an initial value U_{CRC0} equal to 8. The DPLL is enabled. After frequency lock has been achieved (the DPFD output code is U_{DPFD0}), the desired U_{CRCN} is calculated for a nominal value of U_{DPFDN}. The U_{DPFDN} value is equal to 16 - approximately a half of the U_{DPFD} full scale.

b) The value of U_{CRC} is set to U_{CRCN} and U_{DPFD} - to U_{DPFDN}. After such setting, the control current stays practically unchanged and the DPLL frequency is very close to the desired frequency. Only some small refining the DPLL frequency may occur after calibration.

Calculation of the U_{CRCN} is accomplished according to the following equation:

$$U_{CRCN} = \frac{U_{CRC0} \cdot U_{DPFD0}}{U_{DPFDN}}$$

After this calibration, the value of U_{DPFDN} points to the center of the U_{DPFD} range. The frequency step size is

$$\Delta f = \frac{f_{VCO}}{U_{DPFDN}} = \frac{f_{ref} \cdot MF}{U_{DPFDN}}$$

Since U_{DPFDN} is constant and the loop gain is proportional to $\Delta f/MF$, the loop gain depends only on the reference frequency. This is a desired feature which allows to control the PLL bandwidth, jitter and stability.

This calibration adjusts both the VCO frequency step size and assures that the VCO will be close to the center of the frequency range. This will result in wider frequency range and lower jitter.

The Lock Control module contains a set of timers intended to generate all necessary control signals during lock-in time. The module also controls the lock flag.

34.5 DPLL linear analysis

The DPLL linear model for FOL mode is shown in Figure 34-7.

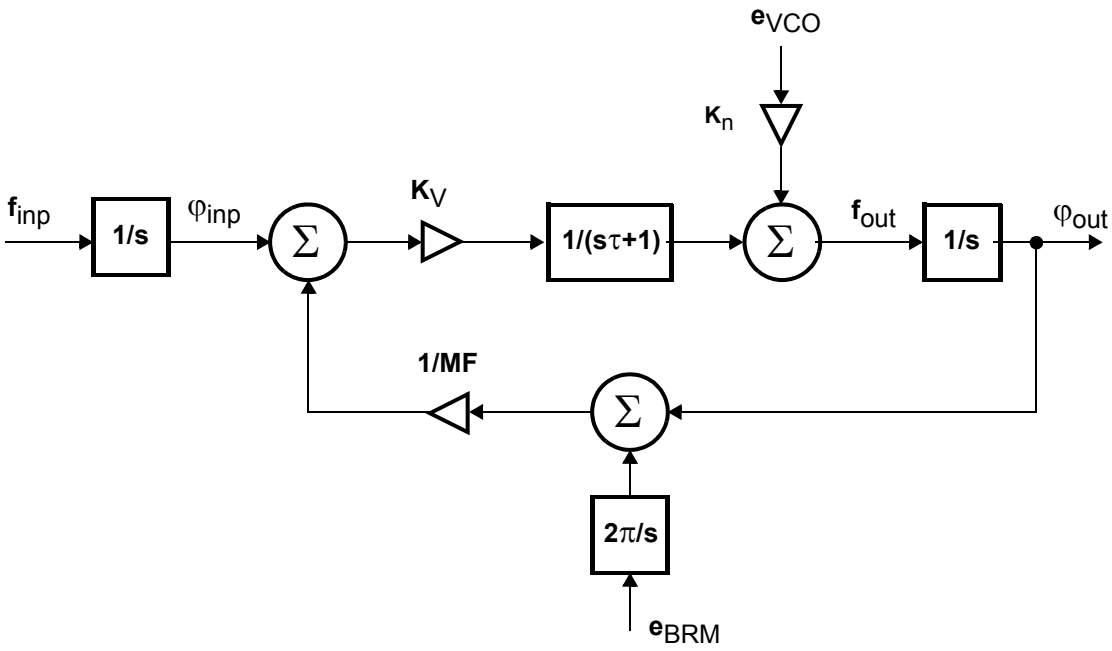


Figure 34-7. The DPLL linear model for FOL mode

In Figure 34-7, f_{inp} and φ_{inp} are a frequency and a phase of the reference clock, f_{out} and φ_{out} are a frequency and a phase of the output clock, K_V is a total VCO and DAC gain defined as a VCO frequency step (Hz) corresponding to the unity step on the DAC input, e_{VCO} and e_{BRM} are VCO and BRM noises, τ is a LPF time constant, K_n is a gain for the VCO noise.

According to this model, the DPLL transfer function relative to the input phase is

$$H_{\varphi}(s) = \frac{K_V}{\tau \cdot s^2 + s + K_V/MF}$$

Because of self-calibration, $K_V=f_{ref}*MF/16$. Therefore, the transfer function is

$$H_{\varphi}(s) = \frac{f_{ref} \cdot MF/16}{\tau \cdot s^2 + s + f_{ref}/16}$$

The transfer function for the VCO noise is as follows:

$$H_n(s) = \frac{K_n \cdot (s \cdot \tau + 1)/\tau}{\tau \cdot s^2 + s + f_{ref}/16}$$

The transfer functions for f_{ref}=10 MHz, τ=1 μsec, MF=10, K_n=1MHz/V are shown in Figure 34-8.

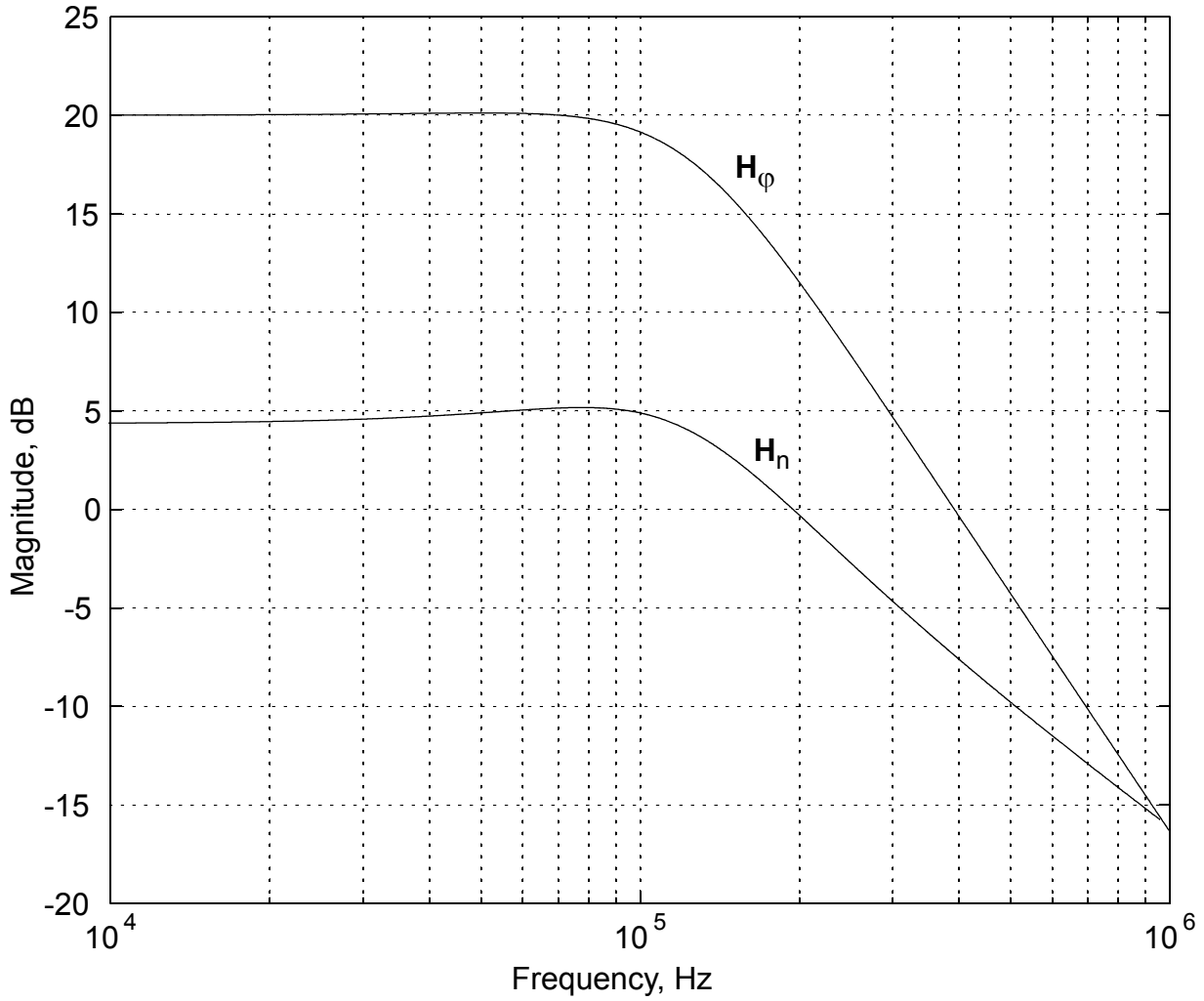


Figure 34-8. The DPLL transfer functions for input phase and VCO noise

The DPLL linear model for FPL mode is shown in Figure 34-9. Here, K_i is a phase integrator gain, e_{VCO} is a quantization noise introduced by the sigma-delta modulator.

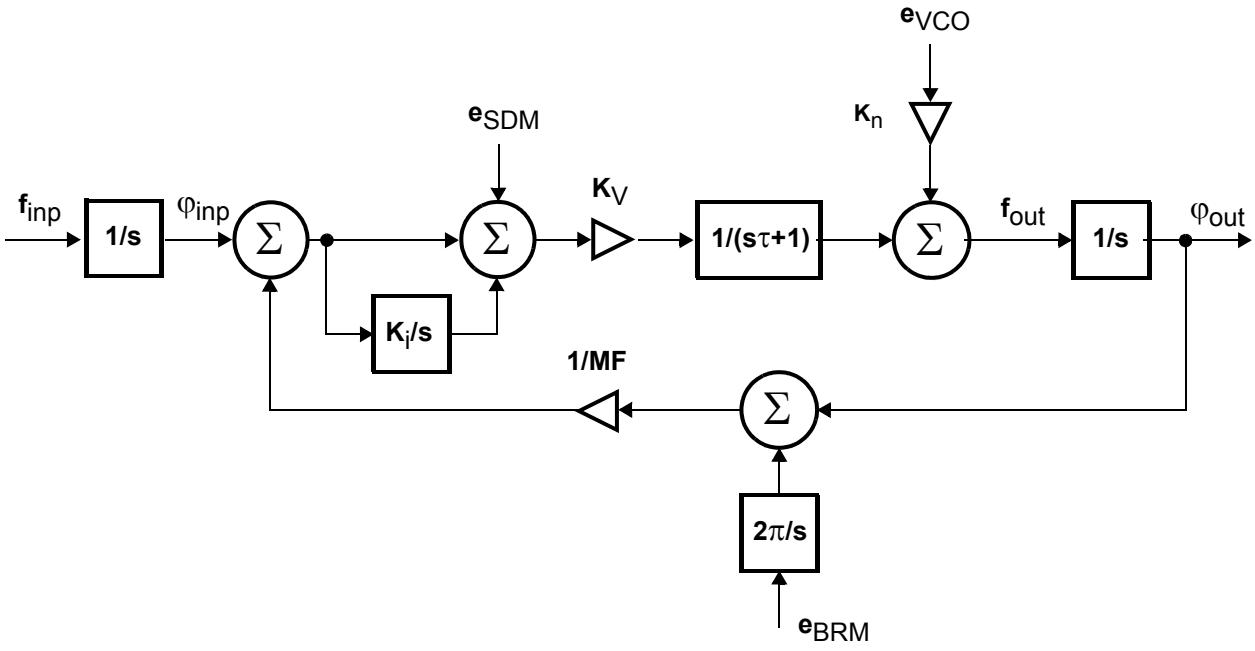


Figure 34-9. The DPLL linear model for FPL mode

The DPLL open loop transfer function in the FPL mode is described by equation:

$$H_{ol}(s) = \frac{K \cdot \left(1 + \frac{s}{\omega_z}\right)}{s^2 \cdot \left(1 + \frac{s}{\omega_p}\right)}$$

where $K=K_{i*}K_V/MF$ is an open loop DC gain, $\omega_z=K_i$ is a zero frequency, $\omega_p=1/\tau$ is a pole frequency.

The transfer function for the input phase in the FPL mode is as follows:

$$H_{\varphi}(s) = \frac{H_{ol}(s) \cdot MF}{1 + H_{ol}(s)}$$

and for the VCO noise

$$H_{\varphi}(s) = \frac{K_n / s}{1 + H_{ol}(s)}$$

After DPLL self-calibration, the integrator gain $K_i=f_{ref}/(2*\pi^2)$. Figure 34-10 shows the transfer functions in the FPL mode.

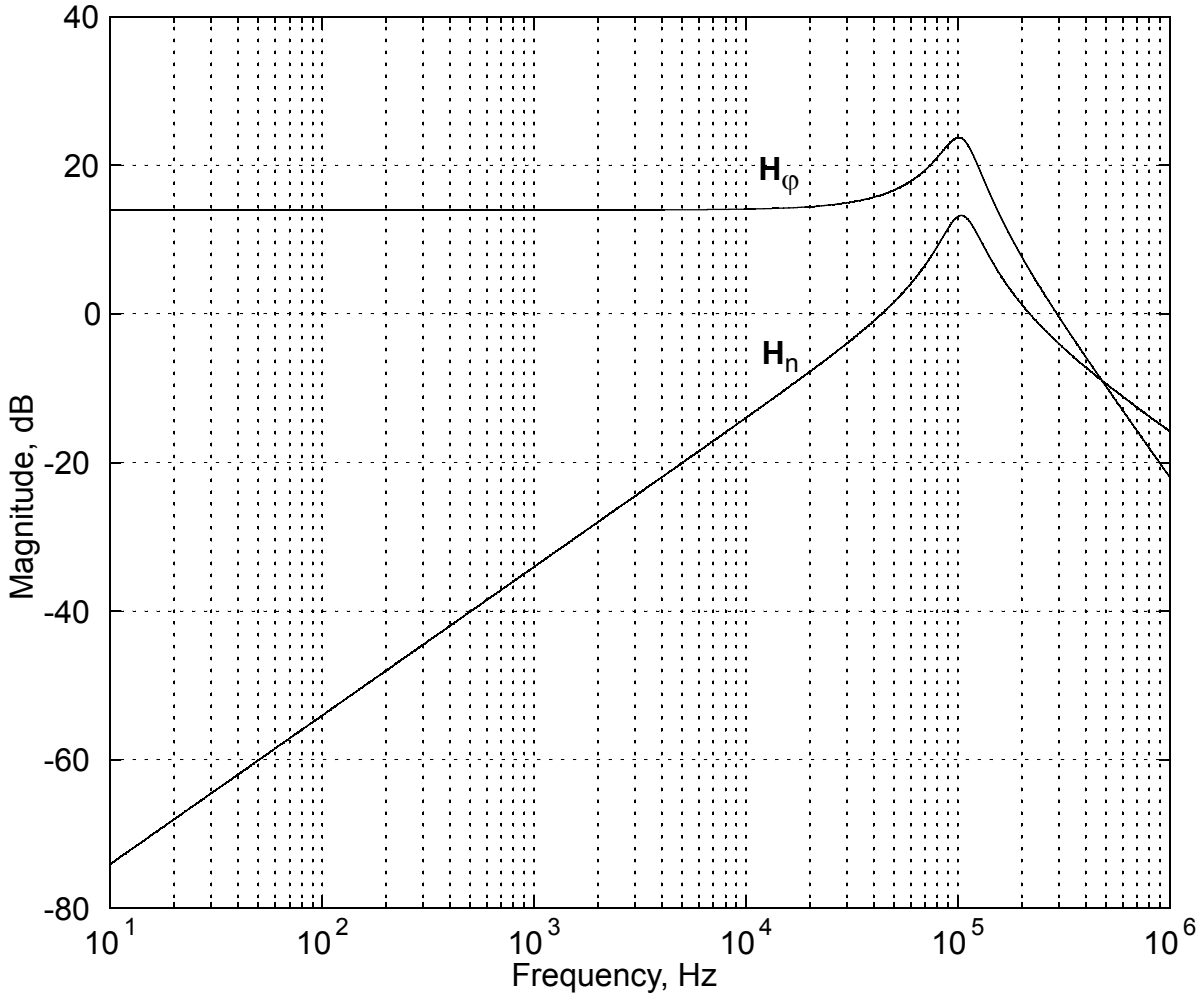


Figure 34-10. The DPLL transfer functions in the FPL mode

Because the digital phase detector is non-linear, the phase integrator gain depends on a phase error. The graphs given in Figure 34-10 correspond to the steady state, when the output phase fluctuates near the desired baseline and the integrator gain is constant.

34.6 DPLL interface signals

34.6.1 DPLL reset

There are three reset signals for the DPLL: the power-up reset CPRES, DPLL enable CPEN and restart CRSTRT signals.

The DPLL is operating when CPRES and CRSTRT are ‘0’ and CPEN is ‘1’, otherwise it is disabled. The DPLL states are presented in Table 34-3.

Table 34-3. The DPLL states

CPRES	CPEN	CRSTRT	State
1	X	X	all DPLL modules are reset
X	0	X	
0	1	1	DPLL modules are reset excluding current reference
0	1	0	DPLL is operating

After the CPRES signal has been set, all the DPLL modules are reset immediately. Reset caused by the CPEN and CRSTRT signals effects with delay of three double clocks. It stops the double clock series synchronously as shown in Figure 34-11.

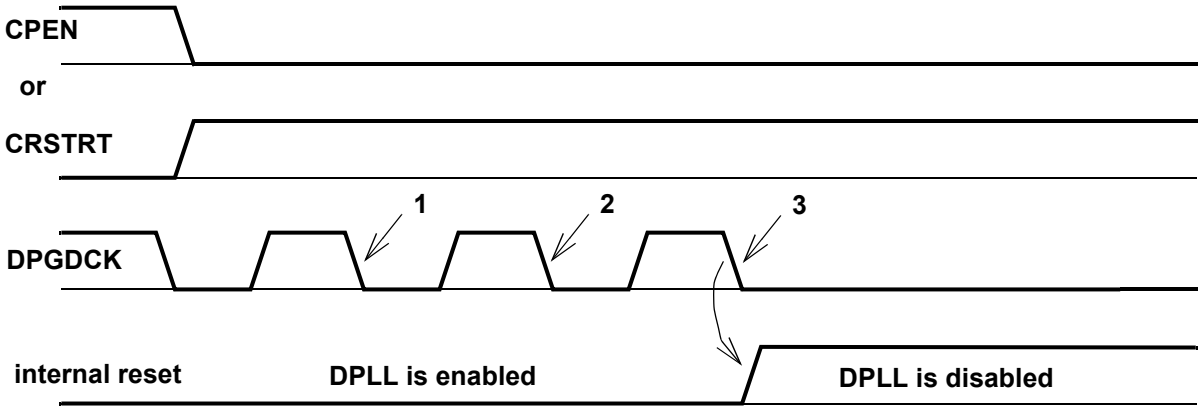


Figure 34-11. The timing diagrams for DPLL stop

The DPLL starts during three reference clock periods (see Figure 34-12). This is provided by a synchronizer.

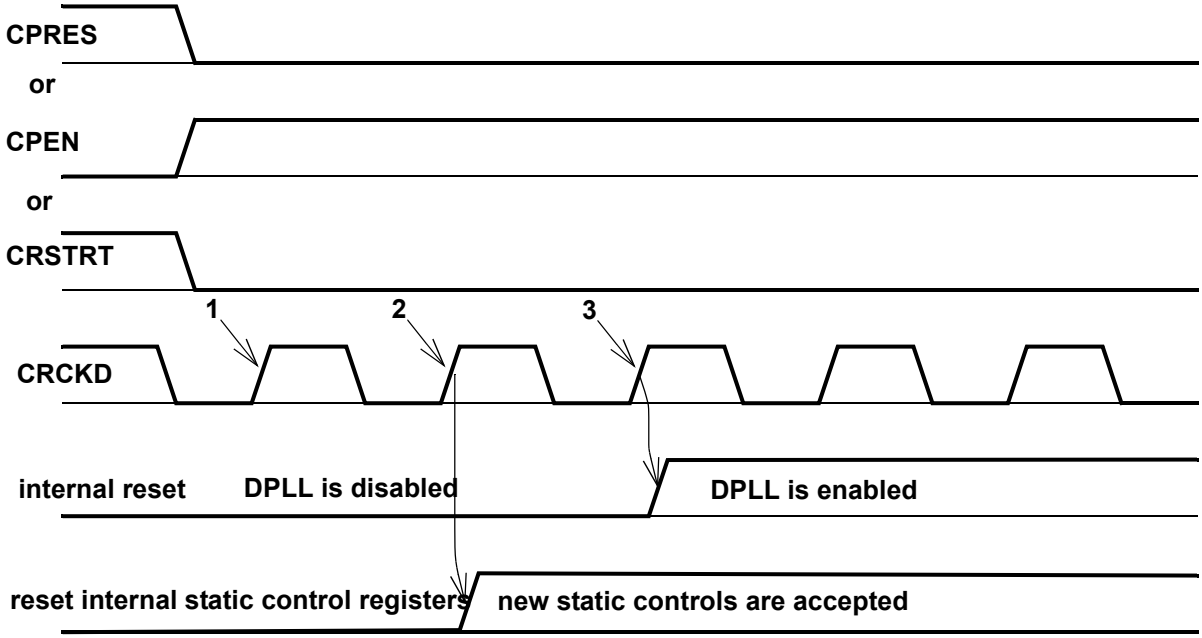


Figure 34-12. The timing diagrams for DPLL start

The VCO is enabled with delay of 31 divided reference clocks after enabling the current reference. If the current reference wasn't disabled during reset (it takes place when the DPLL restarts), the VCO is enabled simultaneously with enabling all the DPLL logic.

34.6.2 Static control signals

The static control signals are accepted by the DPLL at the second positive edge of the reference clock after the CPRES or CRSTST signals have changed to '0' or the CPEN signal has changed to '1' (see Figure 34-12).

34.6.2.1 Predivision Factor Minus One (CPD[3:0])

The CPD[3:0] bits give the predivision factor minus 1. The CPD[3:0] number should be in a range from 0 to 15.

34.6.2.2 Multiplication Factor Integer Part (CMFI[3:0])

The multiplication factor integer part CMFI [3:0] bits should be in a range from 5 to 15. If it is less than 5, the DPLL accepts 5.

34.6.2.3 Multiplication Factor Fractional Part Denominator(CMFD[23:0])

The CMFD[23:0] bits, in a 2's compliment format, give the denominator (bit 23 is always zero). The CMFD[23:0] number should be in a range from 1 to 8,388,606, otherwise the output clock frequency will differ from the desired frequency. The CMFD bits are ignored if the MF numerator is zero.

34.6.2.4 Reference Clock Polarity (CRCP)

If the CRCP bit is cleared, the chip clock is adjusted with a positive edge of the reference clock. If the bit is set, the chip clock is adjusted with its negative edge.

34.6.2.5 Phase Lock Mode (CPLM)

The DPLL operates in the Frequency Only Lock mode, when the CPLM bit is cleared, and in Frequency and Phase Lock mode, when the bit is set. The FPL mode can be used for both an integer and fractional multiplication factor, but phase skew elimination is accomplished only for the integer MF.

34.6.2.6 BRM Order (CBRMO)

When the CBRMO bit is cleared, the BRM has first order, otherwise the BRM order is 2. The first-order BRM should be used only if the MF fractional part denominator is less than 8. In other cases, the BRM order should be 2. This bit is ignored if the MF numerator is zero.

34.6.3 Clocks and other dynamic signals

34.6.3.1 Multiplication Factor Fractional Part Numerator (CMFN[23:0])

The CMFN[23:0] bits, in a 2's compliment format, give the numerator. The CMFN[23:0] bits are the only bits in the PLL that can be changed after the PLL was locked without reset. The absolute change should not increase TBD which might results loss of PLL LOCK. The CMFN[23:0] number should be in a range from -8,388,606 to 8,388,606. If the absolute value of the numerator is larger than denominator (i.e. |numerator|>denominator), the output clock frequency will differ from the desired frequency. If the numerator is zero, a circuit for fractional division is disabled to save power.

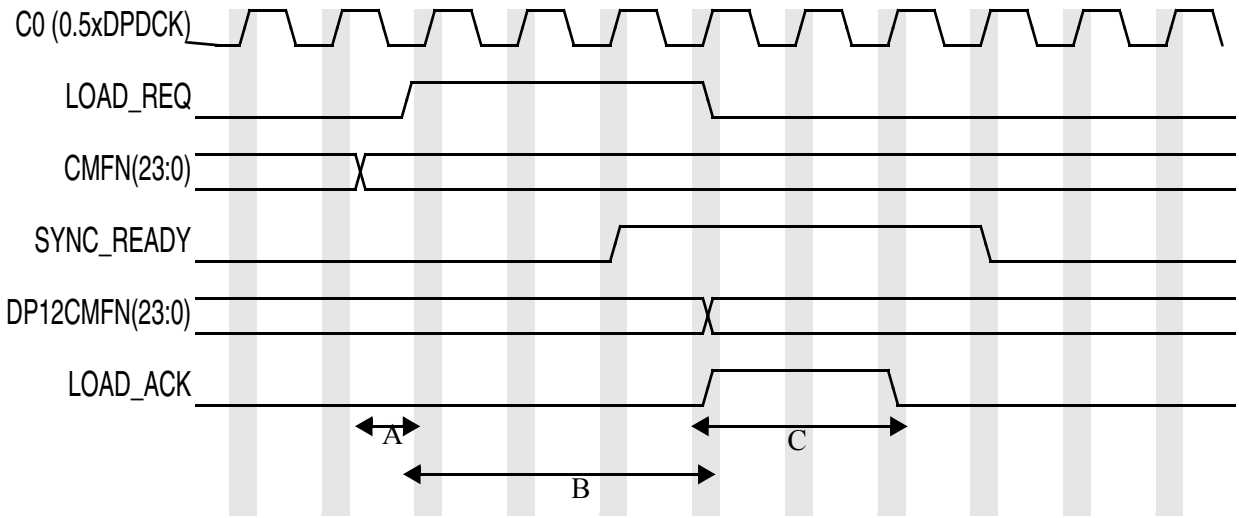


Figure 34-13. The timing diagrams for the CMFN

A- The difference between LOAD_REQ and the new value of CMFN should be grater or equal to zero.

B- The duration from posedge LOAD_REQ to the actual loading of the CMFN new value to the PLL is at least three clocks.

C- LOAD_ACKnolige signal duration is two clocks.

34.6.3.2 Reference Clock (CRCKD)

The reference clock frequency can be in a range from 5 to 100 MHz. For better DPLL performance, the clock duty cycle should be 50+/-20%. The DPLL stays in reset state until the reference clock will appear.

34.6.3.3 Double Clock (DPDCK)

The double clock is the DPLL output and has 50+/-5% duty cycle. This signal is zero during reset. The clocks appear at start of the lock-in time interval.

34.6.3.4 Gated Double Clock (DPGDCK) and Double Clock Enable (CDCKEN)

The DPGDCK output signal is obtained from the double clock by gating with the CDCKEN input signal. Because CDCKEN is synchronized in the DPLL, no spikes or shorted phases take place. Figure 34-14 shows the gating timing diagrams.

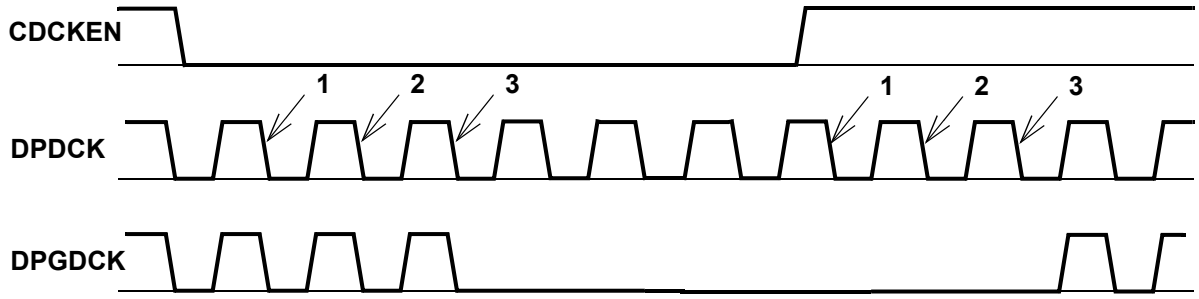


Figure 34-14. The clock gating timing diagrams

34.6.3.5 Post-Divider Synchronization (DPPDSYN)

The DPPDSYN signal is the DPLL output intended for synchronization of a post-divider that can follow the DPLL. Synchronization ensures a constant and minimal phase offset between the reference and post-divider output clocks. The DPPDSYN signal is used only for an integer MF (see Figure 34-15). When MF is non-integer, DPPDSYN is zero.

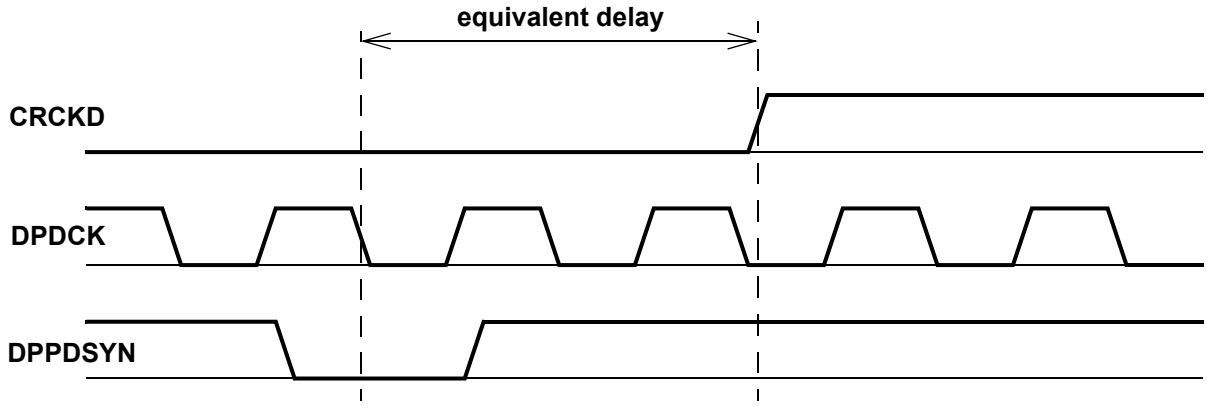


Figure 34-15. The timing diagrams for the DPPDSYN signal

34.6.3.6 Equivalent Delay Input (DPC) and Output (CKEQ) signal

The DPC signal is fed to the Equivalent Delay. The CKEQ signal is a delayed copy of DPC.

34.6.3.7 Lock Ready Flag (DPLRF)

This output signal, when set, indicates that the DPLL is in lock.

If MF is integer and the DPLL is in the FPL mode, the DPLRF bit is primarily set when a phase error is reduced to zero. After that, the bit can be zeroed in the case that the phase error has increased over 25% of the chip clock period.

For a non-integer MF or in the FOL mode, the DPLRF bit is set when the given number of reference clocks arrives after the DPLL starts.

34.6.4 Test mode signals

The test mode signals provide scan testing the DPLL. The DPLL is in the test mode when SCAN_MODE is ‘1’. This signal preserves all asynchronous resets during testing and switches internal clocks to the test clock signal TEST_CLK. The SCAN_EN signal controls serial/parallel modes of flip-flops during testing. The DPLL contains two scan chains with the scans inputs SCAN_REFPLL_SDI_X and the scan outputs REFPLL_SDO_X.

Chapter 35
General Purpose ADC (GPADC)

Revision History Table

Revision	Date	Author	Changes
0.0	03/25/02	May Len	Initial release : Copied from LT latest rev.
0.1	04/18/02	May Len	Remove some testing signals from Table 34-2.
0.2	05/07/03		Updated for LTE specification release.

35.1 General Purpose ADC Block Description

The general purpose ADC will convert an analog signal into a 10-bit digital word and store the digital data in A2DIGL. The control of general purpose ADC is handled by A2DIGL, which is described in Section 35.5.2, General Purpose ADC Interface.

In normal operation, the mux in GPADC selects a data from three inputs: PAD_A2D_DATA, TX_TX_CP_div2, and TX_RX_CP_div2. The GPADC takes four samples from PAD_A2D_DATA. And then it takes one sample each from TX_TX_CP_div2 and TX_RX_CP_div2 using 13 MHz input clock.

In deep sleep mode, the GPADC only samples the data from PAD_A2D_DATA six times using 32 KHz input clock. GPADC will be powered down between samples.

The data at the mux output will pass through a first-order, anti-aliasing, passive low-pass filter (LPF). The -3dB cutoff frequency of the filter is approximately 2.5 MHz at typical conditions.

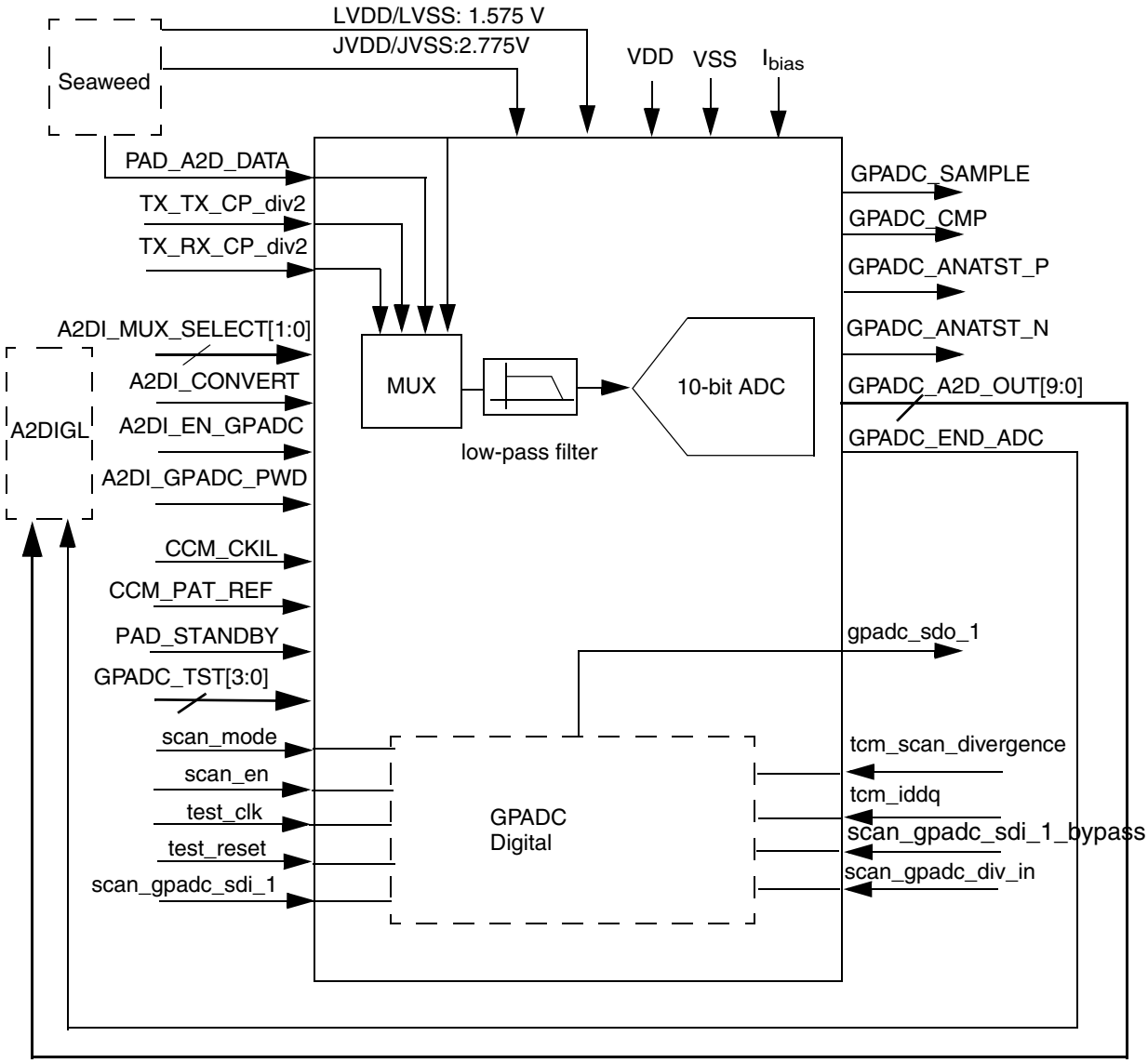


Figure 35-1. GPADC Block Diagram

35.2 General Purpose ADC Input/Output Description

35.2.1 GPADC Module Pin List

Table 35-1 lists all the pins in the GPADC module.

Table 35-1. GPADC Module Pin List

Parameter	Direction	Description	Min	Typ	Max	Units
pad_a2d_data	Input	Analog Signal from Seaweed ADC Multiplexer and Level Translator Block.	0.2		2.45	V
tx_tx_cp_div2	Input	Neptune TX synthesizer charge pump **buffered** output.	0.2		2.45	V
tx_rx_cp_div2	Input	Neptune RX synthesizer charge pump **buffered** output.	0.2		2.45	V
lvdd	Input	voltage reference for GPADC 1.575 +/- 2%	1.54	1.575	1.61	V
jvdd	Input	Voltage supply from IO_REG regulator in Seaweed.	2.68	2.775	2.88	V
jvss, lvss	Input	Clean Analog ground used for IO_REG regulator in Seaweed.		0		V
vdd	Input	Voltage supply by the digital core regulator in Neptune. It should be present in deep sleep mode.	1.48	1.575	1.67	V
vss	Input	Digital ground in Neptune.		0		
regul_txrx_gpadc_ibias	Input	Current generated by Neptune current bias block	0.7	1	1.3	uA
a2di_en_gpadc	Input	After A2D_SYNC (in Neptune A2DIGL block) or PAD_STANDBY (from padring) signal goes high, EN_GPADC goes high. EN_GPADC stays high during sampling, and returns to low after six samples in normal operation mode. In deep sleep mode, EN_GPADC toggles (See Figure 36-3 on page 36-24 and Figure 36 on page 36-1 in Chapter 36, “Mixed-Signal Control Interface (A2DIGL).”) When EN_GPADC =’1’, digital part of GPADC is enabled. When EN_GPADC =’0’, digital part of GPADC is in reset state.				

Table 35-1. GPADC Module Pin List

Parameter	Direction	Description	Min	Typ	Max	Units
a2di_convert	Input	Signal from Neptune A2DIGL block to mark the beginning of the conversion. The conversion starts at the rise edge of the CONVERT signal.		TBD		
ccm_ckil	Input	32 KHz Clock signal from CCM		32		KHz
ccm_pat_ref	Input	13 MHz Clock signal from CCM		13		MHz
anatest_gpadc_test[3:0]	Input	Testing bits from Analog Test module.				
a2di_mux_select[1:0]	Input	Select signals for the GPADC mux [0,0]: A2D_DATA is selected [0,1]: TX_CP_div2 is selected [1,0]: RX_CP_div2 is selected. [1,1]: This state **should not** occur in normal or deep sleep mode operation. If it happens, analog ground (JVSS) will be selected.				
a2di_gpadc_pwd	Input	GPADC analog supply is powered down when PWD='1'. Analog supply is powered up when PWD='0'. The maximum turn on time of GPADC is 5 us.			5	us
pad_standby	Input	In deep sleep mode, STANDBY is set to '1'. It is from Padring.				
scan_mode	Input	scan input from TCM				
scan_en	Input	scan input from TCM				
test_clk	Input	scan input from TCM				
test_reset	Input	scan input from TCM				
scan_gpadc_sdi_1	Input	scan input from TCM				
tcm_scan_divergence	Input	scan input from TCM				
tcm_iddq	Input	scan input from TCM				
scan_gpadc_sdi_1_bypass	Input	scan input from TCM				
scan_gpadc_div_in	Input	scan input from TCM				
Output Signals						
gpadc_a2d_out[9:0]	Output	10 bit digital data signal generated by the analog to digital convertor.				V

Table 35-1. GPADC Module Pin List

Parameter	Direction	Description	Min	Typ	Max	Units
gpadc_end_adc	Output	The rising edge of GPADC_END_ADC signals the end of each data conversion.				V
gpadc_sample	Output	For debugging only.				V
gpadc_cmp	Output	For debugging only. It will be set to “0” when the module is powered down.				V
gpadc_anatest_p	Output	Output to analog testing module. It is set to “0” when GPADC module is not in testing mode.				
gpadc_anatest_n	Output	Output to analog testing module. It is set to “0” when GPADC module is not in testing mode.				
gpadc_sdo_1	Output	scan output to TCM				

** Note: All digital signals are powered by 1.575 V core vdd.

35.2.2 Testing Pins

For testing and debugging purposes, 4 testing pins are needed for GPADC block. In normal mode and deep sleep mode operation, all testing pins should be set to ‘0’. Most of the signals in Table 34-2 are GPADC internal names. Please refer to gpadc_analog schematic for details.

Table 35-2. GPADC Testing Truth Table

State	GPADC_ TST3	GPADC_ TST2	GPADC_ TST1	GPADC_ TST0	GPADC_ANATST_P	GPADC_ANATST_N	Comment
0	0	0	0	0	JVSS	JVSS	non-testing mode
1	0	0	0	1	sah_out	sah_lv	
2	0	0	1	0	sah_out	sah_out1	sah_out1 is buffered 1.575v
3	0	0	1	1	vout_lpf	sah_vfb	
4	0	1	0	0	pbref	pbvin	
5	0	1	0	1	sah_pba	sah_pbb	
6	0	1	1	0	tst_bias_r	tst_bias_x	
7	0	1	1	1	vdsg	vcin	
8	1	0	0	0	JVSS	JVSS	
9	1	0	0	1	JVSS	JVSS	

Table 35-2. GPADC Testing Truth Table

State	GPADC_ TST3	GPADC_ TST2	GPADC_ TST1	GPADC_ TST0	GPADC_ANATST_P	GPADC_ANATST_N	Comment
10	1	0	1	0	TX_RX_CP_div2	TX_TX_CP_div2	see note below
11	1	0	1	1	LVSS	master_nbias_in	
12	1	1	0	0	JVSS	JVDD	
13	1	1	0	1	JVSS	JVDD	high bias mode
14	1	1	1	0	JVSS	JVDD	low bias mode
15	1	1	1	1	JVSS	JVDD	bypass mode

Note: When testing TX_TX_CP_div2 and TX_RX_CP_div2, set a2di_bsavetrsynt = 0 and cptest[2:0] = 000. In addition, set a2di_tr_sel = 1 to select TX_TX_CP_div2, set a2di_tr_sel = 0 to select TX_RX_CP_div2.

The control and data register address of GPADC in analog test mode is listed in Table 34-3, Table 34-4, Table 34-5, and Table 34-6.

Table 35-3. GPADC ANA_CNTL and ANA_DATA2 Register
Figure 0-4.

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ANA_CNTL ($2484_4010)	R																
	W																
	R		anatest_en1	anatest_en0								TST_DC[5:0]					
	W																
ANA_DATA2 ($2484_4018)	R																
	W																
	R W					REG[7:0]								GPADC[3:0]			

Table 35-4. Analog Mux1 GPADC Selection

TST_DC3	TST_DC2	TST_DC1	TST_DC0	Test_output select
0	1	1	0	GPADC_P,N

Table 35-5. Analog Mux2

TST_DC5	TST_DC4	Output_type select
0	0	Unbuffered
0	1	Digital Buffer
1	0	Analog Buffer
1	1	VSS Offset

The data and control register address listed in Table 34-3, Table 34-4, and Table 34-5 is defined in Neptune TCM module spec.

Table 35-6. GPADC Control/Status Register
Figure 0-5.

Name		31	30	29	28	27	26	25	24	23	22
		15	14	13	12	11	10	9	8	7	6
CTRL $2484_6028	R	0	0	0	0	0	0	0	0	0	0
	W										
	R	0	0	0	0	0	0	0	0	TST	PWD
	W										

The control register listed in Table 34-6 is defined in Neptune A2DIGL module spec. When TST = 1, the GPADC will enter a special test mode in which external T_CONVERT will be used as the convert signal to the GPADC module. PAD_A2D_DATA is the only input for GPADC.

35.3 General Purpose ADC Electrical Specification

Table 35-7. GPADC Specification

Parameter	Conditions	Min	Typ	Max	Units
Resolution			10		bits
Input Dynamic Range		0.2		2.45	V
Offset Error		- 10		+ 10	LSB
Full-Scale Error		- 25		+ 25	LSB
INL		- 3		+ 3	LSB
DNL		- 1		+ 1	LSB

Table 35-7. GPADC Specification

Parameter	Conditions	Min	Typ	Max	Units
Conversion Current	Normal Operation			2	mA
DSM Current	Deep Sleep Mode, five conversions per-second		5		uA
Power Down Current			2		uA
Conversion Time	Normal Operation			31	uS
Turn On Time				5	uS

GPADC uses LVDD (1.575 V +/- 2%) as its conversion reference. The A/D count calculation is based on the equation shown below:

$$\text{A/D Count} = \text{INT}\left[\frac{\text{pad\_a2d\_data} - (0.1269*\text{lvdd}) +/- (10\ \text{bits} * 1\text{LSB})}{\frac{(1.4286*\text{lvdd}) +/- (25\text{bits} * 1\text{LSB})}{1023}}\right] +/- \text{DNL} +/- \text{INL}$$

This equation is only applicable for PAD_A2D_DATA or external data input. The equation needs to be modified for TX_TX_CP_div2 and TX_RX_CP_div2 because the equation does not include any error that is introduced by the divider and the buffer in TX analog module.

0.1269 is the result of dividing 0.2 (minimum input) by 1.575 (typical LVDD), and 1.4286 is the result of dividing (2.45 - 0.2) by 1.575. The 1LSB shown in the equation varies with LVDD. The 1LSB value vs. LVDD is listed in the following table.

Table 35-8. GPADC 1LSB Table

lvdd (V)	1LSB (mV)
1.54	2.148
1.575	2.197
1.61	2.248

35.4 General Purpose ADC Timing Diagram

Figure 35-2 shows the timing diagram for normal operation.

GPADC needs at least 5 uS to power up. GPADC takes at least 10 uS to settle after receiving the enable signal or MUX_SELECT[1:0] from A2DIGL module. Timings are controlled by A2DIGL module for both normal and deep sleep mode. Conversion starts after the rise edge of A2DI_CONVERT signal.

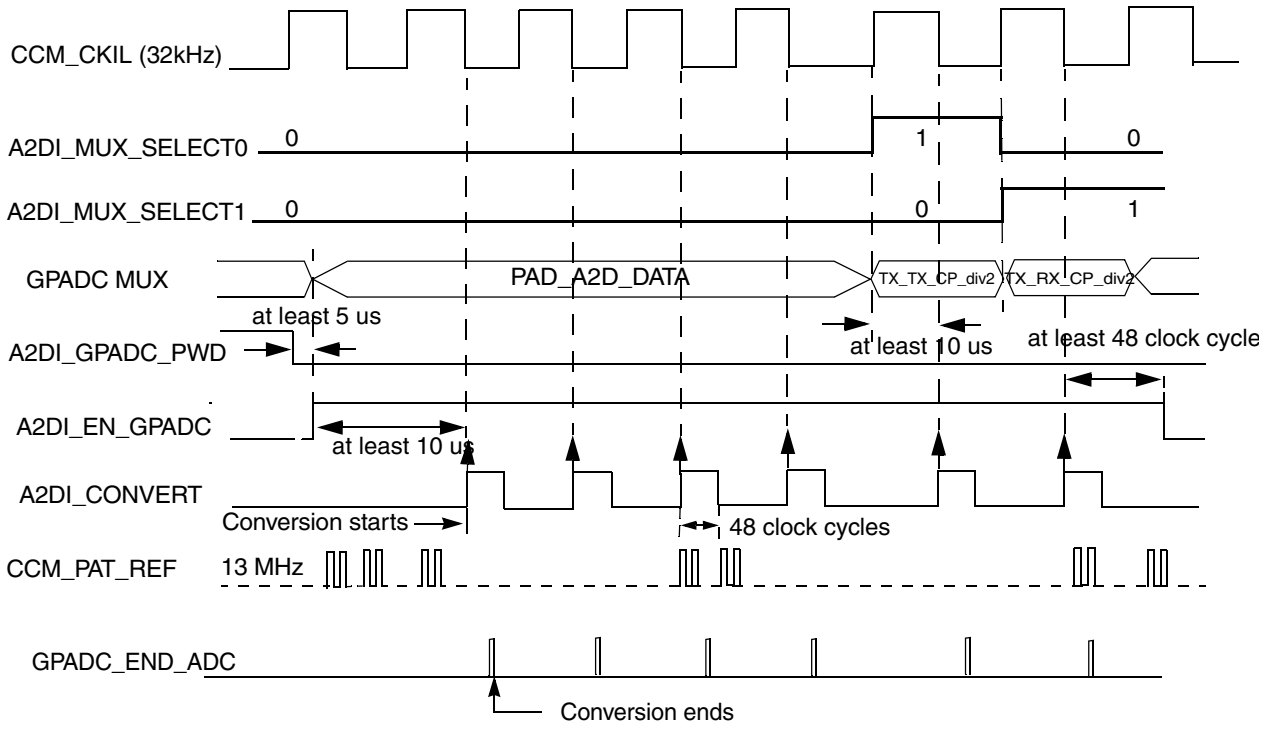


Figure 35-2. GPADC Normal Mode Operation

Figure 35-3 shows a timing diagram for GPADC Deep Sleep Mode operation.

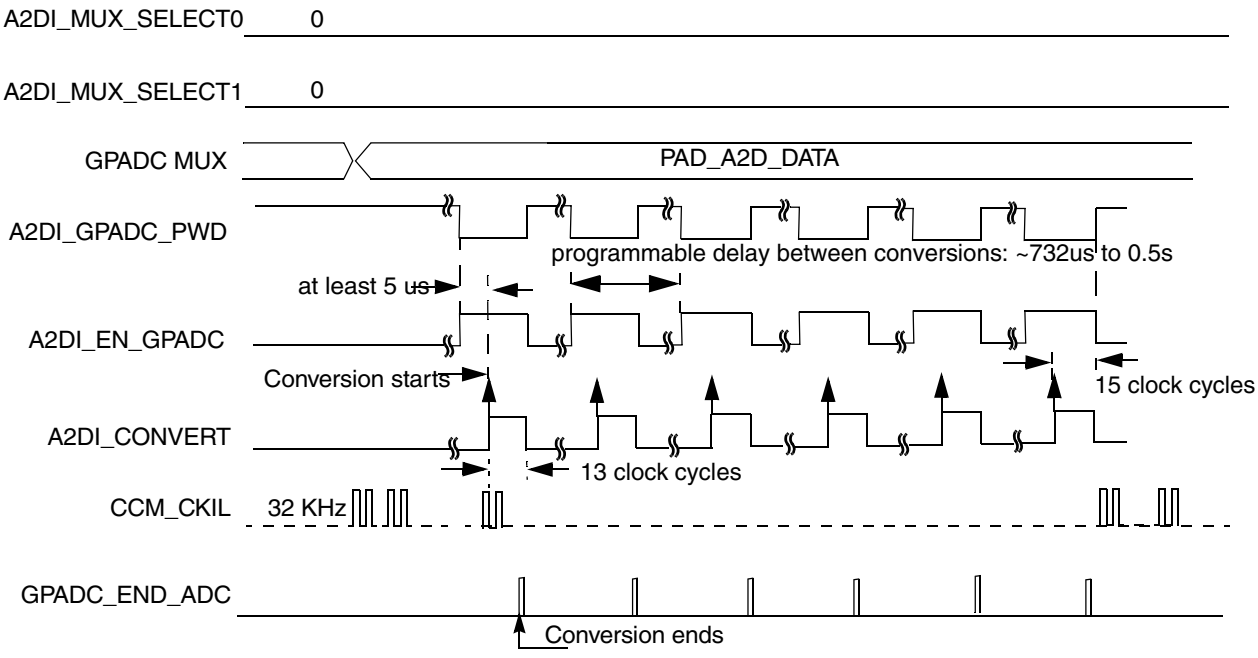


Figure 35-3. GPADC Deep Sleep Mode Operation

Figure 34-4 is the diagram of GPADC test mode operation. The test mode occurs when the control bit TST is set to 1. This is defined in A2DIGL spec.

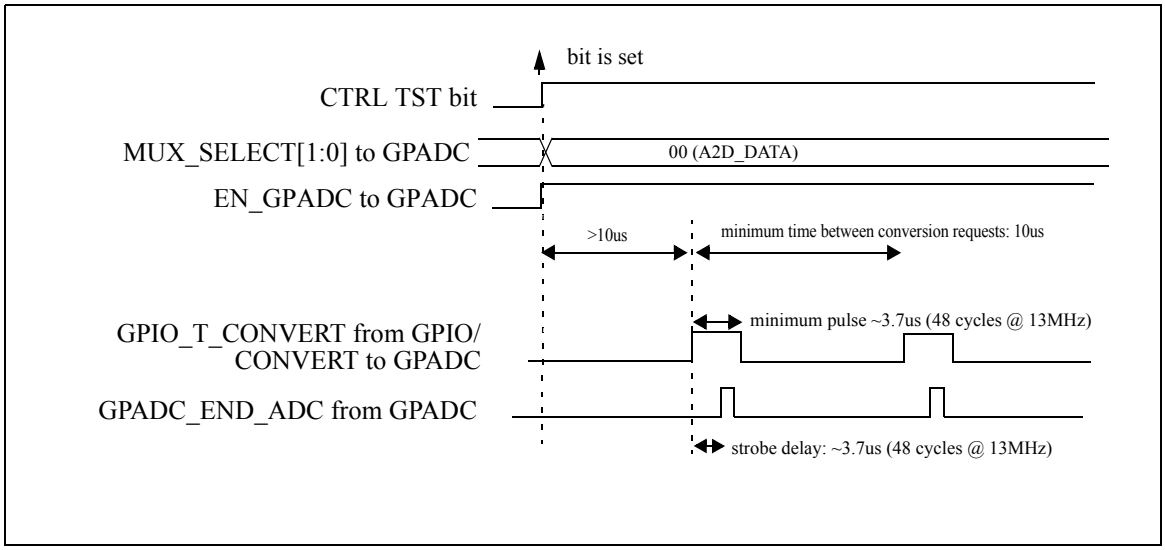


Figure 35-4. GPADC Test Mode Operation

Chapter 36
Mixed-Signal Control Interface (A2DIGL)

Revision History Table

Revision	Date	Author	Changes
1.0	04/18/02	Khursheed Hassan	Initial Creation for LTS by porting the A2DIGL LT spec with the followig changes; - DSPhl12665 : Moved DP_FN_DACb from Group 1 to Group 7. - DSPhl13874 : Increase in size of the Group 3 PHASEADxb regis ter to 10 bits. Additional 2 MSB bits are placed in a the location [37:36] within the same group. - DSPhl13785 : Support BBP Edge transmit. Input from GPIO “gpio_sckb_clk” is muxed out to the sckb port of the BBP. The mux control is provided by an additional bit TxSEL3 in the MUXCTL register. Updated Fig. 36-6 - DSPhl12803 : The GPADC delay timer logic is moved from the PAC into the A2DIGL along with the removal of the associated ports. NEEDS FIXING FIG 36-17/18/19/20 along with the new transmit timing explanation. - DSPhl13914 : Fixed Tunec Fig 36-2 - Removed, renamed some scan signals, removed acc_ref[11:1] bits.
1.1	05/19/02	Khursheed Hassan	- DSPhl14017 : Replace SPI GROUP5 bit “a2di_highv_offset” w ith “a2di_soft_sat”. - DSPhl14275 : Gate off the MQSPI signal to the ALGAE when MQSPI writes are being performed on A2DIGL. DCADAPT SPI writes to ALGAE will not be affected. Added a SPI bit to GROUP1 bit 16 called “ALG_DIS”“. - Removed GROUP1 DAC_6_TST SPI bit. - Fixed Figs. 36-17/18/19/20. Some additions to Fig. 36-22
1.2	06/18/02	Khursheed Hassan	- Typo correction to PWRUNFLT& update to CTRL register

Revision History Table

Revision	Date	Author	Changes
1.3	07/18/2002	K.Hassan	- DSPhl14983 implemeted, along with the new requirement for the control of CNVE and assertion of a2di_pwd. Figures added to the GPADC section to explain this. Removed the GPD counter timing from the TX timing diagrams.. **NOTE : There is a change of functionality for some of the A2DIGL's GPADC related signals, along with the addition of new bits in CTRL register for Neptune LTS.**
1.4	08/14/2002	K.Hassan	MTR register definition modified.
1.5	10/28/2002	K. Hassan	**Additions for LTE/ULS** - DSPhl15121 : The GROUP 5 SPI bit RXCLK_SEL added along with its associated port a2di_rxclk_sel. This is only used in LTE/ULS - DSPhl15339/DSPhl15581 : The OSC26M_DISABLE (a2di_osc26m_disable) is now used to disable PAC. No functional change done for this DDTS. Only the signal definition has changed in the spec. - DSPhl15495/DSPhl15519 : Requirement to have gpadc_a2d_out to be atleast 3 CKIL cycles wide has been removed from the spec.
1.6	04/29/03	Keith Tilley, Shannon Osgood	Updated per DDTS# DSPhl15633. Changed MNCP_INV bit from group 1 bit 31 to group 5 bit 0.
1.7	05/12/03		Initial release of LTE only specification.

36.1 Introduction

The A2DIGL is a digital module intended for control of mixed signal modules. The A2DIGL is divided into two parts: One part contains SPI Interface control; while the other part contains the MCU-based GPADC digital control.The mixed-signal new modules are: REGUL, GPADC, TOSW, and TUNEC. The mixed-signal re-used modules (from MAGIC-LV, GCAP, and Tomahawk) are: PAC, TX, TRSYNT, DCADAPT, RxSDG, RxAFE, and RXCPROC. The re-used logic ensures a high degree of backward compatibility with the Patriot program software.

36.2 A2DIGL Block Diagram

Figure 36-1, A2DIGL Block diagram shows the functions within the A2DIGL and gives a broad overview of the interconnections between the Neptune A2DIGL and the Seaweed, Algae, and Neptune modules.

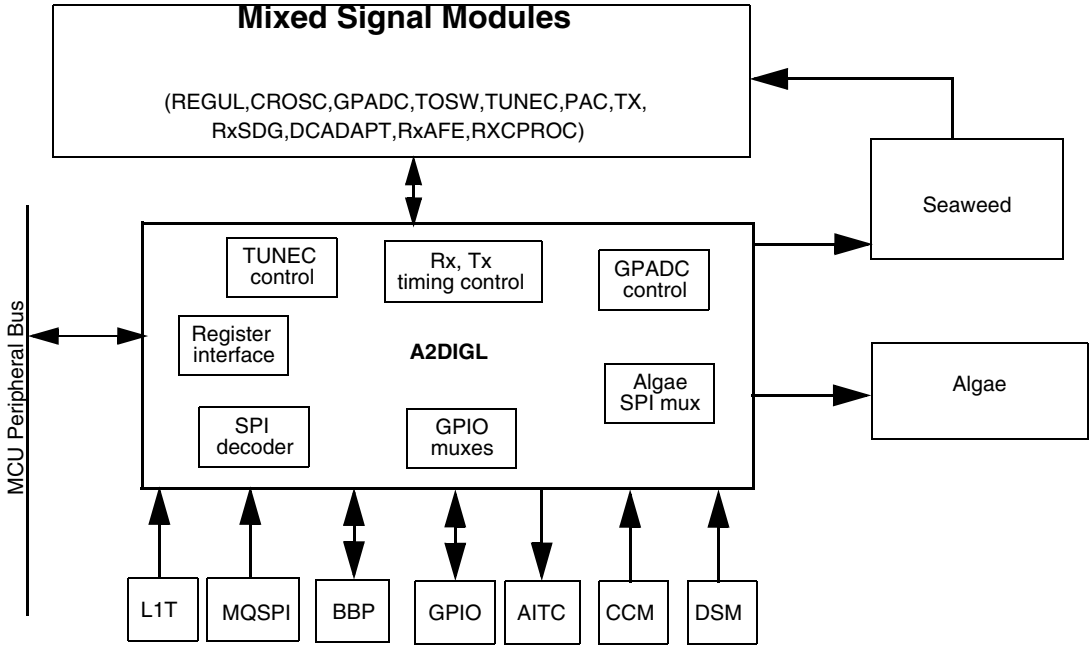


Figure 36-1. A2DIGL Block Diagram

NOTE:

TOSW is in PAD.

The major functions within A2DIGL are described in more detail in the following sections of this chapter:

- “TUNEC control” is described in Section 36.4.1.1, “RC Tuning Element Control Interface,” on page 36-15.
- “Rx, Tx timing control”, and “Algae SPI mux” is described in Section 36.5.1.5, “Receive and Transmit Timing Sequences and Algae SPI Multiplex Operation,” on page 36-28.
- “GPADC control” is detailed in Section 36.5.2, “A2DIGL and General Purpose ADC Interface,” on page 36-74.

- The “Register interface” is described in Section 36.4.1.1.1, “Tuning Element Control Registers,” on page 36-17, Section 36.4.1.2.2, “CROSC Registers,” on page 36-20, Section 36.5.1.8, “Receive and Transmit Modules Interface Register Summary,” on page 36-48, and Section 36.5.2.4, “GPADC Interface Register Summary,” on page 36-82.
- The “SPI decoder” operation is described in Section 36.5.1.4, “SPI Decoder Operation Description,” on page 36-27, and the programming model is defined in Section 36.5.1.10, “SPI Interface Bit Groups,” on page 36-54.
- The “GPIO muxes” for alternate input/outputs through GPIO are described in Section 36.5.1.1, “Support for bypass and control modes,” on page 36-24.

36.3 A2DIGL Module Pin List

Table 36-1 lists all input/output pins for the A2DIGL.

Table 36-1. A2DIGL Module Pin List

Pin Name	Direction	Description
ANATEST/TCM Interface Signals		
tcm_mbist_en	Input	MBIST enable signal
a2di_acc_ref	Output	ACC_REF is used as the tcm_anatest_en0 signal.
a2di_tst_dc[5:0]	Output	Anatest Control Mux1 and Mux2
a2di_gpadc[3:0]	Output	Test bits for GPADC
a2di_reg[7:0]	Output	Test bits for regulators
a2di_det_test_mode	Output	Select DET_FLAG_OBS output of RXODandDG for observation --To anatest module
a2di_rxafe[4:0]	Output	Test bits for Receiver Analog Front End--To anatest module
a2di_voc[1:0]	Output	Test bits for Voice Codec & Interface--To anatest module
a2di_tunec[7:0]	Output	Test bits for Tuning Circuit--To anatest module
a2di_cmon_camp[4:0]	Output	Test bits for System Clock Monitor--To anatest module
a2di_tx_syn[5:0]	Output	Test bits for Transmitter and Synthesizer--To anatest module
a2di_pac[4:0]	Output	Test bits for Power Amplifier Control DAC/ADC--To anatest module
Battery Save Signals		
a2di_bsave	Output	Receiver power save mode. This goes to RXAFE, TUNEC, and REGUL
a2di_bsavetx	Output	TX power save mode. This goes to TUNEC
a2di_bsaverx	Output	RX power save mode.
a2di_bsaverxafe	Output	Receiver SDM Power Save Mode. This goes to RXAFE and RXSDG
a2di_bsavetrsynt	Output	TRSYNT power save mode signal. This goes to TX
a2di_bsaverxclk	Output	Receiver clock power save mode signal. This goes to RXAFE
a2di_bsaverxcproc	Output	RXCPROC battery save mode signal. This goes to RXCPROC
a2di_bsavepac	Output	PAC Power Save Mode. This goes to TX, PAC, and RXAFE
a2di_osc26m_disable	Output	This signal is used to disable the PAC_ANA circuits.
BBP Interface Signals		
bbp_stdb	Input	Serial transmit data from BBP. Connects to BBP's stdb pin.
bbp_stdb_oe	Input	Output enable for bbp_stdb
a2di_sckb	Output	Serial transmit clock to BBP

Table 36-1. A2DIGL Module Pin List (Continued)

Pin Name	Direction	Description
a2di_sc0b	Output	Serial receive clock to BBP
a2di_sc1b	Output	Serial receive framing signal to BBP
a2di_srdb	Output	Serial receive data to BBP
a2di_dmcsg	Output	DMCS signal for BBP glue logic
a2di_magen	Output	Selects Magic SSI configuration for BBP
CCM Signals		
ccm_ckil	Input	32 KHZ clock
ccm_pat_ref	Input	13 MHZ clock
CROSC Interface Signals		
osc26m_peak[1:0]	Input	Indicates current OSC amplitude level
a2di_osc26m_agc[5:0]	Output	Programs the oscillator amplitude AGC DAC
DCADAPT Interface Signals		
dcad_sce	Input	Algae SPI chip enable from DCADAPT
dcad_sdata	Input	Algae SPI data from DCADAPT
dcad_sclk_out	Input	Algae SPI clock from DCADAPT
a2di_fastdigitaladapt	Output	When sets high the receiver section will take control of the Algae SPI lines.
a2di_endfastadapt	Output	Indicates that DCADAPT should perform it's final SPI write to Algae
a2di_agc_high	Output	Input to DCADAPT to use in its Algae SPI writes
a2di_lna_off	Output	Input to DCADAPT to use in its Algae SPI writes
a2di_fg[2:0]	Output	Gain of the digital low pass filter in digital adapt mode.
a2di_dcapol	Output	Selects polarity used by DCADAPT
DSM Signals		
standby	Input	When set to 1 enters deep sleep mode. From DSM STBY_CAMP pin.
GPADC Interface Signals		
gpadc_a2d_out[9:0]	Input	10-bit digital data signal generated by the analog to digital convertor.
gpadc_end_adc	Input	Signalling the end of each data conversion.
a2di_convert	Output	To mark the beginning of the conversion of the GPADC.
a2di_mux_select[1:0]	Output	Select signal for the GPADC mux.
a2di_pwd	Output	Power Down for GPADC

Table 36-1. A2DIGL Module Pin List (Continued)

Pin Name	Direction	Description
a2di_en_gpadc	Output	Enable digital logics for GPADC.
GPIO Interface Signals		
gpio_stdb	Input	Serial transmit data. Alternate input from GPIO
gpio_tx_clk	Input	270.833kHz serial transmit bit clock. Alternate input from GPIO
gpio_srdx	Input	Serial receive data. Alternate input from GPIO
gpio_sclk_out	Input	Serial receive data clock. Alternate input from GPIO
gpio_sdfs	Input	Serial receive data framing signal. Alternate input from GPIO
gpio_rx_acq	Input	RX_ACQ receive acquire signal. Alternate input from GPIO
gpio_dmcs	Input	DMCS transmit enable signal. Alternate input from GPIO.
gpio_spi_data	Input	Serial SPI data input line. Alternate input from GPIO.
gpio_spi_clk	Input	SPI Clock input line. Alternate input from GPIO.
gpio_ce	Input	SPI Decoder chip select. Alternate input from GPIO.
gpio_ce_a	Input	SPI chip select for Algae. Alternate input from GPIO.
gpio_t_convert	Input	Convert signal for GPADC acquisition in test mode
a2di_a2d_sync	Output	Reset Seaweed Mux. To A2D_SYNC pin
a2di_algae_cs	Output	SPI chip select output for Algae. To RF_CS pin
a2di_algae_clk	Output	SPI clock output for Algae. To RF_CLK pin
a2di_algae_data	Output	SPI data output for Algae. To RF_DATA pin
a2di_rx_en_out	Output	Receive enable output to Algae. To RX_EN_OUT pin
a2di_gsm_dcs	Output	Selects GSM or DCS band. Output to GPIO GSM*_DCS pin
ITC Signals		
a2di_irq_b	Output	Interrupt Request to the MCU interrupt controller
L1T Signals		
l1t_rx_acq	Input	RX_ACQ receive acquire signal from L1T
l1t_dmcs	Input	DMCS transmit enable signal from L1T
MQSPI Signals		
mqspi_spi_data	Input	Serial SPI data input line from MQSPI.
mqspi_spi_clk	Input	SPI Clock input line from MQSPI.
mqspi_ce	Input	SPI Decoder chip select from MQSPI.

Table 36-1. A2DIGL Module Pin List (Continued)

Pin Name	Direction	Description
mqspi_ce_a	Input	SPI chip select for Algae from MQSPI.
PAC Interface Signals		
pac_cdet_out	Input	Crossover activity detect from PAC
pac_dig_pwrunflt_spi[9:0]	Input	Output word of the AOC ACC input minus DC_OFFSET (from PAC module)
pac_pwrdac[9:0]	Input	Value of the AOC D/A input. (from PAC module)
pac_errorsig[10:0]	Input	Difference between power ramp and ADC input values (from PAC module)
pac_dn_sig	Input	The rising edge of this signal from the PAC module indicates the end of the transmission burst.
a2di_dmcs	Output	DMCS signal for TX and PAC modules.
a2di_dac_pac_tst	Output	testmode for PAC's DAC. To PAC dac_test pin.
a2di_aoc_fbw[1:0]	Output	Selects the bandwidth of the analog filter which follows the AOC D/A.
a2di_sat_step[4:0]	Output	AOC negative slope when saturated.
a2di_sat_dly[13:0]	Output	Sets the number of PAC clocks from the rising edge of DMCS to enabling saturation detection.
a2di_sat_th[9:0]	Output	Threshold of error between detected power and ramp shape waveform which activates saturation detection.
a2di_dazero	Output	Output smoothing filter control.
a2di_pwr[9:0]	Output	Final power level desired for systems which use the internal look up table for PA ramp shaping.
a2di_aoc_gain[1:0]	Output	Selects the gain of the differential amplifier in front of the AOC A/D converter.
a2di_sat_en	Output	Logic high activates saturation detection. Logic low disables this feature.
a2di_ofs[7:0]	Output	Offset power level for systems which use the internal look up table for PA ramp shaping.
a2di_est_dly[13:0]	Output	Delay from rising edge of DMCS to the end of the estimation stage.
a2di_ramp_dly[13:0]	Output	Delay from rising DMCS to the start of the ramp up in look up mode.
a2di_act_th[9:0]	Output	Threshold of detected power to insure closed loop control.
a2di_err_gain1[3:0]	Output	Error signal attenuation at accum input before BW_DLY.
a2di_propgain1[3:0]	Output	Error signal attenuation at accum output before PROP_DLY.
a2di_propgain2[3:0]	Output	Error signal attenuation at accum output before PROP_DLY.
a2di_gsm_init[9:0]	Output	Slope of the initial linear AOC drive ramp in GSM mode.

Table 36-1. A2DIGL Module Pin List (Continued)

Pin Name	Direction	Description
a2di_aoc_max[9:0]	Output	Maximum permitted value of the AOC D/A.
a2di_cdet_dly[17:0]	Output	Delay from rising TX_KEY to where cross detect is passed to the PAC_CDET_OUT pin.
a2di_ssi_dly[13:0]	Output	Delay from rising DMCS to 1st TX_CLK.
a2di_init_dly[13:0]	Output	Delay from rising edge of DMCS to the start of the estimation stage.
a2di_div_dly[13:0]	Output	Delay from rising DMCS to AOC accumulator clock division is switched from ACC_DIV1 to ACC_DIV2.
a2di_acc_init[9:0]	Output	Estimation state value for the D/A.
a2di_aoc_hold	Output	Disables clock for AOC accumulator forcing a hold state when DIV_DLY expires.
a2di_prop_dly[13:0]	Output	Delay from rising DMCS to proportional gain switch from PROPGain2.
a2di_bw_dly[13:0]	Output	Delay from rising DMCS to when error loop gain is switched.
a2di_pac_dly[11:0]	Output	Sets the number of PAC clocks from the rising edge of DMCS to triggering the reading the PACII DC offset.
a2di_dn_dly[13:0]	Output	Sets the number of PAC clocks from the falling edge of DMCS to power down of the TX section.
a2di_aoc_force[9:0]	Output	Value to which the AOC is forced if AOC_MANUAL is programmed high.
a2di_low_batt_en	Output	If set, the AOC_OUT pin will be grounded when LOW_BATT* pin is low.
a2di_aoc_manual	Output	If high, AOC_FORCE is loaded to AOC D/A.
a2di_ramp_ref[5:0]	Output	Sets the division ratio from the PAC clock to the look up table ramp clock.
a2di_ramp_sel[2:0]	Output	Selects ramp.
a2di_proff	Output	If programmed high then the proportional gain branch is disabled.
a2di_err_gain2[3:0]	Output	Error signal attenuation at accum input after BW_DLY.
a2di_act_en	Output	Logic high activates activity detection and initial linear ramping. Logic low deactivates this feature.
a2di_act_sel	Output	Logic high selects the digital activity comparator, logic low selects the analog activity comparator.
a2di_det_highv	Output	If DETECT_IN < 2.5V, this bit should be set. PAC signal
a2di_soft_sat	Output	Enables the functionality of AOC_MAX, when set to 1. PAC signal
a2di_backward	Output	Applies PWR_RAMP word directly to AOC DAC input.
a2di_cdma_dn	Output	If high then when DMCS goes from high to low the feedback loop will revert to ERRGain1, and PROP_Gain1
a2di_ramp_dn_dly[13:0]	Output	Sets time from DMCS falling edge to ramp down

Table 36-1. A2DIGL Module Pin List (Continued)

Pin Name	Direction	Description
a2di_adi_ref[6:0]	Output	Sets the division ratio from the PAC clock to the internal A/D clock
a2di_paclc[3:0]	Output	PAC limit cycle correction value. (NOTE: This is not currently used by the PAC)
a2di_det_flt	Output	Selects PAC detector filter bandwidth
REGUL Interface Signals		
a2di_rt_reg_en	Output	Enable for RX/TX regulator
a2di_s_reg_en	Output	Enable for Synthesizer regulator
a2di_aud_reg_en	Output	Enable for Audio regulator
RXAFE Interface Signals		
a2di_sdm_z	Output	If high then some of the zeroes of the sigma delta modulator are moved to non-zero frequency. If low then all of the zeros of the sigma delta modulator are located at DC.
a2di_fastbiascharge	Output	Fast Bias Charge for RX or TX.
a2di_rxclk_sel	Output	Selects the clock source for the RxAFE.
RXCPROC Interface Signals		
rxcproc_srdx	Input	Serial receive data from RxCPROC. Connects to RXCPROC sdrx pin.
rxcproc_sclk_out	Input	Serial data clock from RxCPROC. Connects to RXCPROC sclk pin.
rxcproc_sdfs	Input	Serial data framing from RxCPROC. Connects to RXCPROC sdfs pin.
rxcproc_thp1_pulse	Input	From RxCPROC to generate MUX select signal
a2di_tol[1:0]	Output	Droop digital filter setting.
a2di_fhp1[2:0]	Output	DC adapt setting for phase1 in DCR mode.
a2di_fhp2[3:0]	Output	DC adapt setting for phase2 in DCR mode.
a2di_fhp3[3:0]	Output	DC adapt setting for phase3 in DCR mode.
a2di_thp1[2:0]	Output	Duration of DC adapt phase1 in DCR mode.
a2di_thp2[2:0]	Output	Duration of DC adapt phase2 in DCR mode.
a2di_hdfo	Output	Hold DC offset value in non DCR mode.
a2di_ma_select[1:0]	Output	Selects GSM, EDGE, or DCR wide for receive coprocessor
a2di_sign[1:0]	Output	Control four complex quadrature mixer combination.
a2di_gainada[6:0]	Output	Gain adjust a for the complex quadrature mixer.
a2di_gainadb[6:0]	Output	Gain adjust b for the complex quadrature mixer.
a2di_phaseada[7:0]	Output	Phase adjust a for the complex quadrature mixer.

Table 36-1. A2DIGL Module Pin List (Continued)

Pin Name	Direction	Description
a2di_phaseadb[9:0]	Output	Phase adjust b for the complex quadrature mixer.
a2di_testbyp	Output	Bypass filters and mixers.
a2di_iqswap	Output	Q bits sent first if IQSWAP is high otherwise I bits are sent first.
a2di_2x_sel	Output	Select the output rate.
a2di_dbw[2:0]	Output	Digital Filter bandwidth select.
a2di_fbyp	Output	To allow bypass of the DBW filter.
a2di_dif[6:0]	Output	Selects the digital IF local oscillator.
a2di_bbiqlen[3:0]	Output	BBP interface setting.
a2di_rx_acq	Output	RX_ACQ signal for RXCPROC module
RXSDG Interface Signals		
rxsdg_det_flag_b	Input	Saturation detector flag signal. A logic low indicates the sigma-delta input level is excessive.
a2di_det_reset	Output	Used to clear the saturation detection signal DET_FLAG
a2di_det_lvl[2:0]	Output	The programming input for the level comparison threshold of the amplitude detector on the sigma delta output.
a2di_sdd_en	Output	If programmed high then a psuedo random dither is enabled on the sigma delta modulator. This signal also goes to the RXAFE
a2di_dith_det_lvl	Output	Dither detect level threshold for sigma delta converter
a2di_dith_lvl[1:0]	Output	Selects the differential dither levels for the sigma delta converter when the level selected by DITH_DET_LVL is not exceeded. If DITH_DET_LVLis exceeded then the dither is set to zero.
a2di_dith_reset	Output	Used to reset the dither circuit's random number generator
TUNEC Interface Signals		
tunec_out[6:0]	Input	Tuning code from TUNEC module
tunec_code_valid	Input	Tuning code valid signal from TUNEC module
a2di_tunec_res	Output	Reset pin for the tuning circuit.
a2di_tcode[6:0]	Output	Tuning code to mixed signal modules
TX Interface Signals		
tx_tx_clk	Input	270.833kHz bit clock from Tx. Connects to TX's TX_CLK pin.
a2di_stdx	Output	Serial transmit data to Tx. Connects to TX's SDTX pin.
a2di_dp_ofs[9:0]	Output	dual_port programmable offset
a2di_dp_mode	Output	Select source for dual_port.

Table 36-1. A2DIGL Module Pin List (Continued)

Pin Name	Direction	Description
a2di_dp_on_offb	Output	Enable the dual_port system.
a2di_dp_fn_dacb	Output	Selects which path is delayed by DP_DELAY.
a2di_dp_delay[1:0]	Output	datapath delay in 230ns steps.
a2di_dac_9_tst	Output	testmode for TX's 9bit DAC. Connects to TX dp_test pin.
a2di_dp_tst_in[9:0]	Output	testmode for dual_port DAC.
a2di_dp_gain[5:0]	Output	dual_port programmable gain.
a2di_trkl_sel[2:0]	Output	alterUP tricle current to charge pumps.
a2di_cptest[2:0]	Output	Places the charge pumps into test mode.
a2di_mod_inv	Output	Inverts the output of the differential encoder.
a2di_mncp_inv	Output	Select sink or source for charge pump.
a2di_fn_mode	Output	Selects source of data for Frac N
a2di_dp_inv	Output	Selects sense for data in dual_port.
a2di_od[3:0]	Output	divider ratio for Oversampling clock.
a2di_acc2	Output	Used with ACC0 to select the number of accumulators used by the synthesizer
a2di_acc0	Output	Used with ACC2 to select the number of accumulators used by the synthesizer
a2di_tx_vco_div	Output	digital divide by 2 before going into FracN. Connects to TX's tx_vco_div2 pin.
a2di_num[23:0]	Output	The numerator for the 24 bit fractional N divider system.
a2di_n[4:0]	Output	Programming input for the fractional N
a2di_a[2:0]	Output	Programming input for the fractional N
a2di_tr_sel	Output	Selects the charge pump in use. The CP not selected is floating.
a2di_pd_step	Output	Input to the GSM pre-distortion waveform generator
a2di_cp_lv	Output	Sets the synthesizer charge pump output voltage level
a2di_x3d0[15:0]	Output	The GSM predistortion coefficient for the third derivative of the modulation frequency.
a2di_x3d1[15:0]	Output	The GSM predistortion coefficient for the third derivative of the modulation frequency.
a2di_x3d2[15:0]	Output	The GSM predistortion coefficient for the third derivative of the modulation frequency.
a2di_x3d3[15:0]	Output	The GSM predistortion coefficient for the third derivative of the modulation frequency.

Table 36-1. A2DIGL Module Pin List (Continued)

Pin Name	Direction	Description
a2di_x3d4[15:0]	Output	The GSM predistortion coefficient for the third derivative of the modulation frequency.
a2di_x3d5[15:0]	Output	The GSM predistortion coefficient for the third derivative of the modulation frequency.
a2di_x2d0[19:0]	Output	The GSM predistortion coefficient for the second derivative of the modulation frequency.
a2di_x2d1[19:0]	Output	The GSM predistortion coefficient for the second derivative of the modulation frequency.
a2di_x2d2[19:0]	Output	The GSM predistortion coefficient for the second derivative of the modulation frequency.
a2di_x2d3[19:0]	Output	The GSM predistortion coefficient for the second derivative of the modulation frequency.
a2di_x2d4[19:0]	Output	The GSM predistortion coefficient for the second derivative of the modulation frequency.
a2di_x2d5[19:0]	Output	The GSM predistortion coefficient for the second derivative of the modulation frequency.
a2di_x1d0[17:0]	Output	The GSM predistortion coefficient for the first derivative of the modulation frequency.
a2di_x1d1[17:0]	Output	The GSM predistortion coefficient for the first derivative of the modulation frequency.
a2di_x1d2[17:0]	Output	The GSM predistortion coefficient for the first derivative of the modulation frequency.
a2di_x1d3[17:0]	Output	The GSM predistortion coefficient for the first derivative of the modulation frequency.
a2di_x1d4[17:0]	Output	The GSM predistortion coefficient for the first derivative of the modulation frequency.
a2di_x1d5[17:0]	Output	The GSM predistortion coefficient for the first derivative of the modulation frequency.
Miscellaneous Interface Signals		
gpio_sckb_clk	Input	The clk signal to the BBP muxed out through the A2DIGL
a2di_sto_en	Output	While high, the 26MHz clock will be enabled on the TRKG_OSC_OUT pin. This enable signal goes directly to the TRKG_OSC_OUT pad.
IP Bus Signals		
ips_addr[11:2]	Input	
ips_rwb	Input	
ips_wdata[31:0]	Input	

Table 36-1. A2DIGL Module Pin List (Continued)

Pin Name	Direction	Description
ips_byte_en_31_24	Input	
ips_byte_en_23_16	Input	
ips_byte_en_15_8	Input	
ips_byte_en_7_0	Input	
ips_module_en	Input	
ipg_clk	Input	
ipg_async_hard_reset_b	Input	Reset signal
a2di_ips_rdata[31:0]	Output	
a2di_ips_xfr_err	Output	
Scan Test Signals		
ipt_scan_mode	Input	Scan mode
ipt_scan_enable	Input	Scan mode
ipt_si[1:0]	Input	Scan mode scan chain input signal
ipt_so[1:0]	Output	Scan mode output signals

36.4 Mixed Signal Interfaces Summary

Listed below are the control interface blocks contained in the A2DIGL module:

- RX and TX Synthesizers (TRSYNT)
- RX ADC and Channel Filter (RxAFE, RXCPROC)
- RX DC Offset Correction (DCADAPT)
- Receiver Overload Detection and Receiver Sigma-delta Modulator Dither Generation (RxSDG)
- TX GMSK Modulation DAC (TX)
- PA Control (PAC)
- Tracking Clock (TOSW)
- RC time-constant tuning element (TUNEC)
- Regulators (REGUL)
- General Purpose ADC (GPADC)
- 26MHz Reference Oscillator (CROSC)

36.4.1 A2DIGL New Modules Summary

This section summarized the A2DIGL interfaces which are new, and not re-used in any way from MAGIC-LV, GCAP, or Tomahawk. Those new interfaces are:

- RC Tuning Element Control Interface (TUNEC)
- 26MHz Reference Oscillator (CROSC)
- Regulators Control (REGUL)
- Tracking Clock Control (TOSW)
- General Purpose Analog to Digital Converter (GPADC)

NOTE:

The GPADC is a new module since the interface logic has been re-defined to the MCU bus, and is no longer controlled by a SPI interface decoder.

36.4.1.1 RC Tuning Element Control Interface

The tuning code will be updated by the A2DIGL immediately following every Group 5 SPI write during TX. The SPI write will initiate the timing sequence shown below. The TUNEC PWD pin will be pulled low as will the TUNEC_RES in. After seven cycles of TUNEC's 13MHz/16 clock, the tuning code (tunec_out[6:0]) will be valid. One cycle later, TUNEC_CODE_VALID will go high, to signal A2DIGL to latch in the new tuning code value, store it in the TCODE register, and output it to the TX and PAC modules. The TCODE register will only be updated during TX, and it will not be updated during RX. A minimum of 3 cycles later, the A2DIGL will pull TUNEC_RES high. PWD will stay low until BSAVETX goes high.

The A2DIGL also provides a TTEST register which allows the tuning code output from the TUNEC to be bypassed in favor of a user-programmed value. While the BYP bit is set in the TTEST register, the tuning code output by the TUNEC will be stored in the TCODE memory-mapped register but will not be output to the TX and PAC modules. Instead, the TST[6:0] bits in the TTEST memory-mapped register will be output to the TX and PAC.

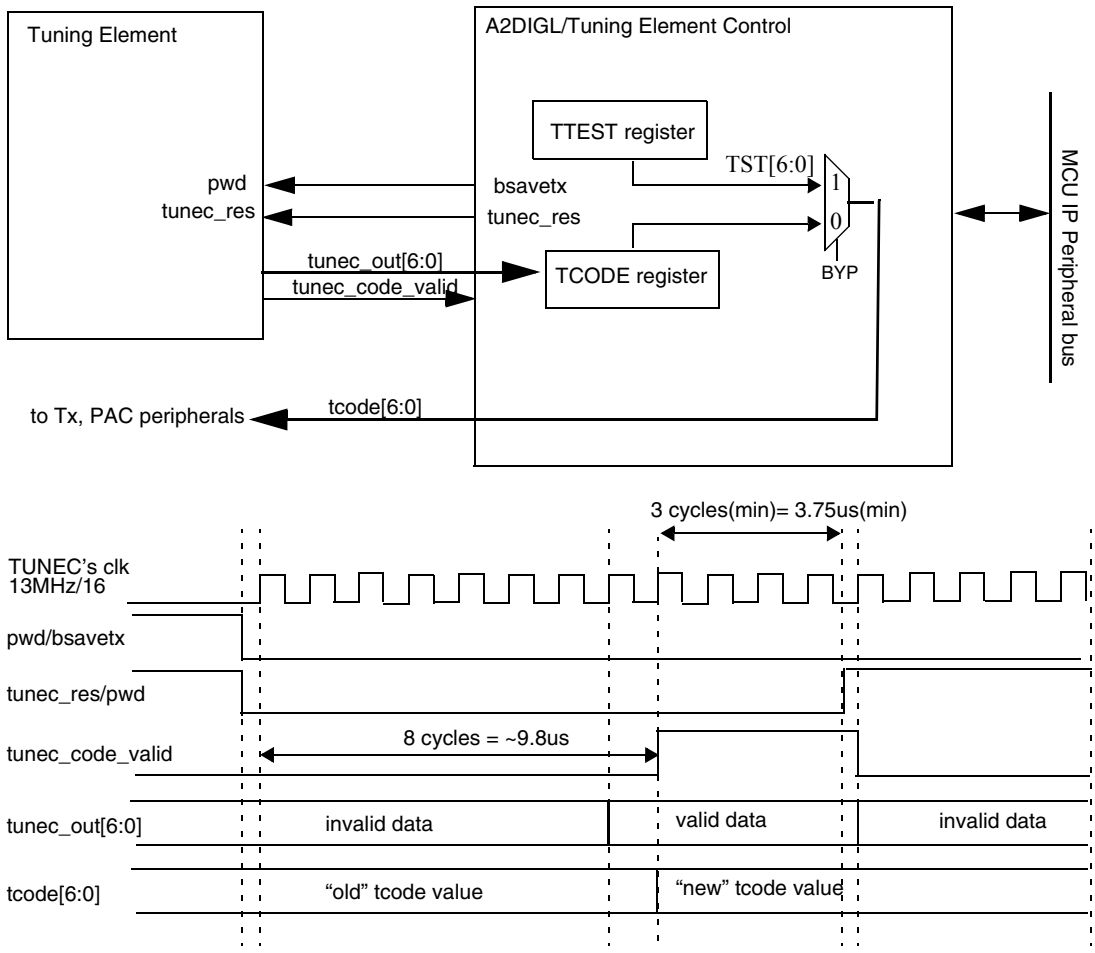


Figure 36-2. RC Tuning Element Control Interface and Timing

36.4.1.1.1 Tuning Element Control Registers

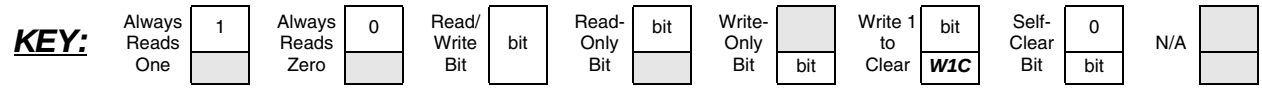


Table 36-2. Tuning Element Control Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCODE $2484_6038	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	TCODE[6:0]						
	W																
TTEST $2484_603C	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	BYP	TST[6:0]						
	W																

36.4.1.1.2 Tuning Element Control Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the Tuning Element control registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and is cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

TCODE TCODE Register **Addr $2484_6038**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
										TCODE[6:0]						
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-3. TCODE Description

Name	Description	Settings
TCODE[6:0] Bits 6-0	**TCODE —** Tuning code register: 7-bit read-only register which contains the tuning code output from the tuning element.	

TTEST TUNEC Test Register **Addr $2484_603C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									BYP	TST[6:0]						
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-4. TTEST Description

Name	Description	Settings
TST[6:0] Bits 6-0	**TST —** If BYP is set, the value written to TST[6:0] will be used for the tuning code.	
BYP Bits 7	**BYP —** This bit, when set, enables TUNEC bypass operation.	

36.4.1.2 CROSC Control Interface

The A2DIGL provides the programming interface for the 26MHz crystal oscillator module. This consists of several SPI bits, and one register, OSC26M, which allows the oscillator's peak status bits to be read. The SPI bits and registers are described below. See Section 30.7, "CROSC- 26 MHz Reference Oscillator (analog)," for more information.

BYP

36.4.1.2.1 CROSC SPI bits

The OSC26M_AGC[5:0] SPI bits program the oscillator's DAC.

36.4.1.2.2 CROSC Registers

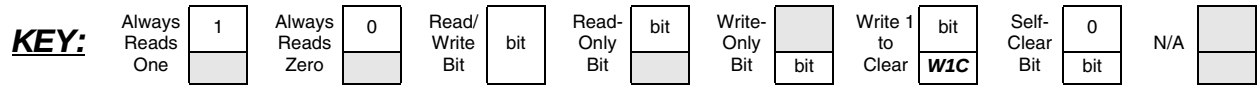


Table 36-5. MCU Peripheral Bus Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSC26M $2484_6040	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PEAK[1:0]	
	W																

36.4.1.2.3 CROSC Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the oscillator control register. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and is cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

OSC26M OSC26M Register **Addr $2484_6040**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
															PEAK[1:0]	
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-6. OSC26M Description

Name	Description	Settings
PEAK[1:0] Bits 1-0	**PEAK** - Indicates the peak amplitude of the oscillator. This value is updated at the CKIL clock rate.	10 - Amplitude too high; trim lower 00 - Amplitude in desired range' 01 - Amplitude too low; trim higher 11 - Invalid state

36.4.1.3 Regulators Control

Programmable control of the on-chip mixed-signal module regulators is provided in the form of SPI bits within the SPI control interface

The SPI bits are defined as follows:

- RT_REG_EN - this bit may be cleared to disable the regulator for the RX and TX blocks. This bit is set at power-on-reset
- ST_RT_REG - this bit controls the standby behavior of the regulator for the RX and TX blocks. If set, the regulator is disabled when STANDBY is high.
- AUD_REG_EN - this bit may be cleared to disable the regulator for the audio codec module. This bit is set a power-on-reset
- ST_AUD_REG - this bit controls the standby behavior of the regulator for the audio codec module. If set, the regulator is disabled when STANDBY is high.
- S_REG_EN - this bit may be cleared to disable the regulator for the Rx and Tx synthesizers. This bit is set at power-on-reset.
- ST_S_REG - this bit controls the standby behavior of the regulator for the Rx and Tx synthesizers. If set, the regulator is disabled when STANDBY is high.

The A2DIGL will output a control signal to each regulator to enable or disable it.

36.4.1.4 Tracking Clock Control

A programmable bit to enable the tracking clock, TRKG_OSC_OUT, (which is output by Neptune to Algae for use in Algae to tune it's filters) is required. This will be provided in the form of an additional SPI bit (STO_EN) which can be controlled more easily under L1T control.

When STO_EN is set, the 26MHz clock will be enabled to Algae. To reduce the number of SPI writes required, the clock will be disabled automatically after the synthesizer lock timer expires. See also Figure 36-9. The STO_EN bit will be cleared at power-on-reset

36.4.1.5 General Purpose ADC Control

The general purpose ADC interface block will provide for the following:

- interface to the general purpose ADC module
- interface to the MCU IP bus
- interface to TRSYNT (charge pump output voltage)
- output signals to the MCU interrupt controller

This module is MCU-controlled, and not controlled by the SPI Interface. For a full description, see Section 36.5.2, "A2DIGL and General Purpose ADC Interface."

36.4.2 Mixed Signal Interface Re-Use Summary

The following blocks are re-used from either the MAGIC-LV, the Tomahawk, or the GCAP3.

36.4.2.1 MAGIC-LV Module Re-use

- Rx and Tx Synthesizers (TRSYNT)
- Rx ADC and Channel Filter (RxAFE, RXCPROC)
- RX DC Offset Correction (DCADAPT)
- Receiver Overload Detection and Receiver Sigma-delta Modulator Dither Generation (RxSDG)
- Tx GMSK Modulation DAC (TX)

  Previously, these modules were controlled from Patriot by the MCU through L1T TOUT control signals, and L1T-triggered MQSPI transfers. All of these control signals will now be internally routed to what used to be the MAGIC-LV and Tomahawk modules. This interface will be largely software compatible with any previous Patriot code written for the MQSPI and MAGIC-LV.

36.4.2.2 Tomahawk Module Re-use

- PA Control (PAC)

  The PA control module is reused from Tomahawk which has a different SPI programming model than MAGIC-LV.

36.4.2.3 GCAP3 Module Re-use

- General Purpose ADC (GPADC)

  Previously, the general purpose ADC was located on the GCAP IC. It was accessed from the Patriot through the MQSPI. The interface to the general purpose ADC will now be handled by the MCU via peripheral registers, so the MCU software will not be compatible with existing software.

36.5 A2DIGL Control

This section is divided into two parts. One section contains detailed descriptions of all control associated with the receive and transmit modules (RXAFE, RxCPROC, DCADAPT, RxSDG, Tx, TRSYNT). The other section contains detailed description of the digital control of the GPADC MCU-based module.

36.5.1 Receive and Transmit Modules Interface Control

This logic block provides for the following:

- Interface to the TRSYNT, RxAFE, RXCPROC, DCADAPT, RxSDG, TX, PAC re-used modules; and TUNEC, REGUL and TOSW new modules
- SPI Bit decode logic
- Interface from internally routed L1T TOUT control signals
- Auxiliary SPI interface connection to Algae
- Interface to the MCU peripheral bus

The MCU peripheral bus interface provides for the following:

- Registers to read PAC parameters PWRFLT, PWRDAC, and PAC_CDET_OUT

- Register to configure bypass and control mode signal muxing

A block diagram of the interfaces is shown below in Figure 36-3.

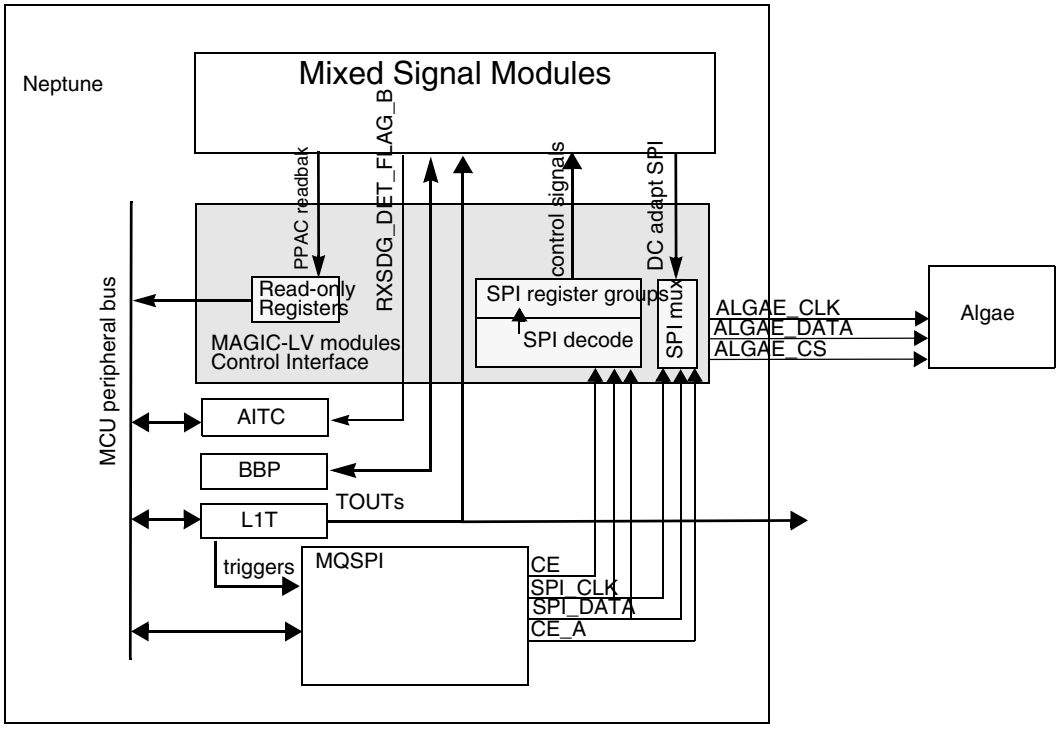


Figure 36-3. Receive and Transmit modules control interface

36.5.1.1 Support for bypass and control modes

The A2DIGL provides signal muxing to support control and bypass modes. The development control mode allows the mixed-signal modules to be controlled, for test purposes, from external pins instead of from internal signals from the MQSPI, L1T, and BBP. The bypass mode allows the IC to bypass the on-chip mixed signal modules to communicate with an external IC such as the Magic-LV. In the figures below, blue dashed lines indicate connections for control mode, and red dotted lines indicate connections for bypass mode.

Figure 36-4 below shows how the SPI signals are muxed. Depending on the SSEL bit in the MUXCTL register, the A2DIGL will accept SPI signals from either the MQSPI or, for the development control mode, external pins routed through the GPIO module. The SPI signal lines (SPI_CLK, SPI_CE_A, SPI_DATA) to the ALGAE are gated off when the SPI writes occur to A2DIGL SPI groups. This gating is achieved based on the ALG_DIS register from the GROUP 1. For completeness, the Algae SPI lines mux is shown in this figure; however, this mux has nothing to do with bypass and control modes; see Section 36.5.1.5, “Receive and Transmit Timing Sequences and Algae SPI Multiplex Operation” for more information on the Algae SPI muxing. See also Figure 6-9, “Netpune LTE EIM (External) Memory Space,” on page 32 in the chip configuration chapter for details on connections to MQSPI and GPIO.

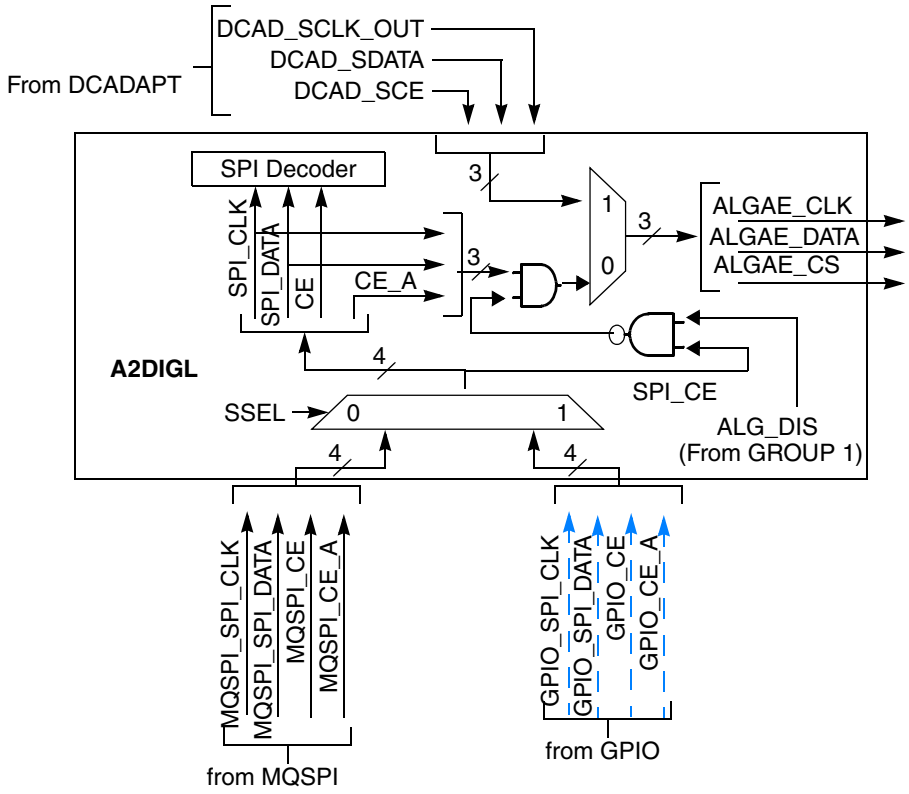


Figure 36-4. A2DIGL SPI input muxing.

Figure 36-5 below shows how the timing control signals are muxed. Depending on the TSEL bit in the MUXCTL register, the A2DIGL will accept timing control signals from either the L1T or, for the control mode, external pins routed through the GPIO module. RX_ACQ, which normally connects to the L1T TOUT0 output, is the receive enable signal inside Neptune; DMCS, which normally connects to the L1T TOUT1 output, is the transmit enable signal inside Neptune. These signals are used by the mixed-signal modules and also by A2DIGL. See also Figure 6-13, “L1T/A2DIGL Internal Connections,” on page 67 in the chip configuration chapter for details on connections to L1T and GPIO.

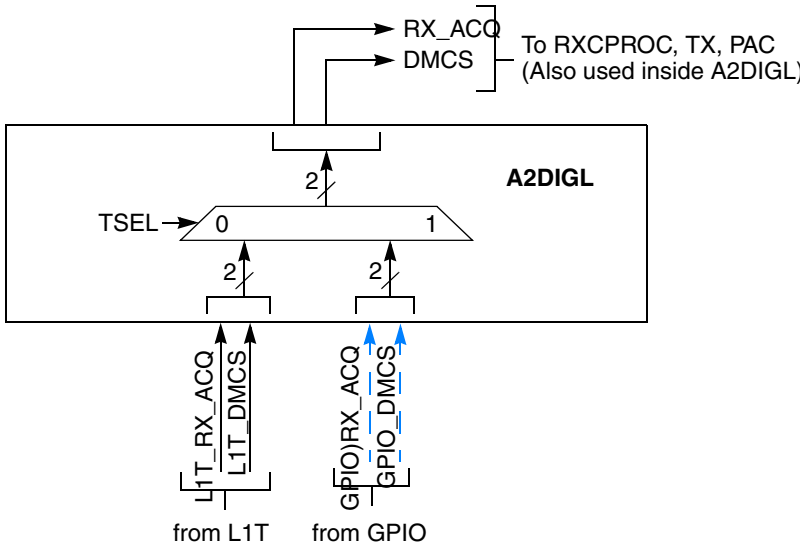


Figure 36-5. A2DIGL timing control signals muxing

Figure 36-6 below shows how the SSI signals are muxed. A2DIGL will route Tx, RXCPROC, BBP and GPIO signals according to the TxSEL1, TxSEL2, TxSEL3 and RxSEL bits in the MUXCTL register. See also Figure 6-15, “Connections for BBP Glue Logic,” on page 69 in the chip configuration chapter for details on connections to BBP, TX, RXCPROC, and GPIO. GPIO_SCKB_CLK comes from the PAD.

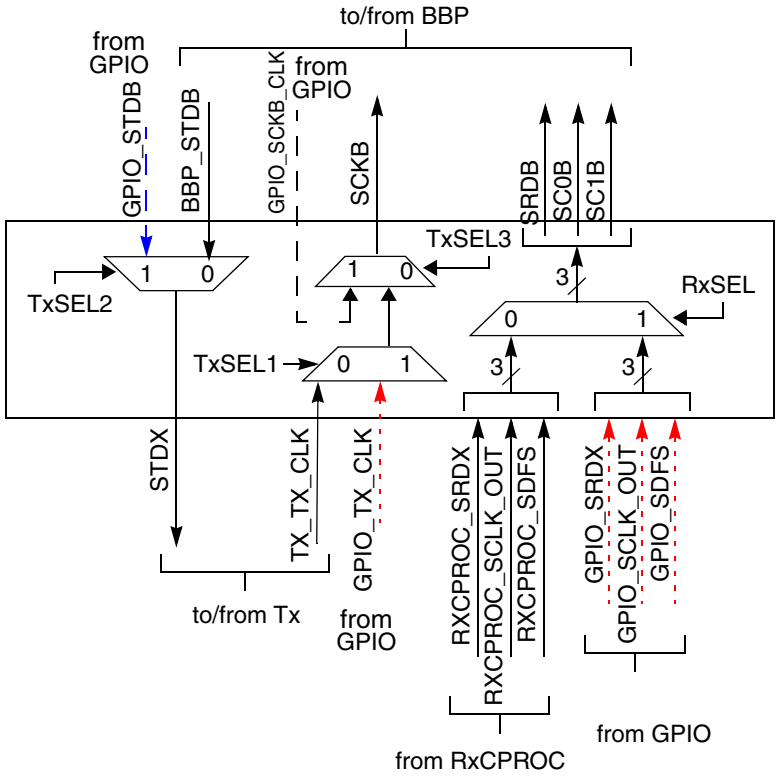


Figure 36-6. A2DIGL SSI muxing

36.5.1.2 SPI Signal Interface Description

SPI_DATA: serial data input line.

SPI_CLK: clock input line, data shifting occurs at the rising edge of this signal.

CE: SPI Decode Chip select line, active high.

CE_A: An input to the Algae chip select mux.

NOTE:

Data is latched into A2DIGL within [5] crystal clock periods after CE transitions from high to low.

36.5.1.3 SPI Decoder Requirements

1. The maximum clock rate is required to be 13MHz.
2. Data are transmitted most significant bit first. Each data field is a total of 80 bits.
3. SPI_DATA and SPI_CLK signals are ignored as long as the chip select is low (logic 0).
4. The chip select should be active (logic 1) only during the serial data transmission.
5. All input data is sampled at the rising edge of the SPI_CLK signal. Any transition on SPI_DATA should occur at least 5nsec. before the rising edge of SPI_CLK and remain stable for at least 5nsec after the rising edge of SPI_CLK. (Typically data transitions on SPI_DATA are done on SPI_CLK falling edge.)
6. The chip select has to be active (logic 1) at least 5nsec before the rising edge of the first SPI_CLK signal, and has to remain active (logic 1) at least 61.5nsec after the last falling edge of SPI_CLK. The recommended time interval for both cases is half a clock cycle.
7. Coincident rising or falling edges of SPI_CLK and the chip select are not allowed. If the SPI_CLK signal is to be held at logic 1 after the data transmission, the falling edge of the SPI_CLK signal must occur at least 5nsec. before the chip select becomes an active logic 1 for the next set of data.
8. If the chip select goes low before enough bits are sent then bad data will be written into the registers.
9. When the chip select goes low to complete the SPI operation then the next rising edge of the chip select must be delayed by at least [30nsec].

36.5.1.4 SPI Decoder Operation Description

The control bits are organized into 16 fields. Each of these 16 fields contains 80 bits. A maximum of 76 data bits are used per field. The remaining four bits are used to address the 16 fields.

For each SPI transfer, the four bit address is written to the SPI_DATA line MSB first. Next, data bits are written to the SPI_DATA pin MSB first. Once all the data bits are written then the data is transferred into the actual registers on the falling edge of CE. CE must go low and return high to start the next SPI data transfer.

The control registers should be implemented as fully static buffers so as to support low-power standby modes.

NOTE:

All unused SPI bits in each field must be written to a zero.

Figure 36-7 is a diagram showing the details of an SPI transfer.

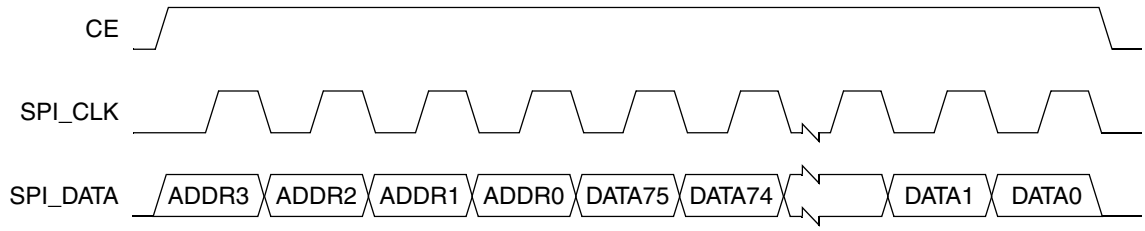


Figure 36-7. Timing Diagram

Figure 36-8 shows a multiple read/write using the SPI bus.

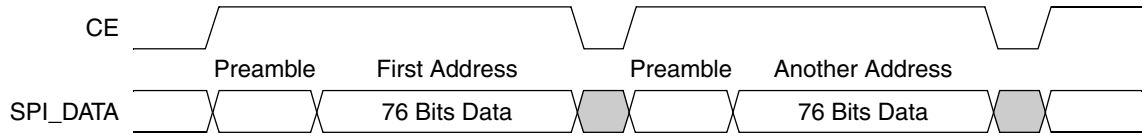


Figure 36-8. Timing Diagram

36.5.1.5 Receive and Transmit Timing Sequences and Algae SPI Multiplex Operation

The A2DIGL is responsible for controlling some of the low-level timing signals for the receive and transmit sequences. One component of this is the Algae SPI multiplexer. Normally the MQSPI can write directly to the Algae. However, during the receive DC adapt sequence, the DCADAPT module needs to be able to write to the Algae SPI. The A2DIGL block handles the muxing of the SPI signals from MQSPI and DCADAPT which are intended for Algae.

Much of the receive and transmit timing described below is programmable through the A2DIGL SPI. The A2DIGL SPI operation is described in Section 36.5.1.4, “SPI Decoder Operation Description,” on page 36-27, and the SPI programming model is described in Section 36.5.1.10, “SPI Interface Bit Groups,” on page 36-54. See also Section 36.5.1.6, “SPI bits and signals requiring additional logic,” on page 36-44.

36.5.1.5.1 Receive VLIF mode timing sequences

The figures below illustrate the timing signals that the A2DIGL will generate when the receiver is operating in VLIF mode. Several figures are presented to illustrate different possibilities which need to be supported. Note that the figures are not to scale, and that signals in bold text are outputs of the A2DIGL.

The first figure shows the case where the ADAPT_EN and AB_SEL SPI bits are both set.ADAPT_EN enables the DC offset correction sequence, and AB_SEL controls the way the receiver is disabled at the end of the burst.

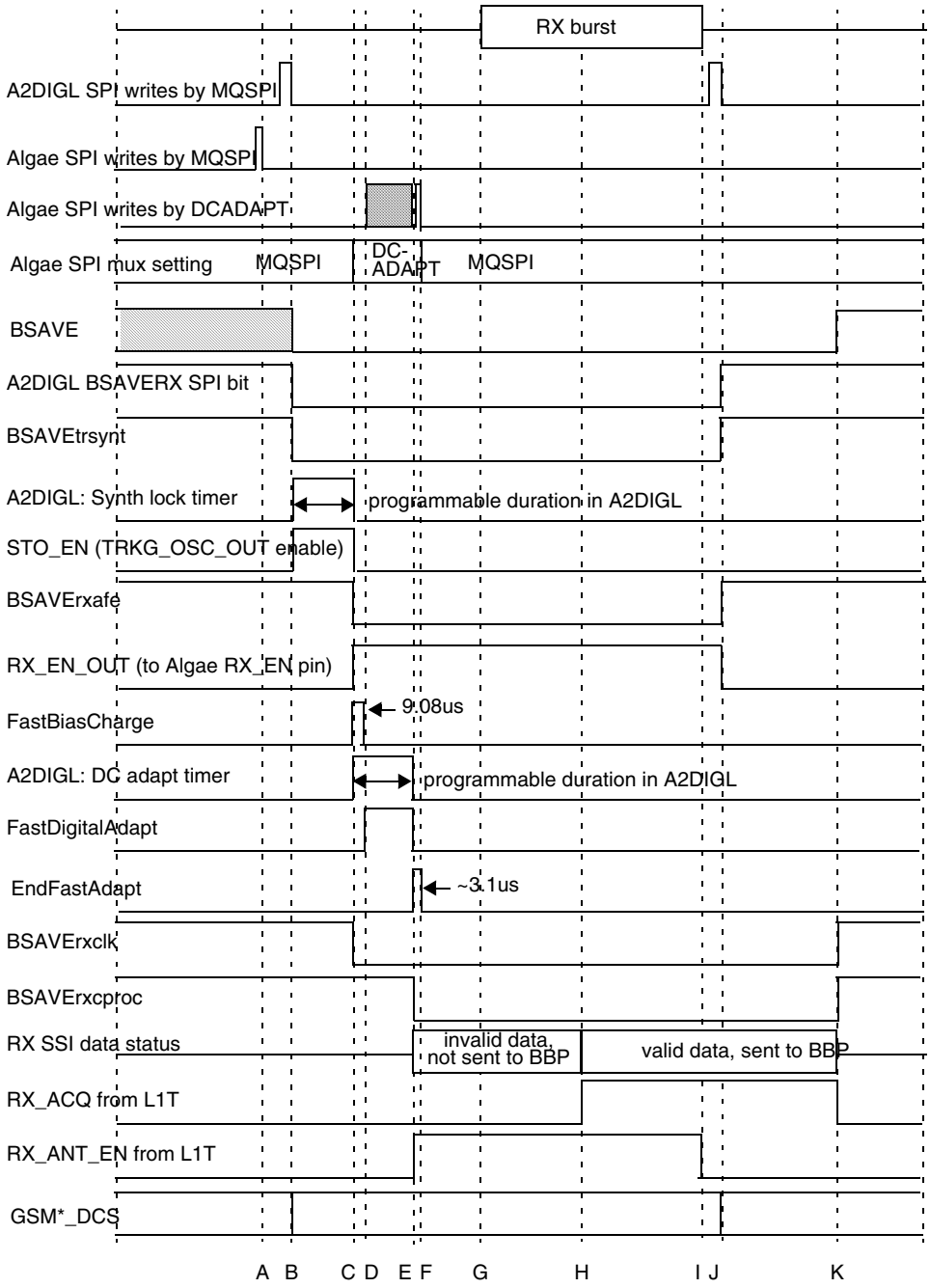


Figure 36-9. Receive Timing Sequence for VLIF mode, AB_SEL=1, ADAPT_EN=1

Marker description for Figure 36-9:

A) The MQSPI writes to the Algae. This turns on the VCO, sets RF and baseband AGC levels, etc.

B) The MQSPI will write to A2DIGL SPI group 5 to clear BSAVERX, set ADAPT_EN, set STO_EN, program GSM*_DCS bit, program the synthesizer, etc. Clearing BSAVERX starts the receive sequence in A2DIGL; this will enable the synthesizer (BSAVEsynth will go low) and start the programmable synthesizer lock timer within A2DIGL which is defined by the SL_TIMER[3:0] bits. Setting STO_EN will enable

the TRKG_OSC_OUT[1]. The GSM*_DCS pin will be asserted or deasserted at this time according to the programming of the GSM*_DCS bit.Any SPI write to a group other than group 4 will automatically clear BSAVE, so BSAVE is also forced low at this time if it is not already low.

C) When the A2DIGL's synthesizer lock timer expires, the tracking clock to Algae will be disabled, BSAVErxafe goes low (the RXAFE module exits battery save mode), BSAVErxclk goes low (enables the digrx_clk output of RXAFE), and the RX_EN_OUT signal will be asserted by A2DIGL (enabling Algae). A 10us FastBiasCharge signal will be generated by A2DIGL to the RXAFE. The SPI multiplexer will switch so that the DCADAPT block can drive the SPI lines to Algae instead of the MQSPI. The programmable DC adapt timer within A2DIGL (defined by the DCA_TIMER[2:0] bits) will be started at this time to control the duration of the DC adapt sequence.

D) After 10us, the FastBiasCharge signal will be deasserted, and since ADAPT_EN bit has been set, the DC adapt sequence will begin. The FastDigitalAdapt signal will be asserted by A2DIGL to the DCADAPT module which performs SPI writes to Algae. (These SPI writes include I/Q DC offset correction values as well as the AGC_HIGH and LNA_OFF bits which configure Algae for the DC adapt sequence.) The duration of the FastDigitalAdapt pulse is the time specified by the DCA_TIMER[2:0] spi bits minus 10us for the FastBiasCharge signal.

E) When A2DIGL's DC adapt timer expires, the DC adapt sequence will end. The FastDigitalAdapt signal will be deasserted, and the EndFastAdapt signal will be asserted by A2DIGL one cycle later. BSAVErxcproc will go low (to enable the RxCPROC module). The deassertion of FastDigitalAdapt and the assertion of EndFastAdapt causes DCADAPT to perform one additional SPI write to Algae which clears it's AGC_HIGH and LNA_OFF bits. (For completeness, the timing diagram shows that RX_ANT_EN goes high near this time as well. One of the L1T TOUT pins on Neptune will be used for RX_ANT_EN, which is the receive antenna enable signal going to the antenna switch. RX_ANT_EN is not used by, or even an input to A2DIGL; it is shown here for reference only.)

F) When the EndFastAdapt is deasserted by A2DIGL, (EndFastAdapt duration is 40 cycles of the PAT_REF clock, ~3.1us), the Algae SPI mux in A2DIGL will switch back so that MQSPI can write to the Algae SPI.

G) The beginning of the receive burst.

H) RX_ACQ is asserted by the L1T to indicate to the RXCPROC module that it should begin sending data to the BBP module. The delay between the start of the RX burst and RX_ACQ going high corresponds to the time required by the signal to pass through the RXCPROC.

I) The end of the receive burst. The RX_ANT_EN signal will be deasserted by the L1T near this time.

J) After the end of the burst, the L1T will trigger an MQSPI write to program A2DIGL group 5 bits to set BSAVERX. This will cause A2DIGL to deassert RX_EN_OUT, and assert BSAVErxafe and BSAVEtrsynt.

K) When the L1T deasserts RX_ACQ (indicates that the RXCPROC has completed sending all valid data to the BBP), A2DIGL will assert BSAVErxcproc, set BSAVE, and 128 clocks (13MHz) later assert BSAVErxclk.

1. If STO_EN=0, the TRKG_OSC_OUT will not be enabled.

In the previous figure, an A2DIGL SPI write at the end of the burst disabled all the receiver modules except the RXCPROC. If the last A2DIGL SPI write did not occur, all of the receiver modules would have been disabled when RX_ACQ went low since AB_SEL=1. This is illustrated below in Figure 36-10.

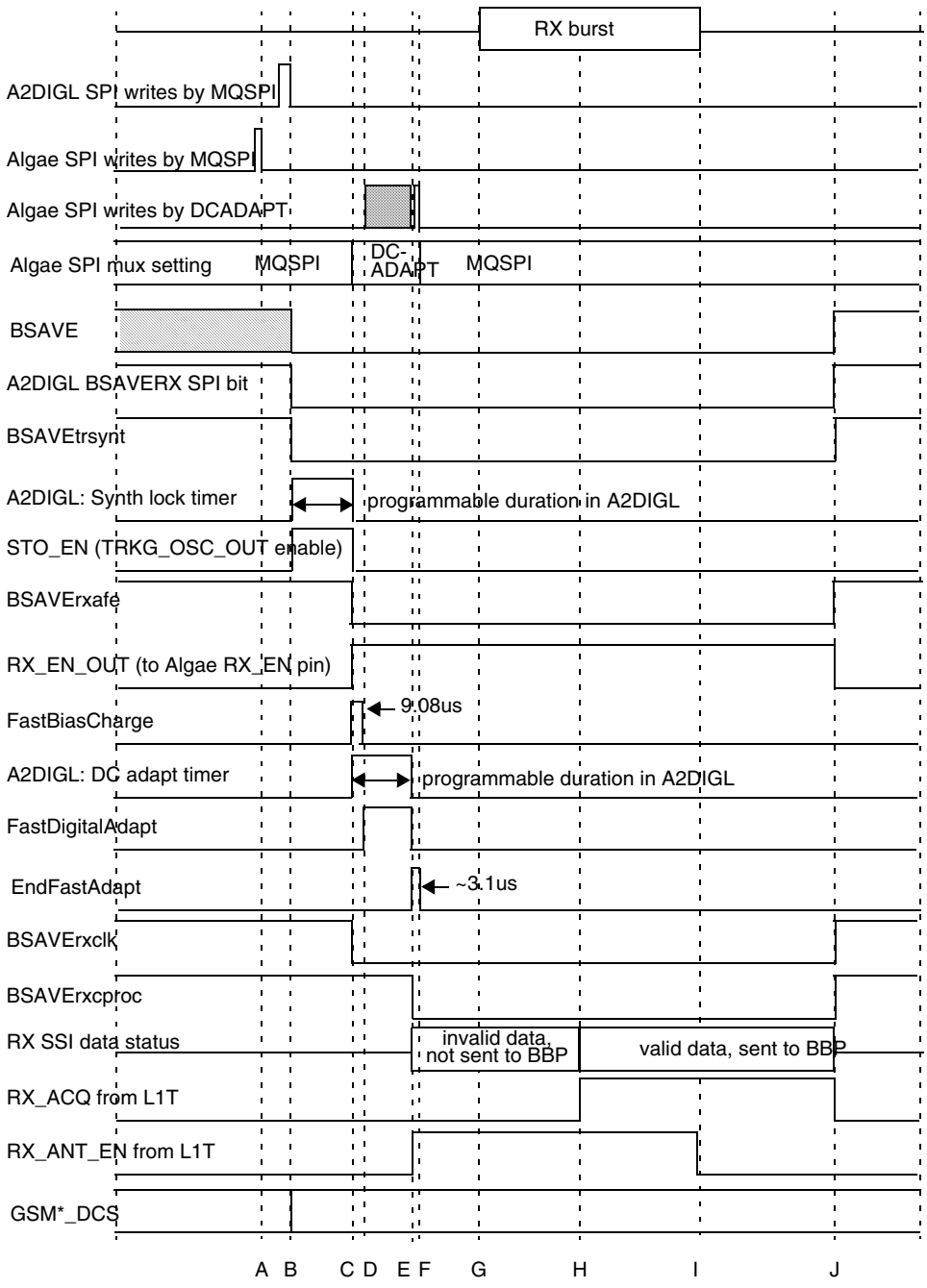


Figure 36-10. Receive Timing Sequence for VLIF mode, AB_SEL=1, ADAPT_EN=1 without a SPI write at the end of the burst

Another possibility is presented below in Figure 36-11. In this case, ADAPT_EN=0, so the DC offset correction will not be performed: the DC adapt timer within A2DIGL will not be enabled, the Algae SPI mux within A2DIGL will not be switched, and the FastDigitalAdapt and EndFastAdapt signals will not be generated by A2DIGL. Also since there is no adapt, BSAVErxcproc goes low when the FastBiasCharge goes low.

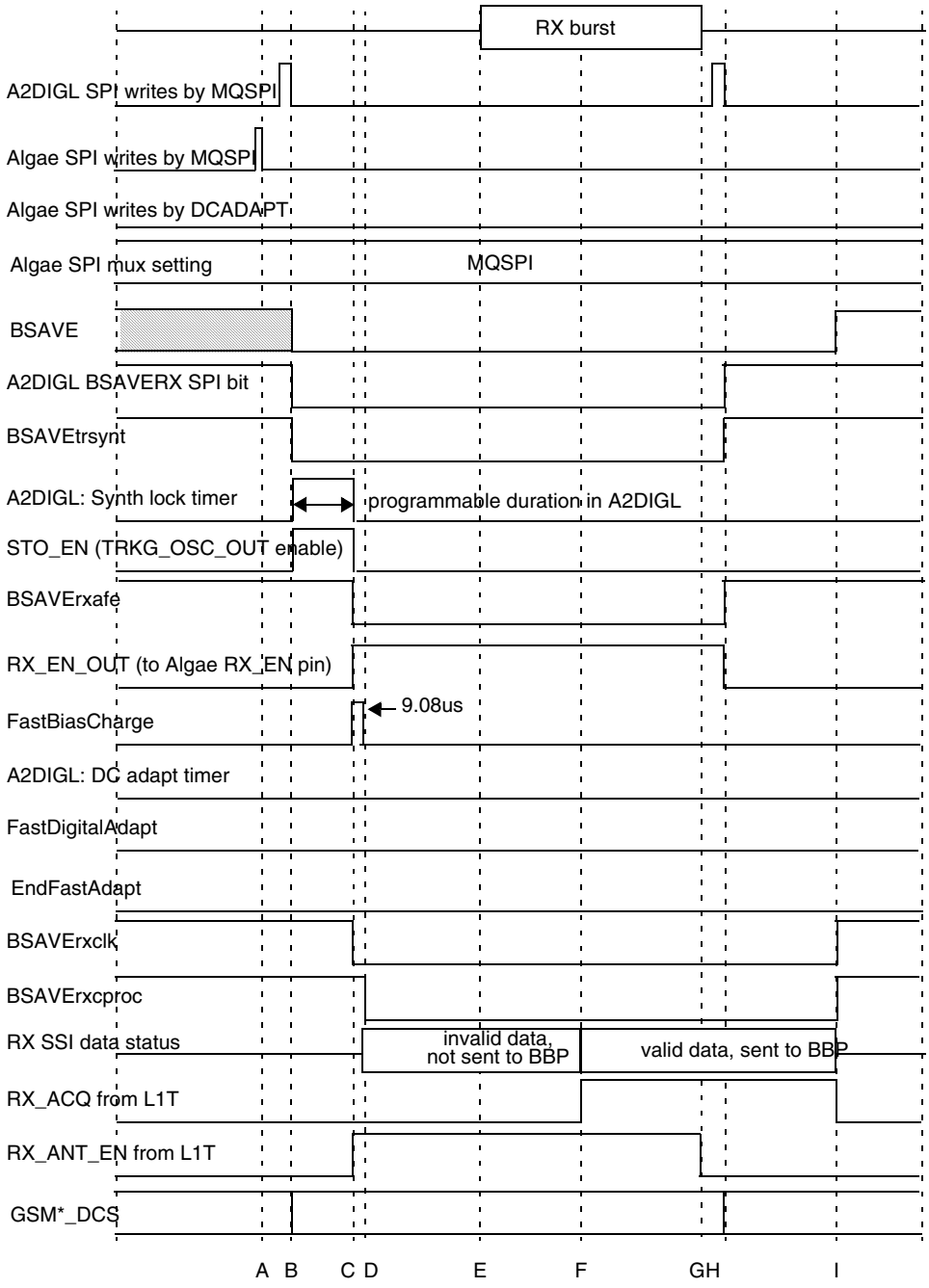


Figure 36-11. Receive Timing Sequence for VLIF mode, AB_SEL=1, ADAPT_EN=0

The case where AB_SEL=0 is shown below in Figure 36-12. Since the AB_SEL SPI bit is cleared, none of the signals will be disabled automatically at the end of the receive burst (except for BSAVErxcproc which always goes high when RX_ACQ goes low and BSAVErxclk, which always goes high 128 clocks after RX_ACQ goes low.). A group 5 SPI write which sets BSAVERX bit is required after the end of the burst to disable the receiver; note that this can occur before the RXCPROC has finished transferring data as RXCPROC module will continue to transfer data until RX_ACQ falling edge. Also note that BSAVE is not set at marker J since AB_SEL=0.

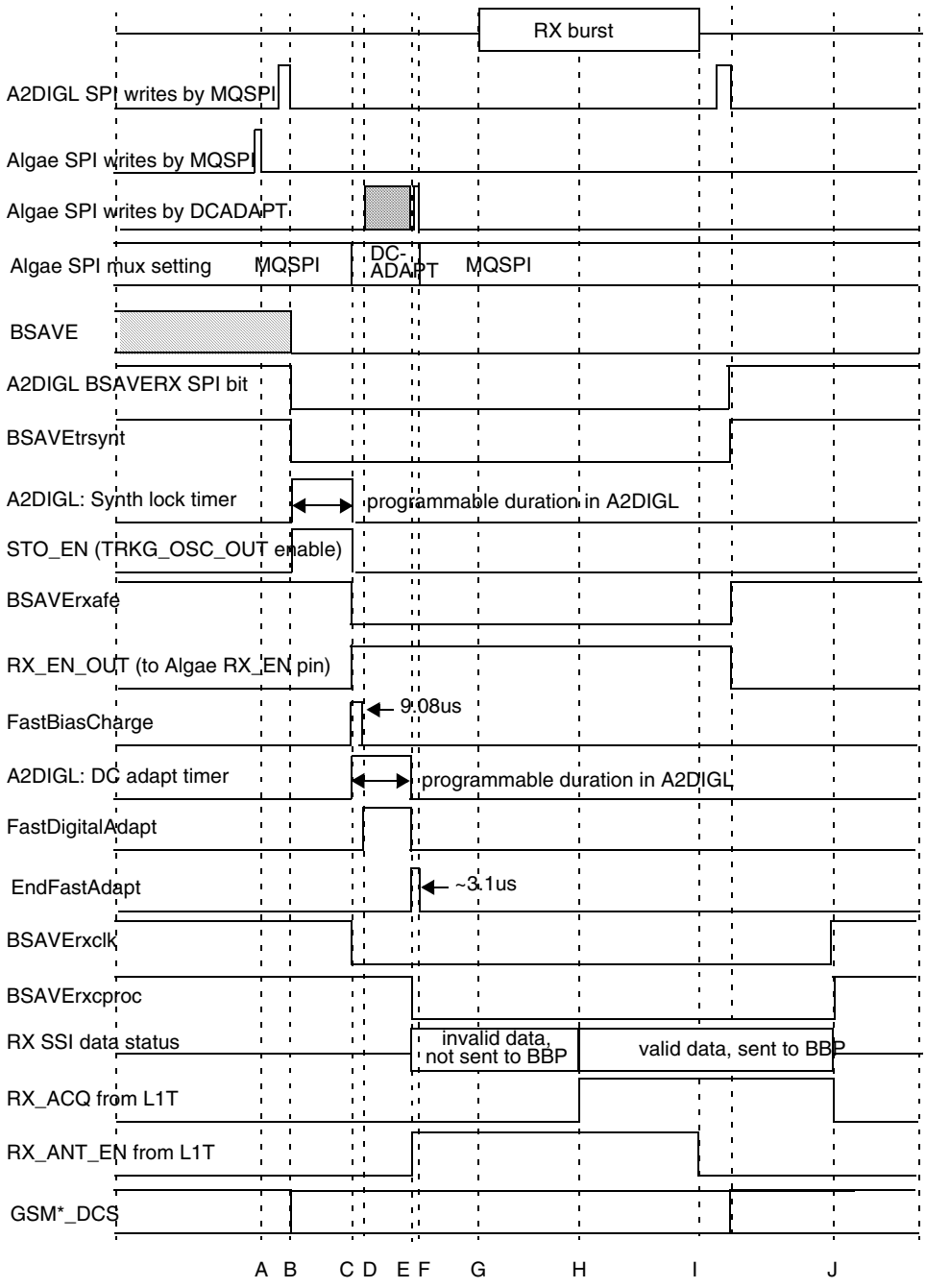


Figure 36-12. Receive Timing Sequence for VLIF mode, AB_SEL=0, ADAPT_EN=1

One other special case that must be considered is when a GPRS FCH scan occurs immediately before a receive slot for the serving cell. This timing is shown below in Figure 36-13.

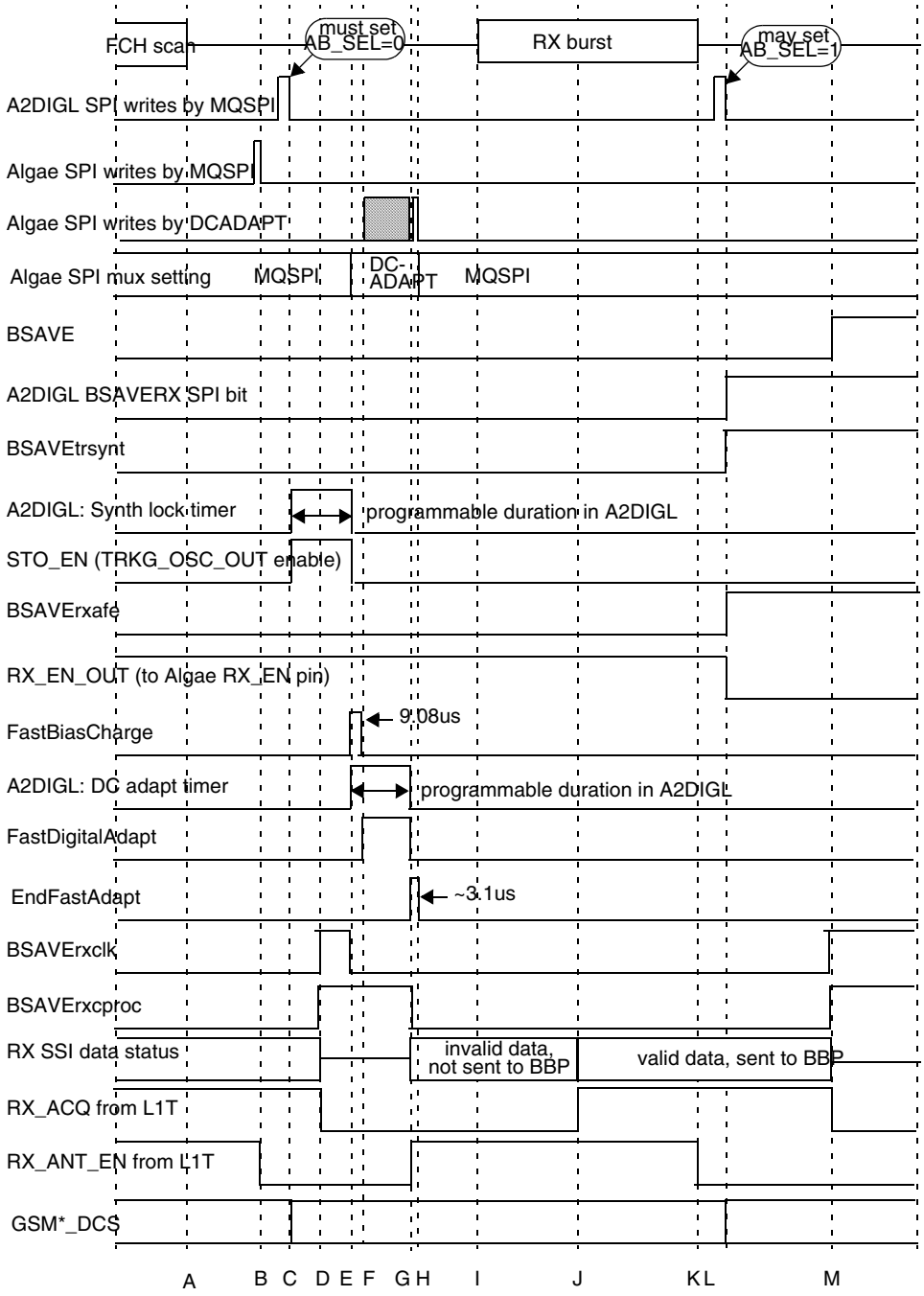


Figure 36-13. Receive Timing Sequence for VLIF mode, GPRS FCH scan prior to Serving Cell

The differences between this figure and the previous figures relates to the following markers:

A) The end of the FCH scan prior to the service cell receive burst.

B) The MQSPI writes to the Algae to configure it for the next receive slot. Near this time, the L1T will disable RX_ANT_EN signal.

C) The MQSPI will write to A2DIGL SPI group 5 and clear BSAVERX, set ADAPT_EN, set STO_EN, program GSM*_DCS, program the synthesizer, etc. The AB_SEL bit must also be cleared in this SPI write for proper operation[1]. The programmable

synthesizer lock timer within A2DIGL defined by the SL_TIMER[3:0] bits will start. Setting STO_EN will enable the tracking clock to Algae on the TRKG_OSC_OUT pin. The GSM*_DCS pin will be asserted or deasserted at this time according to the programming of the GSM*_DCS bit. Since BSAVERX was previously low, BSAVErxafe, BSAVErxclk, and RX_EN_OUT will remain in their previous states.

D) RX_ACQ from the L1T goes low. BSAVErxcproc will go high and BSAVErxclk will go high 128 clocks later. The RXCPROC module will stop sending data to the BBP module. (Note that none of the other BSAVE signals go high at this time since AB_SEL is cleared.)

E) When the synthesizer lock timer within A2DIGL expires, the tracking clock to Algae will be gated off, and a 10us FastBiasCharge signal will be generated by A2DIGL to the RXAFE[1]. The programmable DC adapt timer (defined by the DCA_TIMER[2:0] bits) will be started at this time to control the duration of the DC adapt sequence. BSAVErxclk will go low.

As shown, BSAVERX and the AB_SEL SPI bit are set in the SPI write at marker L. If AB_SEL had been set to 0 by this SPI write, BSAVE would not have gone high at marker M.

36.5.1.5.2 Detailed DC offset correction and SPI mux timing for VLIF

Figure 36-14 below shows the DC Adapt and SPI mux timing sequence for VLIF mode in greater detail. When FastBiasCharge signal output by A2DIGL goes high, the Algae SPI mux within A2DIGL is switched so that the DCADAPT module can write to the Algae SPI, and the programmable DC adapt timer (DCA_Timer[3:0]) within A2DIGL will begin. When the FastBiasCharge signal goes low 10us later, FastDigitalAdapt output of A2DIGL will go high, signalling the DCADAPT module to perform sequential SPI writes to Algae. (During this time, DCADAPT uses the AGC_HIGH and LNA_OFF values provided by A2DIGL.) Note that the pulse width of the FastDigitalAdapt signal output by A2DIGL is the time specified by the DCA_TIMER[3:0] spi bits minus 10us for the FastBiasCharge signal. Each SPI write to Algae by the DCADAPT module takes 17 cycles of the digrx_clk (13MHz uncorrected clock), and there is 1 cycle between writes. When the DC adapt timer within A2DIGL expires, FastDigitalAdapt will be deasserted by A2DIGL, which signals the DCADAPT module to finish any SPI writes in progress. EndFastAdapt output by A2DIGL will go high one cycle later. This signals the DCADAPT module to perform one additional SPI write to reset Algae. This final SPI write by DCADAPT to Algae clears the AGC_HIGH and LNA_OFF bits. EndFastAdapt will be held high by A2DIGL for 40 cycles of pat_ref so that DCADAPT can complete its transfers.

1. If AB_SEL is not cleared at this time, the BSAVE signals would go high on RX_ACQ falling edge, marker “D”, which is not desirable.
1. If RX_ACQ is still high when the synth lock timer within A2DIGL expires, the adapt sequence will be aborted: the DC adapt timer will not be enabled, the FastBiasCharge, FastDigitalAdapt, and EndFastAdapt signals will not be generated, and the Algae SPI mux will not switch to DCADAPT.

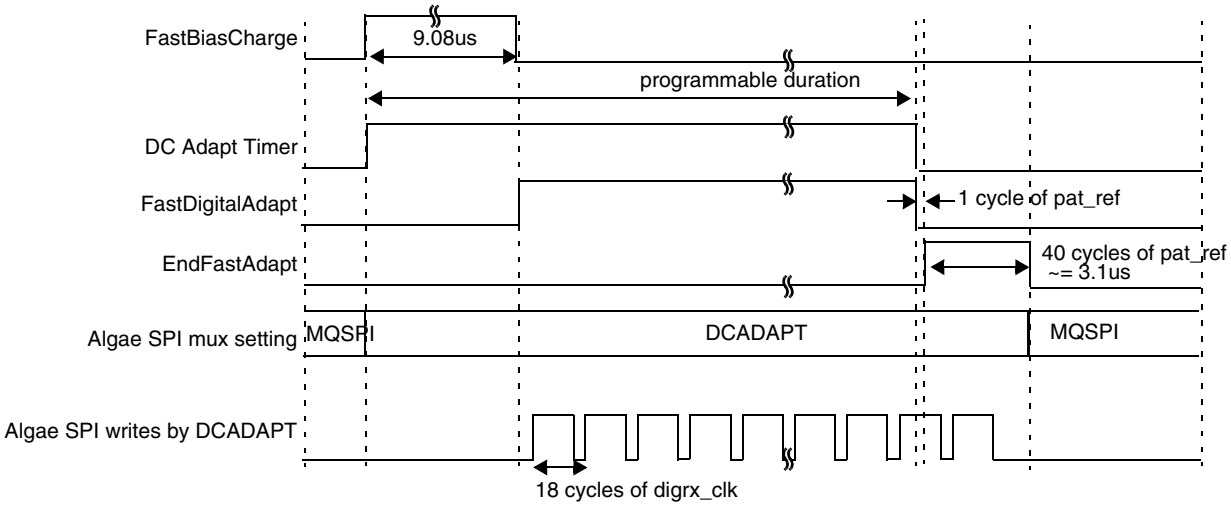


Figure 36-14. DC Adapt and SPI Mux Timing Detail for VLIF mode

36.5.1.5.3 Receive DCR mode timing sequence

When the receiver is configured for DCR mode, A2DIGL will implement the timing sequences shown below. Although only one case is illustrated below, the different scenarios previously described for VLIF which depend upon the programming of ADAPT_EN, AB_SEL, STO_EN, etc. are supported.

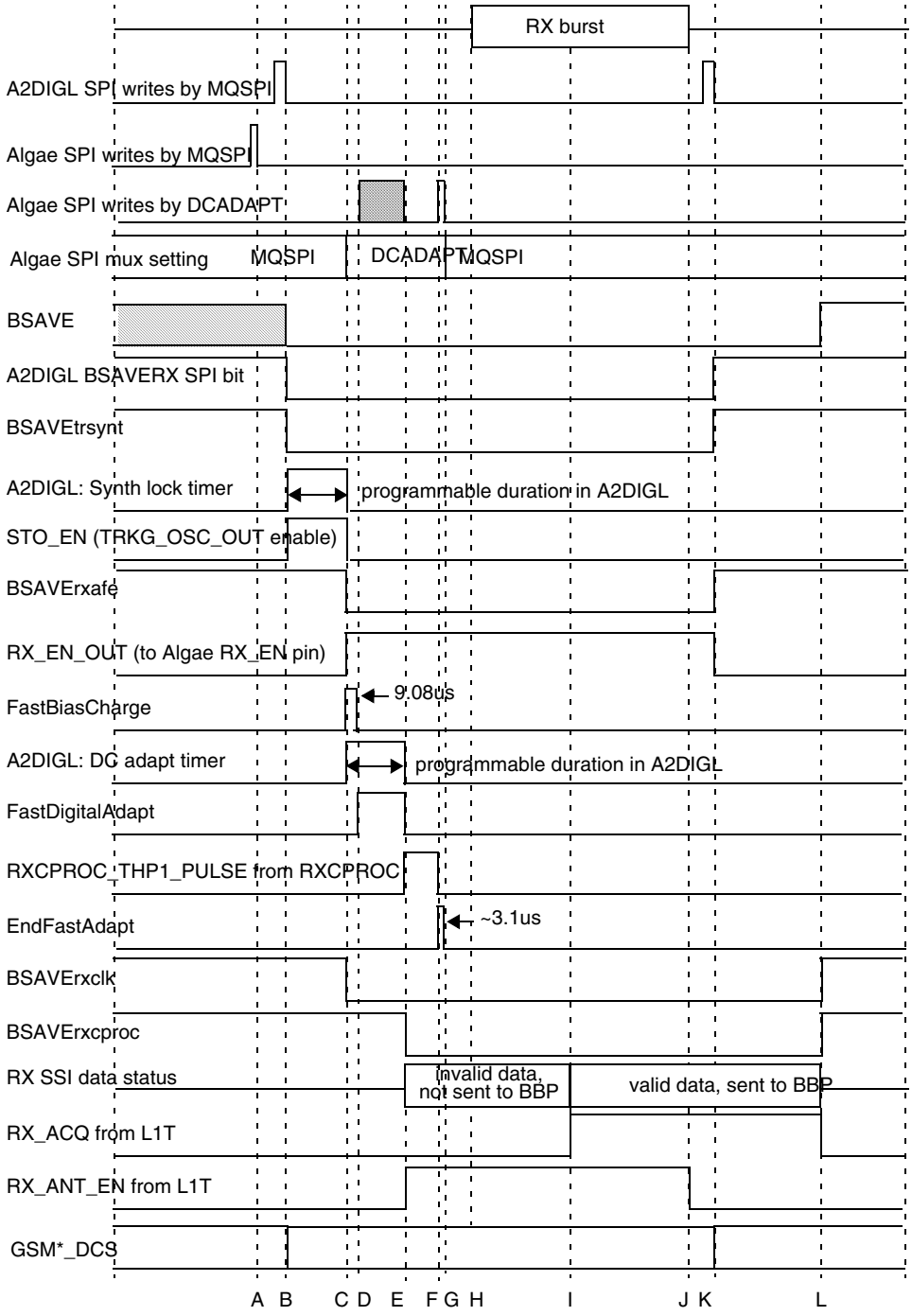


Figure 36-15. Receive Timing Sequence for DCR mode, ADAPT_EN=1, AB_SEL=1

The differences between this figure and the VLIF mode Figure 36-9 are as follows:

E) When the DC adapt timer (defined by DCA_TIMER[3:0] spi bits) within A2DIGL expires, FastDigitalAdapt output by A2DIGL goes low (DCADAPT will complete any current transfer to Algae, and then halt), and the RXCPROC module will be brought out of battery save mode. The RXCPROC's fine DC adapt phase THP1 will commence. (The duration of this phase is controlled by a timer within RXCPROC and determined by programming of the THP1[2:0] bits.)

F) When the RXCPROC's THP1 phase ends (as indicated by the falling edge of RXCPROC_THP1_PULSE input to A2DIGL), EndFastAdapt pulse will be generated by A2DIGL to the DCADAPT module so it can perform one additional SPI write to reset Algae. This pulse will be 40 cycles of the PAT_REF clock.

G) At the end of the EndFastAdapt pulse, the Algae SPI mux within A2DIGL will be switched so that MQSPI can write to Algae.

36.5.1.5.4 Detailed DC offset correction and SPI mux timing for DCR

Figure 36-16 below shows the DC Adapt and SPI mux timing sequence for DCR mode in greater detail. When FastBiasCharge output of A2DIGL goes high, the Algae SPI mux within A2DIGL will be switched so that the DCADAPT module can write to the Algae SPI, and the programmable DC adapt timer (DCA_Timer[3:0]) within A2DIGL will begin. When FastBiasCharge output of A2DIGL goes low 10us later, FastDigitalAdapt will be asserted by A2DIGL, which signals the DCADAPT module to perform sequential SPI writes to Algae. (During this time, DCADAPT uses the AGC_HIGH and LNA_OFF values provided by A2DIGL.) Each SPI write by DCADAPT takes 17 cycles of the digrx_clk (13MHz uncorrected clock), and there is 1 cycle between writes. When the DC adapt timer within A2DIGL expires, FastDigitalAdapt will be deasserted by A2DIGL, which signals DCADAPT to finish any SPI writes in progress. Also at this time, BSAVErxcproc will go low to enable the RXCPROC module. The RXCPROC will generate a signal RXCPROC_THP1_PULSE to indicate when it's "fine DC offset correction" phase 1 is complete (this duration is controlled by a timer within RXCPROC which is programmed by the THP1[2:0] bits). When RXCPROC_THP1_PULSE goes low, EndFastAdapt output of A2DIGL will go high, which signals the DCADAPT module to perform one additional SPI write to reset Algae. This final SPI write by DCADAPT to Algae clears the AGC_HIGH and LNA_OFF bits. The EndFastAdapt output of A2DIGL will be held high for 40 cycles of pat_ref so that DCADAPT can complete its transfer.

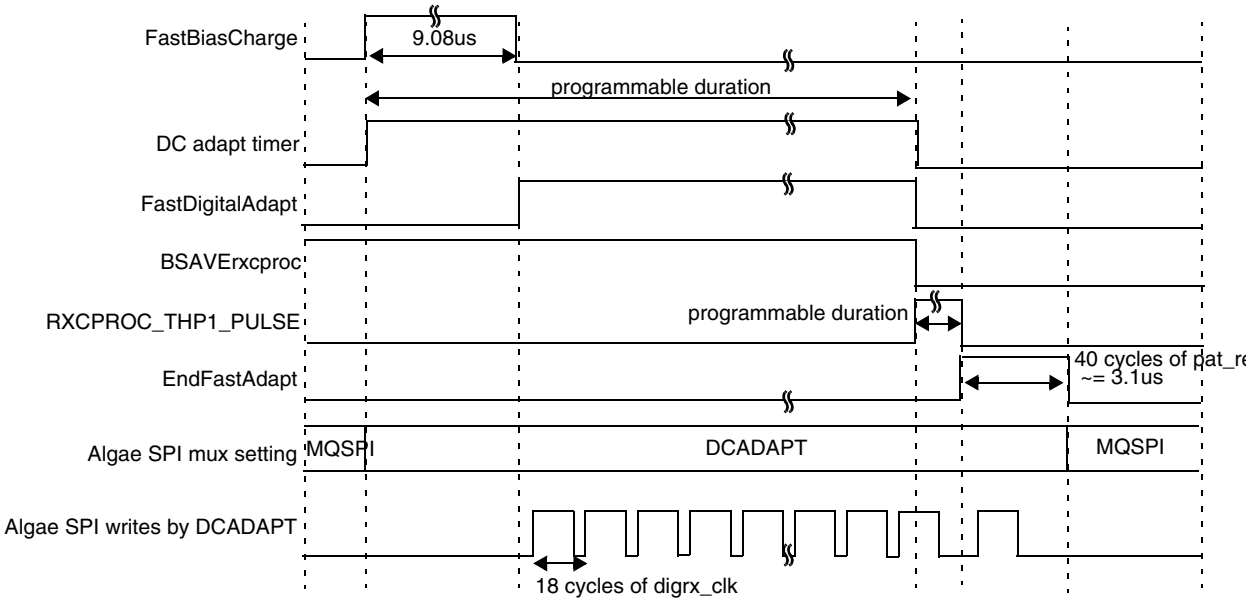


Figure 36-16. DC Adapt and SPI Mux Timing Detail for DCR mode

36.5.1.5.5 Transmit Timing Sequence

Transmit timing diagrams are shown below for the possible programming of the ABR_SEL and ABF_SEL SPI bits. These two bits control the battery save behavior for the transmit timing: ABR_SEL controls when battery save is exited at the beginning of the burst; ABF_SEL controls when battery save is re-entered at the end of the burst.

The first figure, Figure 36-17, shows the case where ABR_SEL=0 and ABF_SEL=1. ABF_SEL, if set, forces the transmitter into battery save mode automatically at the end of the transmit burst as indicated by the rising edge of the PAC_DN_SIG signal which is input to A2DIGL from the PAC.

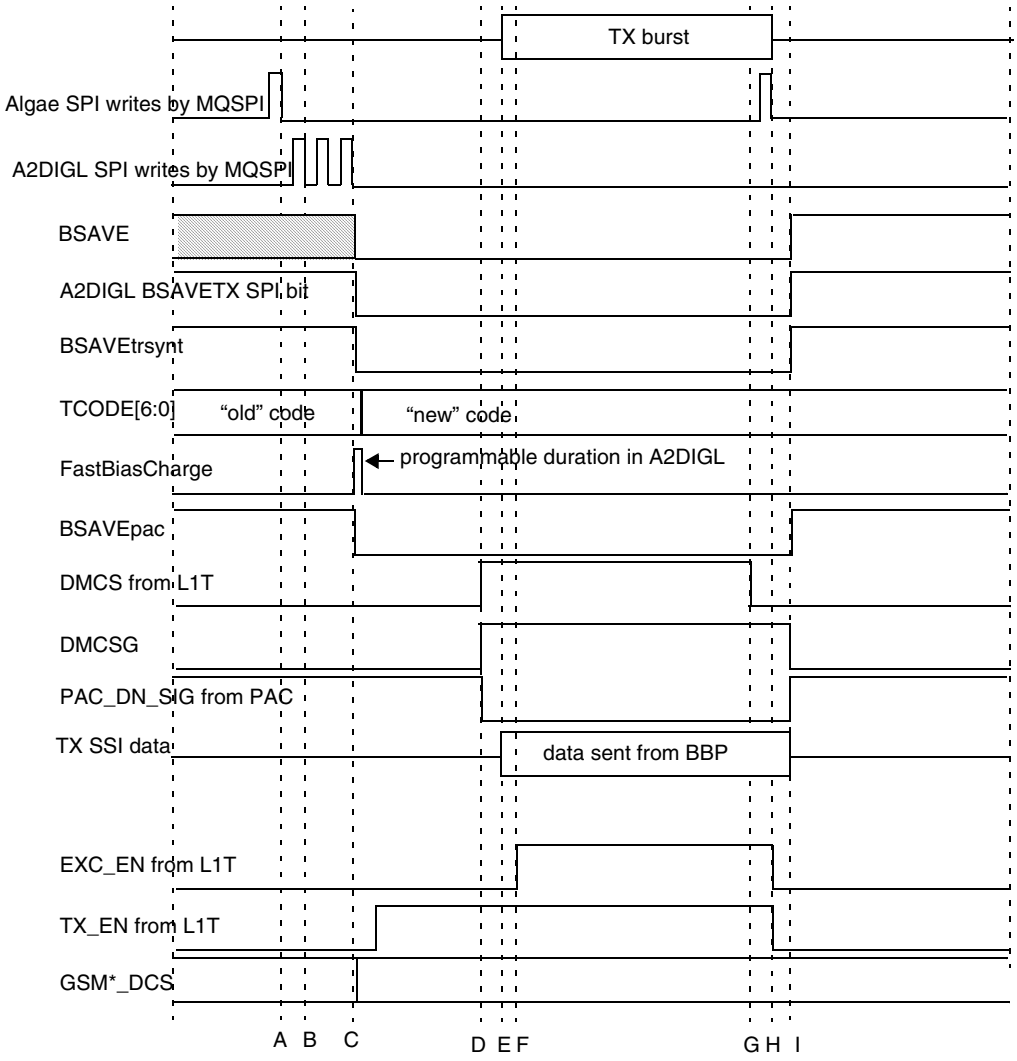


Figure 36-17. Transmit Timing Sequence, ABR_SEL=0, ABF_SEL=1

A detailed description of the markers for this figure is given below.

A) Algae SPI write by MQSPI to enable the TX VCO.

B) A2DIGL group 7 and 8 SPI writes by MQSPI are also required prior to transmit slots to program additional TX and PAC parameters that could not be fit into group 5.

C) A2DIGL group 5 SPI write by MQSPI to clear BSAVETX, program GSM*_DCS, program synthesizer, etc. Clearing BSAVETX starts the A2DIGL transmit timing sequence; this will pull BSAVE and BSAVEtrsynt low. Since ABR_SEL=0, BSAVEpac will be pulled low at this time also. At the falling edge of BSAVETX, if the CNVE (in the CTRL register) bit is set, the count down (based on the GPD [3:16] in the MTR memory mapped register of A2DIGL) will begin. With the count down to zero, A2DIGL will start asserting the GPADC acquisition sequence (see Section 36.5.2, “A2DIGL and General Purpose ADC Interface” for more details).The group 5 write will cause the A2DIGL to request a new tuning code from the TUNEC module and output it to the TX and PAC modules (update occurs ~10 us after the SPI write). The FastBiasCharge signal will be generated by A2DIGL at this time; this signal has a fixed 10us duration for receive slots, but a programmable duration for transmit slots which is controlled by a timer within A2DIGL and specified by the FBC_TD[3:0] group 1 SPI bits. The GSM*_DCS pin will be asserted or deasserted at this time according to the programming of the GSM*_DCS bit. The L1T will pull TX_EN high around this time to enable the PA control IC prior to the TX burst (this signal is not used by the A2DIGL and is shown here for reference purposes only).

D) The L1T pulls DMCS input to A2DIGL high. The A2DIGL pulls its DMCSG output high at this point (DMCSG goes to the BBP glue logic, and the reset value of DMCSG must be 0 for proper initialization of BBP.)

E) The TX will initiate SSI data transfers from the BBP a short time after DMCS goes high (specified by the SSI_DLY SPI bits).

F) Near this time, the L1T will pull EXC_EN high (this signal is not used by the A2DIGL, and is shown here for reference only).

G) The L1T pulls DMCS low. This signal is used by the PAC for ramp down. The SSI will continue to transmit (dummy) data after this time.

H) Algae SPI write by MQSPI to turn TX VCO off. L1T will deassert TX_EN and EXC_EN near this time as well.

I) DN_DLY timer within the PAC expires as indicated by the rising edge of the PAC_DN_SIG signal from the PAC. SSI data is discontinued. The A2DIGL will pull DMCSG low at this time. Also at this time since ABF_SEL=1 the following signals will be pulled high by A2DIGL: BSAVEpac, BSAVEtrsynt, BSAVETX, and BSAVE.

The next figure, Figure 36-18, is the same as the previous figure except that ABF_SEL=0. In this case, the transmitter does not automatically enter battery save mode at the end of the transmit slot; an A2DIGL group5 SPI write (shown at marker J) is necessary to enter battery save mode (by setting BSAVETX=1).

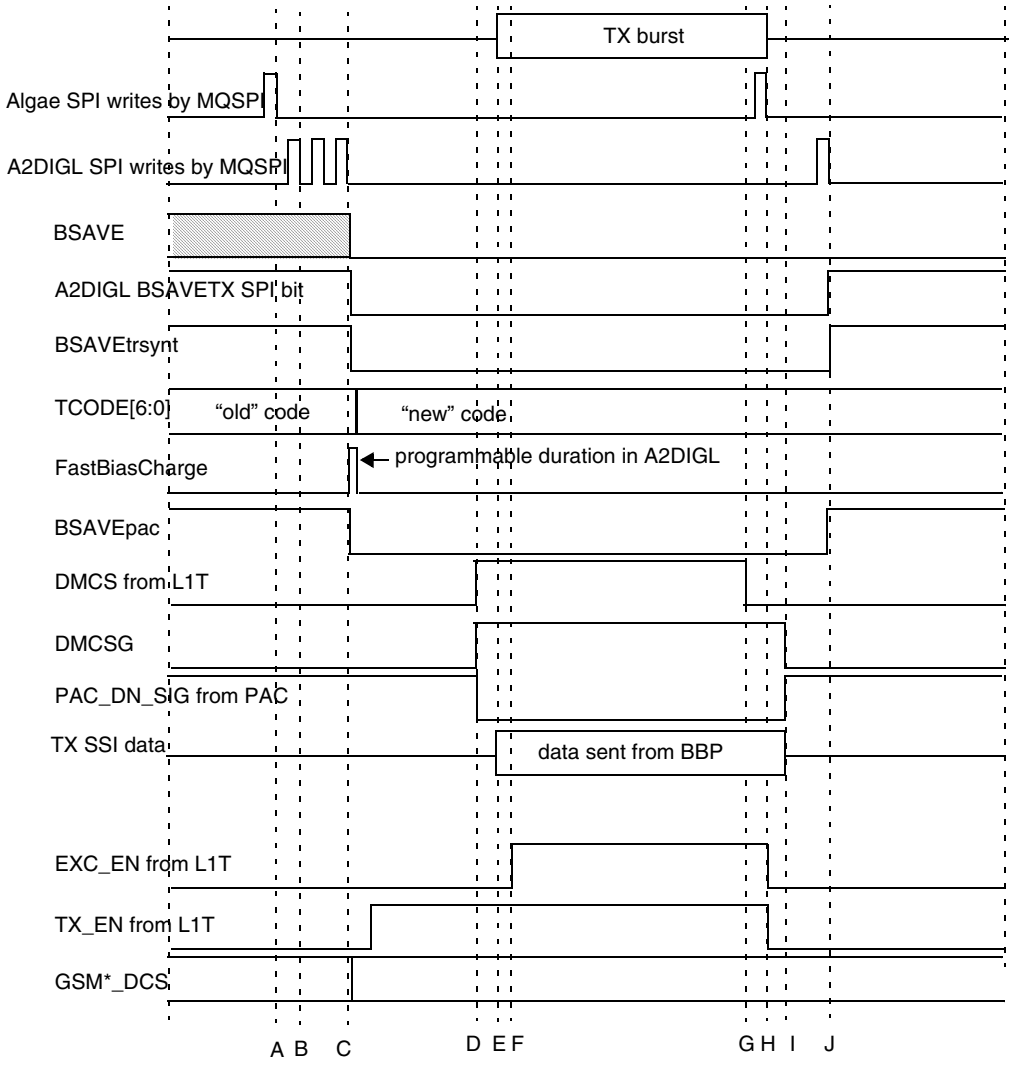


Figure 36-18. Transmit Timing Sequence, ABR_SEL=0, ABF_SEL=0

The next two figures show cases where ABR_SEL=1. ABR_SEL, if set, will delay the exit from battery save mode prior to the burst until DMCS goes high. For these cases, the BSAVEpac A2DIGL output signal does not go low with BSAVETX -- BSAVEpac goes low with the rising edge of DMCS.

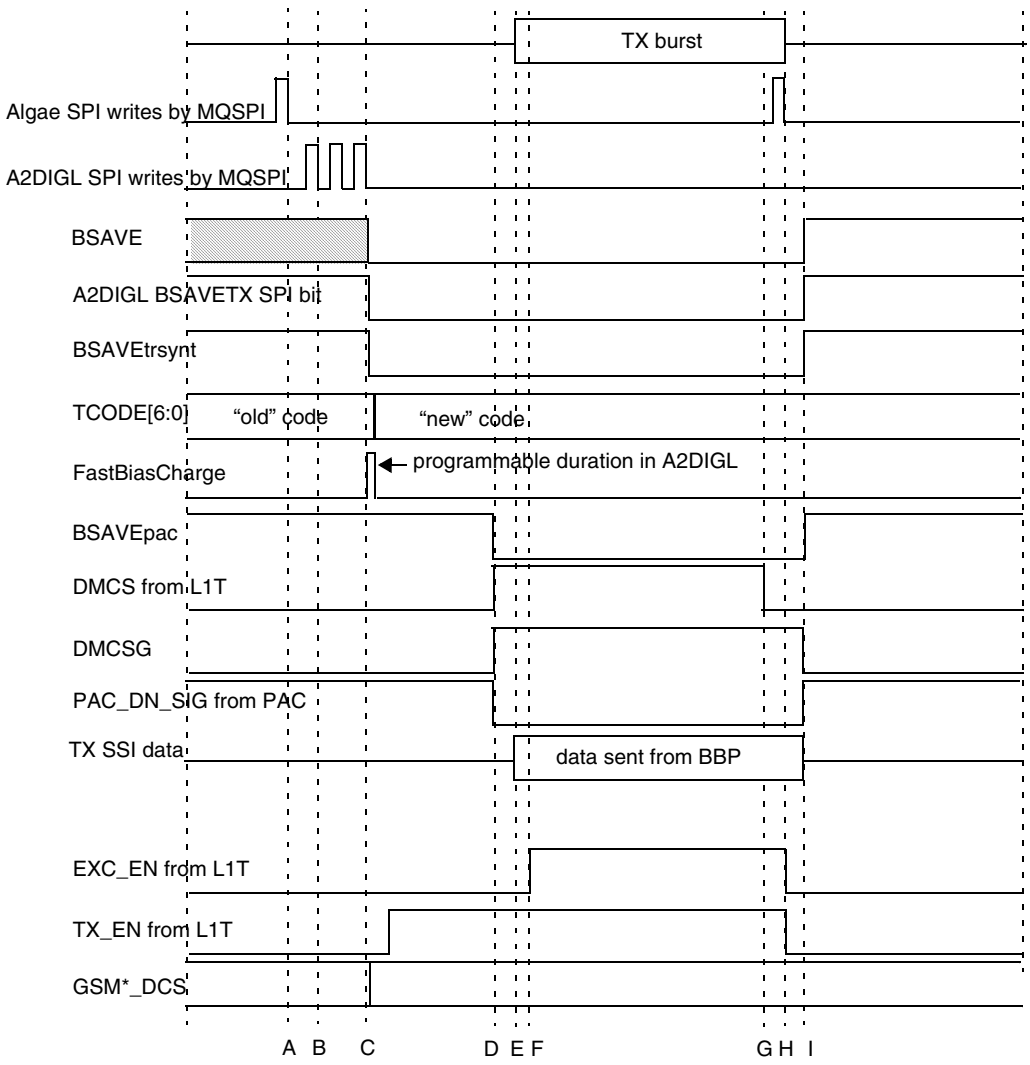


Figure 36-19. Transmit Timing Sequence, ABR_SEL=1, ABF_SEL=1

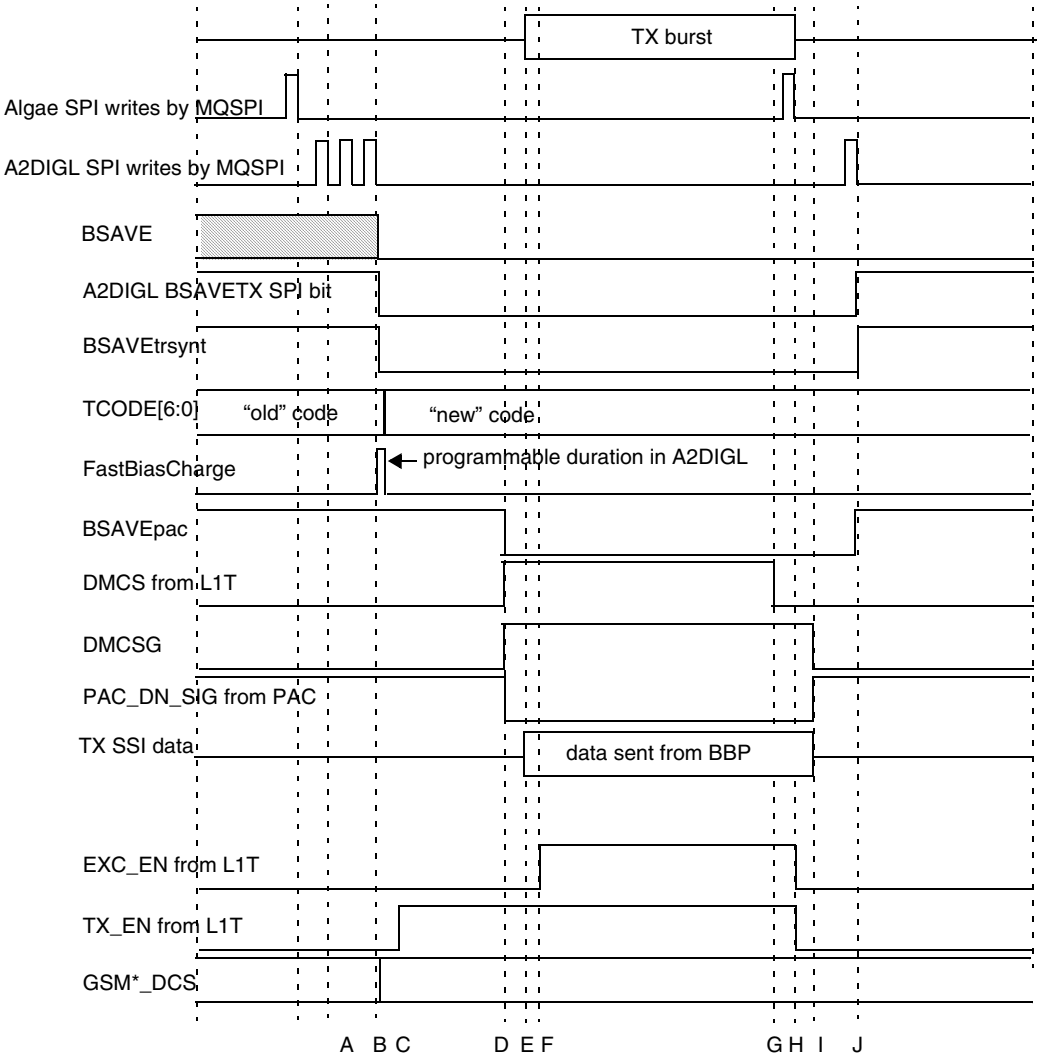


Figure 36-20. Transmit Timing Sequence, ABR_SEL=1, ABF_SEL=0

36.5.1.6 SPI bits and signals requiring additional logic

Most of the SPI bits correspond directly to signals output from the A2DIGL. The signal's logic levels correspond to the programming of the SPI bit, and they change value whenever the SPI bit is written. However, there are some SPI bits and related signals that require additional logic. These are listed below.

36.5.1.6.1 ADAPT_EN

This bit is used by the A2DIGL to enable or disable the DC adapt sequence for the receiver. For more information, see Chapter 36.5.1.5.1, "Receive VLIF mode timing sequences."

36.5.1.6.2 SL_TIMER[3:0] bits

These bits are used in the A2DIGL to determine the duration of the programmable synthesizer lock timer within A2DIGL. For more information, see Chapter 36.5.1.5.1, “Receive VLIF mode timing sequences.”

36.5.1.6.3 DCA_TIMER[3:0] bits

These bits are used to determine the duration of the programmable DC adapt timer within A2DIGL. For more information, see Chapter 36.5.1.5.2, “Detailed DC offset correction and SPI mux timing for VLIF.”

36.5.1.6.4 FastDigitalAdapt signal

This signal is generated by the A2DIGL to control the DCADAPT module. This signal is defined in more detail in Chapter 36.5.1.5.2, “Detailed DC offset correction and SPI mux timing for VLIF.” and Chapter 36.5.1.5.4, “Detailed DC offset correction and SPI mux timing for DCR.”

36.5.1.6.5 EndFastAdapt signal

This signal is generated by the A2DIGL to control the DCADAPT module. This signal is defined in more detail in Chapter 36.5.1.5.2, “Detailed DC offset correction and SPI mux timing for VLIF.” and Chapter 36.5.1.5.4, “Detailed DC offset correction and SPI mux timing for DCR.”

36.5.1.6.6 RX_EN_OUT signal

This signal is generated by the A2DIGL to turn on the Algae receiver. This signal is output to the RX_EN_OUT Neptune pin. This signal is defined in more detail in Chapter 36.5.1.5.1, “Receive VLIF mode timing sequences.” and Chapter 36.5.1.5.3, “Receive DCR mode timing sequence.”

36.5.1.6.7 Battery Save bits and signals

The A2DIGL outputs the following battery save signals:

- BSAVE - this is master battery save signal. This is connected to the RXAFE BSAVE pin. This follows the programming of the group 4 SPI BSAVE bit, but will be automatically cleared on the falling edge of the SPI chip select during SPI write to any group other than group 4. This signal will be set automatically at the end of a receive or transmit burst if AB_SEL or ABF_SEL are set, respectively.
- BSAVETX - this is the generic battery save signal used for TX. The TUNEC digital now use BSAVETX instead of BSAVE
- BSAVERX - this is the generic battery signal used for RX.
- BSAVEtrsynt - this enables the transmit and receive synthesizer. This is connected to the TX/TRSYNT BSAVE pin.
- BSAVErxafe - this signal enables the RXAFE sigma-delta. This is connected to the RXAFE BSAVERX pin.
- BSAVErxclk - this signal enables the RXAFE’s digrx_clk. This is connected to the RXAFE BSAVErxclk pin. This should be low whenever the RXCPROC, RXAFE, DCADAPT, or RXSDG need to be enabled. BSAVErxclk should be low 128 clocks longer than BSAVErxcproc.
- BSAVErxcproc - this signal enables the RXCPROC. This is connected to the RXCPROC BSAVERXCPROC pin.
- BSAVEpac - this signal is used to enable the PAC and the TX. This is connected to the PAC BSAVEpac pin, the TX BSAVE pin and also to the RXAFE BSAVETX pin.

The A2DIGL has three other SPI bits which are related to battery save but are not brought out of the A2DIGL module: BSAVETX, BSAVERX, AB_SEL, ABR_SEL, and ABF_SEL. BSAVETX and BSAVERX are used by A2DIGL to initiate transmit and receive sequences, respectively. AB_SEL is used by A2DIGL to determine when battery save mode will be re-entered at the end of a receive slot. ABR_SEL and ABF_SEL determine when battery save mode will be exited and re-entered, respectively, for transmit slot.

The PAC outputs a signal called PAC_DN_SIG to the A2DIGL. The A2DIGL BSAVE, BSAVETX, BSAVEtrsynt, BSAVEpac will go high on the rising edge of PAC_DN_SIG if the ABF_SEL SPI bit is set.

The battery save bits and signals are illustrated in more detail in Chapter 36.5.1.5.1, “Receive VLIF mode timing sequences.” Chapter 36.5.1.5.3, “Receive DCR mode timing sequence.” and Chapter 36.5.1.5.5, “Transmit Timing Sequence.”

36.5.1.6.8 GAINADxa[6:0], GAINADxb[6:0], GainLUPADR, GCSTEP

These SPI bits are used to store and select the GAINADa[6:0] and GAINADb[6:0] signals that are output to the RXCPROC. Two values for GAINADa {GAINAD0a, GAINAD1a} and two values for GAINADb {GAINAD0b, GAINAD1b} are stored in the A2DIGL. These are programmed by writing to the GAINADxa[6:0], GAINADxb[6:0], and GainLUPADR bits in SPI Group 3; the GainLUPADR bit selects which values {0=GAINAD0a,GAINAD0b, 1=GAINAD1a,GAINAD1b} are being stored. The SPI Group 5 parameter GCSTEP selects which values {0 or 1} will be output to GAINADa[6:0] and GAINADb[6:0], respectively.E.g, if GCSTEP=0 then GAINAD0a and GAINAD0b are output to GAINADa[6:0] and GAINADb[6:0], respectively.

36.5.1.6.9 PHASEADxa, PHASEADxb, PhLUPADR[1:0], PHASEADSEL

These SPI bits are used to store and select the PHASEADa[7:0] and PHASEADb[9:0] signals that are output to the RXCPROC. Four values for PHASEADa {PHASEAD0a, PHASEAD1a, PHASEAD2a, PHASEAD3a} and four values for PHASEADb {PHASEAD0b, PHASEAD1b, PHASEAD2b, PHASEAD3b} are stored in the A2DIGL. These are programmed by writing to the PHASEADxa[7:0], PHASEADxb[7:0], PHASEADxb_MSB[1:0], and PhLUPADR[1:0] bits in SPI Group 3; the PhLUPADR[1:0] bits select which values {00=PHASEAD0a,PHASEAD0b, 01=PHASEAD1a,PHASEAD1b, etc.} are being stored. The SPI Group 5 parameter PHASEADSEL[1:0] selects which values {0, 1, 2, or 3} will be output to PHASEADa[7:0] and PHASEADb[9:0], respectively.E.g, if PHASEADSEL[1:0]=’01’ then PHASEAD1a and PHASEAD1b are output to PHASEADa[7:0] and PHASEADb[9:0], respectively.

36.5.1.6.10 FastBiasCharge signal

For receive slots, the FastBiasCharge signal generated by A2DIGL has a fixed 10us duration. For more information, see Chapter 36.5.1.5.1, “Receive VLIF mode timing sequences.”

For transmit slots, the FBC_TD[3:0] bits are used to program the duration of this signal generated by A2DIGL. For more information, see Chapter 36.5.1.5.5, “Transmit Timing Sequence.”

36.5.1.6.11 Saturation detect reset (DET_RESET)

The DET_RESET bit is used to reset the saturation detector flag signal RXSDG_DET_FLAG_B (which can trigger an MCU interrupt). Once the RXSDG_DET_FLAG_B pin goes low then the DET_RESET bit MUST be written to clear the RXSDG_DET_FLAG_B pin. The set state of the RXSDG_DET_FLAG_B pin is to cause the DET_RESET bit to clear.

36.5.1.6.12 Dither reset (DITH_RESET)

The DITH_RESET bit is used to reset the random number generator in the dither circuit. The DITH_RESET bit will self clear after outputting a pulse of two 13MHz clock periods to the dither circuit.

36.5.1.6.13 Regulators Control

See Section 36.4.1.3, “Regulators Control.” for a discussion of the Regulator control SPI bits

36.5.1.7 SPI Interface Address Decoding

The address decoding is as follows, and a detailed description of the programming bits in each group is given in Section 36.5.1.10, “SPI Interface Bit Groups.”

- 0000 - group 1, Tx and general control, normally only programmed at power on reset.
- 0001 - group 2, Rx control, normally only programmed at power-on reset.
- 0010 - group 3, Rx control, normally only programmed at power-on reset.
- 0011 - group 4, normally only programmed when transmission standards change
- 0100 - group 5, bits that need to be changed on each receive and transmit timeslot
- 0101 - group 6, Predistortion parameters
- 0110 - group 7, TX,PAC parameters that need to be programmed before transmit slots
- 0111 - group 8, PAC parameters that need to be programmed before transmit slots
- 1000 - group 9, PAC control, normally only programmed at power-on reset
- 1001- group 10, PAC control, normally only programmed at power-on reset
- 1010- group 11, PAC control, normally only programmed at power-on reset
- 1011- group 12, ANATEST control
- 11xx- Groups 13-15 reserved

36.5.1.8 Receive and Transmit Modules Interface Register Summary

This section contains a summary of the memory-mapped registers associated with the Receive and Transmit Modules Interface.

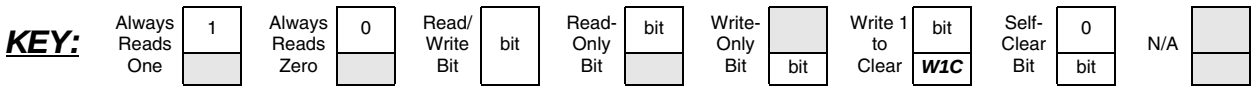


Table 36-7. Receive and Transmit Modules Interface Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PWRUNFLT $2484_6000	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	CDET_OUT	PWRUNFLT[9:0]									
	W																
PWRDAC $2484_6004	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	PWRDAC[9:0]									
	W																
ERRORSIG $2484_6008	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	ERRORSIG[9:0]										
	W																
MUXCTL $2484_600C	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	TxSEL3	MAGEN	TxSEL2	TxSEL1	RxSEL	TSEL	SSEL
	W																

36.5.1.9 Receive and Transmit Module Interface Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the Receive and Transmit Modules Interface registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and is cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

PWRUNFLT PWRUNFLT Register **Addr $2484_6000**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
						CDET _OUT	PWRUNFLT[9:0]									
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-8. PWRFLT Description

Name	Description	Settings
PWRUNFLT Bits 9-0	**PWRUNFLT —** PAC's AOC ACC input minus DC_OFFSET	
CDET_OUT Bits 10	**CDET_OUT —** PAC's Crossover detect value	

PWRDAC PWRDAC Register **Addr $2484_6004**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
							PWRDAC[9:0]									
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-9. PWRFLT Description

Name	Description	Settings
PWRDAC Bits 9-0	**PWRDAC —** Value of the PAC's AOC D/A input before DLPF	

ERRORSIG ERRORSIG Register **Addr $2484_6008**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
						ERRORSIG[9:0]										
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-10. PWRDAC Description

Name	Description	Settings
ERRORSIG Bits 10-0	**ERRORSIG —**Difference between the PAC's power ramp and ADC input value.	

MUXCTL MUXCTL Register **Addr $2484_600C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
										TxSEL3	MAGEN	TxSEL2	TxSEL1	RxSEL	TSEL	SSEL
TYPE	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Table 36-11. MUXCTL Description

Name	Description	Settings
SSEL Bit 0	**SSEL —**Selects muxing of the SPI input signals. This bit should be cleared for normal operation and set for the control mode.	0 - SPI signals from MQSPI 1 - SPI signals from GPIO
TSEL Bit 1	**TSEL —**Selects the muxing of the timing control input signals. This bit should be cleared for normal operation and set for the control mode.	0 - timing control signals from L1T 1 - timing control signals from GPIO
RxSEL Bit 2	**RxSEL —**Selects the muxing of the Rx SSI signals. This bit should be cleared for normal operation and set for the bypass mode.	0 - RxCPROC SSI muxed to BBP 1 - GPIO muxed to BBP
TxSEL1 Bit 3	**TxSEL1 —**Selects the muxing of the Tx SSI clock signal. This bit should be cleared for normal operation and set for the bypass mode.	0 - Tx SSI clock muxed to BBP Tx clock 1 - GPIO muxed to BBP Tx clock
TxSEL2 Bit 4	**TxSEL2 —**Selects the muxing of the Tx SSI data signal. This bit should be cleared for normal operation and set for the control mode.	0 - BBP data muxed to Tx SSI data 1 - GPIO muxed to Tx SSI data
MAGEN Bit 5	**MAGEN —**MAGIC Serial Communication Protocol Enable — Communication protocol selection.	0 = The standard RSI communication protocol is enabled for the BBP. 1 = The MAGIC Serial Communication Protocol is enabled on BBP pins and replaces the standard RSI communication protocol at this Port. The A2DIGL DMCSG signal is internally routed to the glue logic for this purpose.
TxSEL3 Bit 6	**TxSEL3 —** Selects the muxing of the GPIO_SCKB_CLK with the TX_TX_CLK/ GPIO_TX_CLK. This bit should be cleared for normal operation	0 = TX_TX_CLK or the GPIO_TX_CLK as selected by the TxSEL1, muxed out to the BBP's SCKB input. 1 = GPIO_SCKB_CLK muxed out to the BBP's SCKB input.

36.5.1.10 SPI Interface Bit Groups

This is a list of all of the SPI bit groupings and register descriptions for the modules which are now on Neptune and which came from MAGIC-LV, Tomahawk, or GCAP; there are also a few new SPI bits for other mixed-signal Neptune modules such as the TUNEC, TOSW and REGUL.

Table 36-12. Group 1 TX and General Control Description

Name	Module	Description
Bit 15-0		Reserved
ALG_DIS Bit 16 Reset Value: 0	A2DIGL	Block SPI signals from MQSPI to the ALGAE, when MQSPI is writing to A2DIGL. Does not affect the DCADAPT SPI writes to ALGAE.
RT_REG_EN Bit 17 Reset Value: 1	REG	Enable for RX/TX regulator
S_REG_EN Bit 18 Reset Value: 1	REG	Enable for Synthesizer regulator
AUD_REG_EN Bit 19 Reset Value: 1	REG	Enable for Audio codec regulator
ST_RT_REG Bit 20 Reset Value: 0	REG	Standby for RX/TX regulator. If high, the rt_reg is disabled when STANDBY is high,
ST_S_REG Bit 21 Reset Value: 0	REG	Standby for Synthesizer regulator. If high, the s_reg is disabled when STANDBY is high.
ST_AUD_REG Bit 22 Reset Value: 0	REG	Standby for Audio codec regulator. If high, the aud_reg is disabled when STANDBY is high.
ABF_SEL Bit 23 Reset Value: 0	PAC	If programmed high then BSAVEpac is set true after DN_DLY expires.
ABR_SEL Bit 24 Reset Value: 0	PAC	If programmed high then BSAVEpac is set false on the rising edge of DMCS.
TRKL_SEL[2:0] Bit 27-25 Reset Value:0	TRSYNT	Intentional leakage offset for the charge pumps.

Table 36-12. Group 1 TX and General Control Description (Continued)

Name	Module	Description
CPTEST[2:0] Bit 30-28 Reset Value: 0	TRSYNT	Places the charge pumps into test mode as follows: CPTEST[2:0] — CP operation (main and reference) 000 — Normal operation 001 — Reserved 010 — Reserved 011 — All Q-Pump o/ps to source current 100 — Reserved 101 — All Q-Pump o/ps to sink current 110 — Reserved 111 — Both source & sink Q-pumps are active
OSC26M_AGC[5:0] Bit 37-32 Reset Value: 0x3F	CROSC	6 bit input for AGC of oscillator amplitude
OSC26M_DISABLE Bit 38 Reset Value: 0	PAC	This signal is used to disable the PAC_ANA circuits specifically for EDGE more Transmission. The signal is active high and when set placed PAC_ANA circuits into BSAVE mode.
DAC_PAC_TST Bit 39 Reset Value: 0	PAC	Program high for testing the PAC's DAC
FBC_TD[3:0] Bit 43-40 Reset Value: 0	A2DIGL	Sets the duration of the FastBiasCharge signal from 0µs ('0000') to a maximum of 14µs ('1111') in 1µs increments. FBC_TD[3:0] = 0000, the TX fastbiascharge pulsewidth = zero µs FBC_TD[3:0] = 0001, the TX fastbiascharge pulsewidth = minimum pulsewidth (1/13 µs) FBC_TD[3:0] = 0010 to 1111, the TX fastbiascharge pulsewidth = 1µs to 14µs in increment of 1µs
MOD_INV Bit 44 Reset Value: 1	TX	If programmed to 1 the output of the modulation differential encoder is 'normal'. If programmed to a zero then the output is inverted.
FN_MODE Bit 45 Reset Value: 0	TX	If programmed low the data source for the FN modulation is from the GMSK look-up table. If programmed high the data source is from the predistortion system. Default at power up is low.
DP_INV Bit 46 Reset Value: 0	TX	If programmed high the sense of the data for the high frequency portion of the dual port modulation option is inverted.
Bit 47		Reserved
DP_OFS[9:0] Bits 57-48 Reset Value: 0	TX	dual_port programmable offset.
DP_MODE Bit 58 Reset Value: 0	TX	If programmed high the data source for the dual port D/A is from the pre-distortion system. If programmed low the data source is from the GMSK look up table.
DP_ON_OFFB Bit 59 Reset Value: 0	TX	If programmed high then the DP_DELAY function is enabled and there is a delay between the FN path and the dual port DAC path. This bit also activates the Dual Port D/A. If this bit is set low then the dual port system is disabled.

Table 36-12. Group 1 TX and General Control Description (Continued)

Name	Module	Description
OD[3:0] Bit 63-60 Reset Value:0x 6	TX	The programming input for the division factor from the output of the main divider to the oversample clock input of the GSM transmit modulator.
DP_TST_IN[9:0] Bit 73-64 Reset Value:0	TX	Used to test the dual-port DAC
DAC_9_TST Bit 74 Reset Value: 0	TX	Program high for testing the TX's 9bit DAC
Bit 75		Reserved
GRPSEL Bit 79-76		Group selection bits. Must be written to 0000 for selection of Group 1 parameters.

Table 36-13. Group2 RX Control 1 Description

Name	Module	Description
Bit 34-0		Reserved
STO_FORCE Bit 35 Reset Value: 0	A2DIGL	If STO_FORCE is set, it will force the tracking oscillator output on, independent of the receiver timing. IF STO_FORCE is clear, while STO_EN is set, the 26MHz clock will be enabled on the TRKG_OSC_OUT pin.
DCAPOL Bit 36 Reset Value: 0	DCadapt	Determines the polarity used by DCADAPT. If set, the data input to the DCADAPT module from RXAFE is inverted.
AGC_HIGH Bit 37 Reset Value: 0	DCadapt	If set, a fast load SPI write to Algae will occur at the beginning of the fast DC adapt to set the Algae AGC_HIGH bit high.At the end of the fast DC adapt, another fast load SPI write to Algae will clear it's AGC_HIGH bit.
LNA_OFF Bit 38 Reset Value: 0	DCadapt	If set, a fast load SPI write to Algae will occur at the beginning of the fast DC adapt to set the Algae LNA_OFF bit high.At the end of the fast DC adapt, another fast load SPI write to Algae will clear it's LNA_OFF bit.
SDD_EN Bit 39 Reset Value: 0	RXSDG	If programmed high then a psuedo random dither is enabled on the sigma delta modulator.
DET_LVL[2:0] Bit 42-40 Reset Value:0x 3	RXSDG	The programming input for the level comparison threshold of the amplitude detector on the sigma delta output. DET_LVL[2:0] — Threshold w.r.t.250mV peak 000 — -2.9dB 001 — -1.0dB 010 — +0.6dB 011 — +2.0dB 100-111 — +3.1dB

Table 36-13. Group2 RX Control 1 Description (Continued)

Name	Module	Description
DITH_DET_LVL Bit 43 Reset Value: 0	RXSDG	If programmed high then the threshold to move the dither level is set at –5.4dB from the full scale of the sigma delta input. If the bit is programmed low then the threshold is set to –8.9dB from the full scale of the sigma delta input.
DITH_LVL[1:0] Bit 45-44 Reset Value: 0	RXSDG	Selects the differential dither levels for the sigma delta converter when the level selected by DITH_DET_LVL is not exceeded. If DITH_DET_LVLis exceeded then the dither is set to zero. DITH_LVL[1:0] — Dither Level 00 — +/-30mV 01 — +/-60mV 10 — +/-90mV 11 — +/-120mV
ACC2 Bit 46 Reset Value:0	TRSYNT	If programmed high then two accumulators are used instead of three.
ACC0 Bit 47 Reset Value:0	TRSYNT	If programmed low then all accumulators are disabled. (Overrides ACC2)
SL_TIMER[3:0] Bit 51-48 Reset Value: 0	A2DIGL	Programs duration of the synthesizer lock timer to 0 (test), or from 55us to 125us in increments of 5us. SL_TIMER[3:0] — Timer (usec) 0000 — 0 0001 — 55 0010 — 60 0011 — 60 0100 — 70 0101 — 75 0110 — 80 0111 — 85 1000 — 90 1001 — 95 1010 — 100 1011 — 105 1100 — 110 1101 — 115 1110 — 120 1111 — 125

Table 36-13. Group2 RX Control 1 Description (Continued)

Name	Module	Description
DCA_TIMER[3:0] Bit 55-52 Reset Value: 0	A2DIGL	Programs duration of the DC adapt timer from 30us to 58us in increments of 2us. Note that this time includes 10us for the FastBiasCharge pulse, so the duration of the FastDigitalAdapt pulse is actually 10us less than the time specified by DCA_TIMER[3:0]. For development/characterization, it is possible to force the timer to never expire by setting DCA_TIMER to all ones. DCA_TIMER[3:0] — Timer (usec) 0000 — 30 0001 — 32 0010 — 34 0011 — 36 0100 — 38 0101 — 40 0110 — 42 0111 — 44 1000 — 46 1001 — 48 1010 — 50 1011 — 52 1100 — 54 1101 — 56 1110 — 58 1111 — infinite
TOL[1:0] Bit 57-56 Reset Value: 0	RXcproc	Selects three different compensations of the droop digital filter. This is used to compensate to low, nominal, or high bandwidth of the analog filters which preceed the sigma delta converter. TOL[1:0] — Analog filter compensated 00 — Nominal 01 — Minimum BW 10 — Maximum BW
THP1[2:0] Bits 60-58 Reset Value: 0x3	RXcproc	Programs the time from the end of the fast DC adapt to the end of the time at which the high value corner FHP1 is selected in DCR mode. THP1[2:0] — High Pass Corner Freq 000 — 10/f2X 001 — 20/f2X 010 — 30/f2X 011 — 40/f2X 100 — 50/f2X 101 — 60/f2X 110 — 70/f2X 111 — 80/f2X
THP2[2:0] Bits 63-61 Reset Value: 0x3	RXcproc	Programs the time during which the medium value corner FHP2 is selected. THP2[2:0] — High Pass Corner Freq 000 — 10/f2X 001 — 20/f2X 010 — 30/f2X 011 — 40/f2X 100 — 50/f2X 101 — 60/f2X 110 — 70/f2X 111 — 80/f2X

Table 36-13. Group2 RX Control 1 Description (Continued)

Name	Module	Description
FHP1[2:0] Bits 66-64 Reset Value: 0	RXcproc	Programs the high pass value of the first step of the fine digital DC adapt in DCR mode. FHP1[2:0] — High Pass Corner (GSM) 000 — 11.5kHz 001 — 5.57kHz 010 — 2.74kHz 011 — 1.36kHz 100 — 678Hz 101 — 338Hz 110 — Clear the DC correction 111 — Hold the DC correction
CP_LV Bit 67 Reset Value: 1	TRSYNT	Sets the charge pump output voltage level. 0=high voltage charge pump output. 1=low voltage charge pump output.
FHP2[3:0] Bits 71-68 Reset Value: 1	RXcproc	Programs the high pass corner value of the second step of the fine digital DC adapt in DCR mode.
FHP3[3:0] Bits 75-72 Reset Value: 0xF	RXcproc	Programs the high pass corner value of the last step of the fine digital DC adapt in DCR mode. FHP2[3:0] or FHP3[3:0] — High Pass Corner (GSM) 0000 — 5.57kHz 0001 — 2.74kHz 0010 — 1.36kHz 0011 — 678Hz 0100 — 338Hz 0101 — 169Hz 0110 — 84.7Hz 0111 — 43.8Hz 1000 — 23.8Hz 1001 — 10.9Hz 1010 — 5.94Hz 1011 — 2.64Hz 1100 — 1.32Hz 1101 — 0.68Hz 1110 — Clear DC correction 1111 — Hold DC correction
GRPSEL Bit 79-76		Group selection bits. Must be written to 0001 for selection of Group 2 parameters

Table 36-14. Group3 RX Control 2 Description

Name	Module	Description
Bit 35-0		Reserved
PHASEADxb_MSB[1:0] Bit 37-36 Reset Value: 0	RXcproc	These are the MSB bits for the PHASEADxb register. To have backward software code compatibility these additional bits were placed here. (See also PHASEADxb description)

Table 36-14. Group3 RX Control 2 Description (Continued)

Name	Module	Description
HDFO Bit 38 Reset Value: 1	RXcproc	If programmed high then the digital high pass filter prior to the complex multiplier operates in continuous mode. If programmed low then the filter operates in hold mode.
GainLUPADR Bit 39 Reset Value: 0	RXcproc	There are two pairs of internal storage registers used for the phased values of gain correction versus AGC. This SPI group selects which one of the two registers are written in the corresponding SPI initialization write. GainLUPADR — Gain correction value written 0 — GAINAD0a/b 1 — GAINAD1a/b
GAINADxa[6:0] Bit 46-40 Reset Value: 0	RXcproc	Signed Gain error adjust for image rejection at IF passband frequency A. There are two registers corresponding to this SPI group. The register used in radio operation is selected by GCSTEP. The register written at initialization or mode change is selected by GainLUPADR.
TESTByp Bit 47 Reset Value: 0	RXcproc	If programmed high then the digital mixer and filter of the IF section are bypassed to allow external digital filtering by the DSP.
GAINADxb[6:0] Bit 54-48 Reset Value: 0	RXcproc	Signed Gain error adjust for image rejection at IF passband frequency B. There are two registers corresponding to this SPI group. The register used in radio operation is selected by GCSTEP. The register written at initialization or mode change is selected by GainLUPADR.
IQSWAP Bit 55 Reset Value: 0	RXcproc	If this bit is programmed low then the I bits are sent first on SDFS. If programmed high then the Q bits are sent first.
PHASEADxa[7:0] Bit 63-56 Reset Value: 0	RXcproc	Signed Phase error adjust for image rejection at IF passband frequency A. There are four registers corresponding to this SPI group. The register used in radio operation is selected by PHASEADSEL[1:0] and is linked to the RF frequency selected by the radio software. The register written at initialization is selected by PhLUPADR[1:0]
PHASEADxb[7:0] Bit 71-64 Reset Value: 0	RXcproc	Signed Phase error adjust for image rejection at IF passband frequency B. There are four registers corresponding to this SPI group. The register used in radio operation is selected by PHASEADSEL[1:0] and is linked to the RF frequency selected by the radio software. The register written at initialization is selected by PhLUPADR[1:0]. The overall size of the PHASEADxb is 10 bits {PHASEADxb_MSB[1:0], PHASEADxb[7:0]} (See also PHASEADxb_MSB description)
SIGN[1:0] Bit 73-72 Reset Value: 0	RXcproc	Program any possible mixing combination for I and Q outputs in the balanced complex multiplier SIGN[1:0] — Combination 00 — (I+jQ)*exp(+j*2*pi*DFI*t) 01 — (I+jQ)*exp(-j*2*pi*DFI*t) 10 — (I-jQ)*exp(+j*2*pi*DFI*t) 11 — (I-jQ)*exp(-j*2*pi*DFI*t)

Table 36-14. Group3 RX Control 2 Description (Continued)

Name	Module	Description
PhLUPADR[1:0] Bit 75-74 Reset Value: 0	RXcproc	There are four pairs of internal storage registers used for the phased values of phase correction versus RF frequency. This SPI group selects which one of the four registers are written in the corresponding SPI initialization write. PhLUPADR[1:0] — Phase correction value written 00 — PHASEAD0a/b 01 — PHASEAD1a/b 10 — PHASEAD2a/b 11 — PHASEAD3a/b
GRPSEL Bit 79-76		Group selection bits. Must be written to 0010 for selection of Group 3 parameters

Table 36-15. Group 4 Fields that are expected to change with mode change

Name	Module	Description
Bit 54-0		Reserved
DAZERO Bit 55 Reset Value: 0	PAC	If set to a logic high then the output of the smoothing filter is shorted out after the DN_DLY timer has expired.
AOC_FBW[1:0] Bit 57-56 Reset Value: 0	PAC	Selects the bandwidth of the analog filter which follows the AOC D/A. 00 50kHz 01 100kHz 10 500kHz 11 1MHz
FG[2:0] Bit 60-58 Reset Value: 0x3	DCadapt	Gain of the digital low pass filter in digital adapt mode. FG[2:0] Gain 000 2^{-5} 001 2^{-6} 010 2^{-7} 011 2^{-8} nominal GSM setting 100 2^{-9} 101 2^{-10} 110 2^{-11} 111 2^{-12}
BSAVE Bit 61 Reset Value: 1	A2DIGL	Program low for normal operation. If BSAVE is programmed high then the reference voltages generated by RXAFE will be in a low bias current standby mode. BSAVE will revert to a logic low on the next CE transition from high to low, regardless of which SPI segment is programmed.
MA_SELECT[1:0] Bit 63-62 Reset Value: 0	RXcproc	Selects the radio standard used. This selects the order of the digital filters for each standard as well as the DCR or VLIF mode receive timing. 00 GSM & VLIF mode receive timing 01 EDGE & VLIF mode receive timing 10 DCR wide & DCR mode receive timing 11 VLIF mode receive timing by default
DIF[6:0] Bit 70-64 Reset Value: 0x36	RXcproc	Selects the digital IF local oscillator. The bandwidth is DIF *2.116kHz in GSM mode.
SDM_Z Bit 71 Reset Value: 1	RXAFE	If programmed high then some of the zeroes of the sigma delta modulator are moved to non-zero frequency. If programmed to zero then all of the zeros of the sigma delta modulator are located at DC.
DBW[2:0] Bit 74-72 Reset Value: 0x5	RXcproc	Digital Filter bandwidth select. DBW[2:0] 3dB bandwidth 000 67kHz (GSM) 001 71kHz (GSM) 010 75kHz (GSM) 011 79kHz (GSM) 100 83kHz (GSM) 101 90kHz (GSM) 110 100kHz (GSM) 111 not used

Table 36-15. Group 4 Fields that are expected to change with mode change (Continued)

Name	Module	Description
FBYP Bit 75 Reset Value: 0	RXcproc	If programmed high then the RXcproc will bypass the 14th order filter
GRPSEL Bit 79-76		Group selection bits. Must be written to 0011 for selection of Group 4 parameters

Table 36-16. Group 5 Fields expected to change with channel selection

Name	Module	Description
MNCP_INV Bit 31 Reset Value: 0	TRSYNT	If programmed to 0, the charge pump will source current to increase frequency. If programmed to 1, the charge pump will sink current to increase frequency.
RXCLK_SEL Bit 1 Reset Value: 0	RXAFE	Selects the clock source for the RxAFE. When set high the corrected clock "ccm_sync_patrefx2_doze_wait_clk" is used and when set low the Oscillator clock "osc26m_rx_clk26n,p" is used in RXAFE. Both clocks run at 26MHz.
BSAVERX Bit 2 Reset Value: 1	A2DIGL	If programmed high then the receiver is placed in standby mode. When programmed low, the receive timing sequence will be initiated.
BSAVETX Bit 3 Reset Value: 1	A2DIGL	If programmed high then the AOC is placed in standby mode. When programmed low, the transmit timing sequence will be initiated.
GSM*_DCS Bit 4 Reset Value: 0	GPIO	The GSM*_DCS pin, which is used as a band select signal, will follow the programming of this bit.
ADAPT_EN Bit 5 Reset Value: 1	A2DIGL	Program high to perform DC Offset cancellation for receive slots.
STO_EN Bit 6 Reset Value: 0	TOSW	While set, the 26MHz clock will be enabled on the TRKG_OSC_OUT pin. This bit will be cleared when the synth lock timer expires in the receive timing sequence.
AB_SEL Bit 7 Reset Value: 1	A2DIGL	If programmed high then when RX_ACQ goes high to low, BSAVE, BSAVERX, BSAVErxafe, and BSAVEtrsynt will be set/go high, and RX_EN_OUT will go low. Software will need to insure that the last valid data has been transferred from the RX interface before switching RX_ACQ.

Table 36-16. Group 5 Fields expected to change with channel selection (Continued)

Name	Module	Description
BBIQLEN[3:0] Bit 11-8 Reset Value: 0xE	RXcproc	Configures the BB serial interface for 8-bit, 12bit, or 16 bit I/Q data transfers. Note that 8-bit and 12 bit I/Q data is just the 16-bit I/Q data with 4 bits or 8 bits discarded. The dynamic range of the converter will result in 14 bits of valid data in wide(>60kHz) bandwidths and 15 bits in narrow(10-20kHz) bandwidths. Note that discarding of MSBs must only be done only in conjunction with the occurrence of low signal levels of the total received signal at the RX A/D converter to avoid clipping as this would significantly degrade receiver performance. BBIQLEN[3:0] — Word Width 0001 — 8 bits (remove 8 MSBs) 0010 — 8 bits (remove 7 MSBs & 1 LSB) 0011 — 8 bits (remove 6 MSBs & 2 LSBs) 0100 — 8 bits (remove 5 MSBs & 3 LSBs) 0101 — 8 bits (remove 4 MSBs & 4 LSBs) 0110 — 8 bits (remove 3 MSBs & 5 LSBs) 0111 — 8 bits (remove 2 MSBs & 6 LSBs) 1000 — 8 bits (remove 1 MSB & 7 LSBs) 1001 — 8 bits (remove 8 LSBs) 1010 — 12 bits (remove 4 MSBs) 1011 — 12 bits (remove 3 MSBs & 1 LSB) 1100 — 12 bits (remove 2 MSBs & 2 LSBs) 1101 — 12 bits (remove 1 MSB & 3 LSBs) 1110 — 12 bits (remove 4 LSBs) 1111 — 16 bits
PHASEADSEL[1:0] Bit 13-12 Reset Value: 0	RXcproc	These bits select which phase correction values to apply for image rejection. It is assumed that these bits are correlated to frequency ranges of the channel selection to account for variances in phase balance over the entire frequency band of the radio. The phase correction values are loaded into a lookup table using the PhLUPADR and the PHASEAD bits in Group 1. The phase correction values are addressed by PHASEADSEL as follows: PHASEADSEL[1:0] — Phase correction value 00 — PHASEAD0a/b 01 — PHASEAD1a/b 10 — PHASEAD2a/b 11 — PHASEAD3a/b
2X_SEL Bit 14 Reset Value: 0	RXcproc	Selects the output rate of the A/D as 1X per GSM bit time if programmed low or 2X per GSM bit time if programmed high.
GCSTEP Bit 15 Reset Value: 0	RXcproc	Gain correction step. This bit is used to step between two programmed values of the gain correction of the Balanced Complex Modulator. It is assumed that factory calibration of two settings of the PMA_AGC and AMP_AGC have been determined which provide the desired overall AGC step. When PMA_AGC and AMP_AGC are programmed between these two settings then this bit is used to shift the gain correction. GCSTEP — Gain correction value 0 — GAINAD0a/b 1 — GAINAD1a/b
NUM[23:0] Bit 39-16 Reset Value (in hex): 4EC4ED	TRSYNT	The numerator for the 24 bit fractional N divider system. (The denominator is fixed at 2^{24} = 16,777,216)

Table 36-16. Group 5 Fields expected to change with channel selection (Continued)

Name	Module	Description
N[4:0] Bit 44-40 Reset Value: 0	TRSYNT	Fractional N pulse swallow parameter N. Programmed frequency of the synthesizer is $(N*5 + A + 3 + NUM)2^{24})*f_{ref}$
A[2:0] Bit 47-45 Reset Value: 0	TRSYNT	Fractional N pulse swallow parameter A.
TX_VCO_DIV Bit 48 Reset Value: 0	TRSYNT	If set, the Tx VCO divider will be 2. If cleared, the divider will be 1.
TR_SEL Bit 49 Reset Value: 0	TRSYNT	Activates CP_RX if programmed to 0 and activates CP_TX if programmed to 1. The CP which is not selected is floating.
DET_RESET Bit 50 Reset Value: 1	RXSDG	If programmed high then the RXSDG_DET_FLAG_B pin is reset. Once the RXSDG_DET_FLAG_B pin is clear then this bit MUST be written to set the pin. The set state of the RXSDG_DET_FLAG_B pin is to cause the DET_RESET bit to clear.
DITH_RESET Bit 51 Reset Value: 0	RXSDG	Program high to reset the random number generator in the dither circuit. This bit will self-clear after outputting a pulse of two 13MHz clock periods to the dither circuit.
SAT_EN Bit 52 Reset Value: 0	PAC	Logic high activates saturation detection. Logic low disables this feature.
DET_HIGHV Bit 53 Reset Value: 0	PAC	This bit should be set if DETECT_IN voltage is <2.5V to prevent damage to the detector input circuit.
SOFT_SAT Bit 54 Reset Value: 0	PAC	When set to 1, the AOC_MAX_sig will update as described in the specfication, when set to 0 the AOC_MAX_sig is blocked
CDMA_DN Bit 55 Reset Value: 0	PAC	If programmed high then when DMCS goes from high to low the feedback loop will revert to ERRGain1, and PROP_Gain1
OFS[7:0] Bit 63-56 Reset Value: 0	PAC	Offset power level for systems which use the internal look up table for PA ramp shaping.(This parameter is taken from the phasing table stored in flash).
PWR[9:0] Bit 73-64 Reset Value: 0	PAC	Final power level desired for systems which use the internal look up table for PA ramp shaping.
AOC_GAIN[1:0] Bit 75-74 Reset Value: 0	PAC	Selects the gain of the differential amplifier in front of the AOC A/D converter. 00 Gain=1 01 Gain=3 10 or 11 Gain=10

Table 36-16. Group 5 Fields expected to change with channel selection (Continued)

Name	Module	Description
GRPSEL Bit 79-76		Group selection bits. Must be written to 0100 for selection of Group 5 parameters.

Table 36-17. Group 6 pre-distortion

Name	Module	Description
x3d(P_ADR)[15:0] Bit 15-0 Reset Value: 0	TX	The GSM predistortion coefficient for the third derivative of the modulation frequency. (If P_LOAD=1)
PD_STEP Bit 16 Reset Value: 1	TX	If programmed low then the pre-distortion waveform generator will hold the last value upon entering a data state of all 0 or all 1. If programmed high then the output modulation will be forced to +/-67kHz regardless of the last state of the waveform generator.(If P_LOAD=1)
Bits 23-17		Reserved
P_LOAD Bit 24 Reset Value: 0	TX	If programmed high then this SPI group. Loads the GSM predistortion coefficients. Default at power up is low.
P_ADR[2:0] Bit 27-25 Reset Value: 0	TX	Selects which group of predistortion coefficients are to be loaded, if P_LOAD is high.
x2d(P_ADR)[19:0] Bit 47-28 Reset Value: 0	TX	The GSM predistortion coefficient for the second derivative of the modulation frequency. (If P_LOAD=1
x1d(P_ADR)[17:0] Bit 65-48 Reset Value: 0	TX	The GSM predistortion coefficient for the first derivative of the modulation frequency. (If P_LOAD=1)
Bits 75-66		Reserved
GRPSEL Bit 79-76		Group selection bits. Must be written to 0101 for selection of Group 6 parameters.

Table 36-18. Group 7 TX,PAC parameters which must be programmed before a TX burst

Name	Module	Description
Bit 0		Reserved
DP_FN_DACb Bit 1 Reset Value: 0	TX	Selects which path is delayed by DP_DELAY. If programmed high then the FN path is delayed. If programmed low then the dual port DAC path is delayed.
PACLC[3:0] Bit 5-2 Reset Value: 0	PAC	PAC limit cycle correction value. NOTE: ***This is not currently used by the PAC module.***
DP_DELAY[1:0] Bit 7-6 Reset Value: 0	TX	Selects the delay of the dual port modulation path relative to the main FN PLL path. DP_DELAY[1:0] — Oversample Clocks 00 — 1 01 — 2 10 — 3 11 — 4

Table 36-18. Group 7 TX,PAC parameters (Continued)which must be programmed before a TX burst

Name	Module	Description
DP_GAIN[5:0] Bit 13-8 Reset Value: 0	TX	Selects the gain of the dual port modulation system.
AOC_HOLD Bit 14 Reset Value: 0	PAC	If programmed high then the clock from the AOC accumulator is removed thus forcing a hold state when DIV_DLY expires.
Proff Bit 15 Reset Value: 0	PAC	If programmed high then the proportional gain branch is disabled.
ERR_Gain1[3:0] Bits 19-16 Reset Value: 0	PAC	Selects the number of bit shifts right of the error signal fed to the AOC accumulator input before BW_DLY expires. 0000 Gain=1 0001 Gain=1/2 0010 Gain=1/4 0011 Gain=1/8 0100 Gain=1/16 0101 Gain=1/32 0110 Gain=1/64 0111 Gain=1/128 1000 Gain=1/256 1001 Gain=1/512 1010 Gain=1/1024 1011 Gain=1/2048 1100 through 1111 Gain=1/4096
ERR_Gain2[3:0] Bits 23-20 Reset Value: 0	PAC	Selects the number of bit shifts right of the error signal fed to the AOC accumulator input after BW_DLY expires. (Gain selection is the same as above.)
SAT_STEP[4:0] Bit 28-24 Reset Value: 0	PAC	AOC negative slope when saturated.(This parameter is taken from the phasing table stored in flash).
RAMP_SEL[2:0] Bits 31-29 Reset Value: 0	PAC	Selects the look up ramp shape. 000 Raised Cosine 001 Half Raised Cosine others not used
SAT_DLY[13:0] Bit 45-32 Reset Value: 0	PAC	Sets the number of PAC clocks from the rising edge of DMCS to enabling saturation detection. (This parameter is taken from the phasing table stored in flash).
ACT_EN Bit 46 Reset Value: 0	PAC	Logic high activates activity detection and initial linear ramping. Logic low deactivates this feature.
ACT_SEL Bit 47 Reset Value: 0	PAC	Logic high selects the digital activity comparator, logic low selects the analog activity comparator.
SAT_TH[9:0] Bit 57-48 Reset Value: 0	PAC	Threshold of error between detected power and ramp shape waveform which activates saturation detection. LSB justified to the error word. (This parameter is taken from the phasing table stored in flash).

Table 36-18. Group 7 TX,PAC parameters (Continued)which must be programmed before a TX burst

Name	Module	Description
PROPGain1[3:0] Bits 61-58 Reset Value: 0	PAC	Selects the number of bit shifts right of the error signal fed to the AOC accumulator output before PROP_DLY expires. 0000 Gain=1 0001 Gain=1/2 0010 Gain=1/4 0011 Gain=1/8 0100 Gain=1/16 0101 Gain=1/32 0110 Gain=1/64 0111 Gain=1/128 1000 Gain=1/256 1001 Gain=1/512 1010 Gain=1/1024 1011 Gain=1/2048 1100 through 1111 Gain=1/4096
PROPGain2[3:0] Bits 65-62 Reset Value: 0	PAC	Selects the number of bit shifts right of the error signal fed to the AOC accumulator output before PROP_DLY expires. 0000 Gain=1 0001 Gain=1/2 0010 Gain=1/4 0011 Gain=1/8 0100 Gain=1/16 0101 Gain=1/32 0110 Gain=1/64 0111 Gain=1/128 1000 Gain=1/256 1001 Gain=1/512 1010 Gain=1/1024 1011 Gain=1/2048 1100 through 1111 Gain=1/4096
ACT_TH[9:0] Bits 75-66 Reset Value: 0	PAC	Threshold of detected power to insure closed loop PA control. LSB justified to the PA detected word.(This parameter is taken from the phasing table stored in flash).
GRPSEL Bits 79-76		Group selection bits. Must be written to 0110 for selection of Group 7 parameters.

Table 36-19. Group 8 PAC parameters which must be programmed before a TX burst

Name	Module	Description
GSM_INIT[9:0] Bits 9-0 Reset Value: 0	PAC	Slope of the initial linear AOC drive ramp in GSM mode.
EST_DLY[13:0] Bits 23-10 Reset Value: 0	PAC	Sets the number of PAC clocks from the rising edge of DMCS to the end of the estimation state
ACC_INIT[9:0] Bits 33-24 Reset Value: 0	PAC	Estimation state value for the D/A.

Table 36-19. Group 8 PAC parameters (Continued)which must be programmed before a TX burst

Name	Module	Description
RAMP_DLY[13:0] Bits 47-34 Reset Value: 0	PAC	Sets the number of PAC clocks from the rising edge of DMCS to the start of the ramp up in lookup mode.
INIT_DLY[13:0] Bits 61-48 Reset Value: 0	PAC	Sets the number of PAC clocks from the rising edge of DMCS to the start of the estimation state.
DIV_DLY[13:0] Bits 75-62 Reset Value: 0	PAC	Sets the number of PAC clocks from the rising edge of DMCS to the point at which the AOC accumulator clock division is switched from ACC_DIV1 to ACC_DIV2.
GRPSEL Bit 79-76		Group selection bits. Must be written to 0111 for selection of Group 8 parameters.

Table 36-20. Group 9 PAC Control

Name	Module	Description
CDET_DLY[17:0] Bits 17-0 Reset Value: 0	PAC	Sets the number of clocks from DMCS to the time when the PAC readback parameters are updated. It also sets the number of clocks from DMCS rising edge to triggering the latching of cross-over detection.
Bits 31-18		Reserved
SSI_DLY[13:0] Bits 45-32 Reset Value: 0	PAC	Sets the number of PAC clocks from the rising edge of DMCS to the 1st SSI TX_CLK out of the TX module.
BACKWARD Bit 46 Reset Value: 0	PAC	Applies PWR_RAMP word directly to AOC DAC input
DET_FLT Bit 47 Reset Value: 0	PAC	Selects bandwidth of detector filter. If cleared, the 3dB cutoff frequency is 500KHz (min). If set, the 3dB cutoff frequency is 1.3MHz (typ), 1.0MHz (min).
RAMP_DN_DLY[13:0] Bits 61-48 Reset Value: 0	PAC	Sets the number of clocks from the falling edge of DMCS to the beginning of the ramp down.
Bits 63-62		Reserved
AOC_MAX[9:0] Bits 73-64 Reset Value: 0	PAC	Maximum permitted value of the AOC D/A. If the value from the accumulator attempts to exceed this value then AOC_MAX is held at the D/A inputs. Set to 3FF to disable. (This parameter is taken from the phasing table stored in flash).
Bits 75-74		Reserved
GRPSEL Bit 79-76		Group selection bits. Must be written to 1000 for selection of Group 9 parameters.

Table 36-21. Group 10 PAC Control

Name	Module	Description
Bits 15-0		Reserved
PAC_DLY[11:0] Bits 27-16 Reset Value: 0	PAC	Sets the number of PAC clocks from the rising edge of DMCS to triggering the reading the PACII DC offset.
Bits 31-28		Reserved
PROP_DLY[13:0] Bits 45-32 Reset Value: 0	PAC	Sets the number of PAC clocks from the rising edge of DMCS to the point at which the proportional gain is changed from PROPGain1 to PROPGain2.
Bits 47-46		Reserved
BW_DLY[13:0] Bits 61-48 Reset Value: 0	PAC	Sets the number of PAC clocks from the rising edge of DMCS to the point at which the loop gain is switched from ERR_Gain1 to ERR_Gain2.
Bits 63-62		Reserved
ACC_REF Bits 64 Reset Value: 0	ANATEST	It is used as the tcm_anatest_en0 signal.
Bits 75-65		Reserved
GRPSEL Bit 79-76		Group selection bits. Must be written to 1001 for selection of Group 10 parameters.

Table 36-22. Group 11 PAC Control

Name	Module	Description
Bits 15-0		Reserved
AOC_FORCE[9:0] Bits 25-16 Reset Value: 0	PAC	Value to which the AOC is forced if AOC_MANUAL is programmed high.
RAMP_REF[5:0] Bits 31-26 Reset Value: 0	PAC	Sets the division ratio from the PAC clock to the look up table ramp clock.
DN_DLY[13:0] Bits 45-32 Reset Value: 0	PAC	Sets the number of PAC clocks from the falling edge of DMCS to power down of the TX section.
Bits 47-46		Reserved
Bits 61-48		Reserved.
LOW_BATT_EN Bit 62 Reset Value: 0	PAC	If set, the AOC_OUT pin will be grounded when LOW_BATT* pin is low.
Bit 63		Reserved
ADI_REF[6:0] Bits 70-64 Reset Value: 0	PAC	Sets the division ration from the clock to the internal A/D clock. NOTE: ***These bits are also used by the PAC AOC_REF. Specifically, AOC_REF[11:0] = {0,0,0,0,0,ADI_REF[6:0]}.***
AOC_MANUAL Bit 71 Reset Value: 0	PAC	If set high then AOC_FORCE is loaded continuously to the AOC D/A. If ACT_EN is also high then AOC_MANUAL is overridden when the detected power exceeds ACT_TH.
Bits 75-72		Reserved
GRPSEL Bit 79-76		Group selection bits. Must be written to 1010 for selection of Group 11 parameters.

Table 36-23. Group 12 ANATEST Control[1]

Name	Module	Description
Bit 23-0		Reserved
TST_DC[5:0] Bits 29-24 Reset Value: 0	ANAtest	Bits [3:0] are for Anatest Control Mux1— Select one out of thirteen blocks and bring out two outputs. See**Table 72-3** and **Table 72-4**. Bits [4:5] are for Anatest Control Mux2 -- Select one out of four output connections. See [MUX2 table cross reference in ANATEST]
Bit 31-30		Reserved
RXAFE[4:0] Bits 36-32 Reset Value: 0	ANAtest	Receiver Analog Front End — Input bits.

Table 36-23. Group 12 ANATEST Control[1]

Name	Module	Description
DET_TEST_MODE bit 37 Reset Value: 0	ANAtest	DET_TEST_MODE - Select DET_FLAG_OBS output of RxSDG for observation.
VOC[1:0] Bits 39-38 Reset Value: 0	ANAtest	Voice Codec & Interface — Input bits.
TUNEC[7:0] Bits 47-40 Reset Value: 0	ANAtest	Tuning Circuit — Input bits.
CMON_CAMP[4:0] Bits 52-48 Reset Value: 0	ANAtest	System Clock Monitor — Input bits.
TX_SYN[5] Bit 53 Reset Value: 0	ANAtest	Transmitter and Synthesizer — Input bits.
PAC[4:0] Bits 58-54 Reset Value: 0	ANAtest	Power Amplifier Control DAC/ADC— Input bits.
TX_SYN[4:0] Bits 63-59 Reset Value: 0	ANAtest	Transmitter and Synthesizer — Input bits.
Reg[7:0] bit 71-64 Reset Value: 0	ANAtest	Regulator
GPADC[3:0] bit 75-72 Reset Value: 0	ANAtest	GPADC circuit--Input bits
GRPSEL Bit 79-76		Group selection bits. Must be written to 1011 for selection of Group 12 parameters.

1. All of the bits used by ANATEST are in group 12 except one. See the group10 SPI entry for ACC_REF for more information.

36.5.2 A2DIGL and General Purpose ADC Interface

The general purpose ADC interface of the A2DIGL provides control signals to the GPADC module which is described in “Chapter 35, General Purpose ADC (GPADC)”.

The general purpose ADC interface block provides interface to the general purpose ADC module, the MCU peripheral bus, a test mode convert signal gpio_t_convert, interface to TRSYNT (charge pump output voltages), and output signals to the MCU interrupt controller.

36.5.2.1 Block Diagram

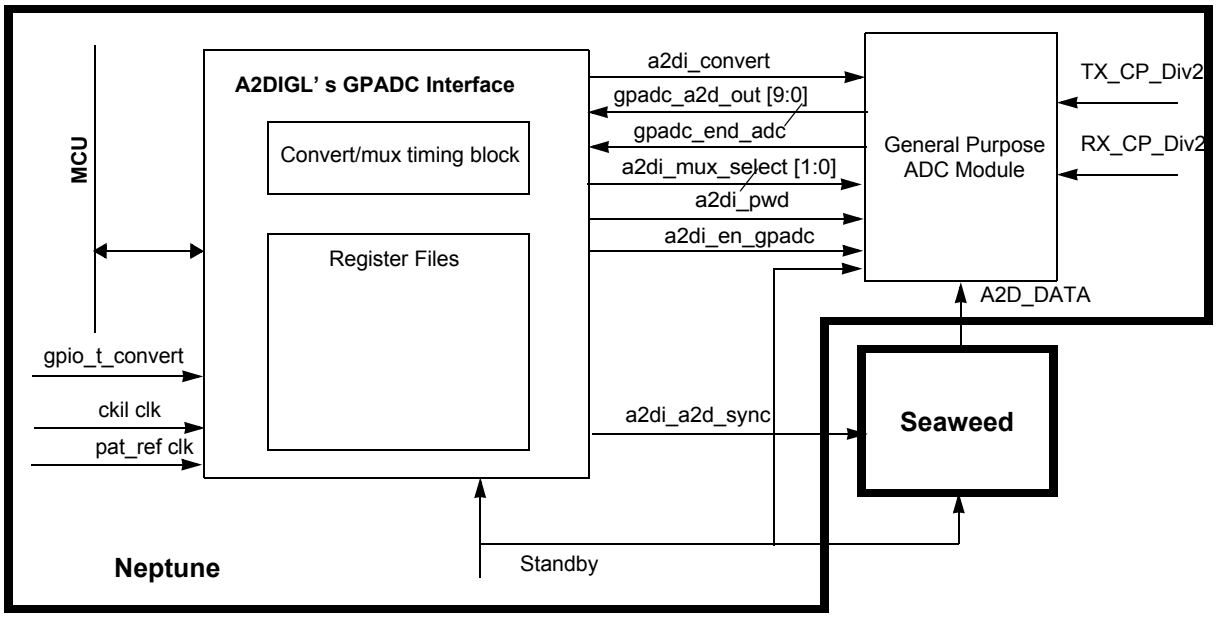


Figure 36-21. General Purpose ADC Interface

36.5.2.2 GPADC Operation

The General Purpose ADC will take samples from the A2D_DATA signal from Seaweed, and the charge pump output voltage signals routed internally in Neptune from the transmit and receive synthesizers (TRSYNT). The GPADC conversion are classified to have three different modes namely, Normal Mode, Deep Sleep Mode and the Test Mode. The conversions will be completed even if the chip enters the low power mode.

36.5.2.2.1 Normal Mode of Operation

Neptune will take 4 samples from the Seaweed A2D_DATA pin, and one each from the Tx charge pump output voltage signal and the Rx charge pump output voltage. For the 4 samples from A2D_DATA, Neptune can program Seaweed to select only a single A/D channel (which will be sampled 4 times) or to cycle through all 4 Seaweed A/D channels. In either case, conversion can be initiated in one of 2 ways:

1. By setting a ‘A/D start conversion’ bit (ASC) in the General Purpose ADC control register, CTRL.
2. By setting the CNVE bit. By this method the conversion process (the GPD timer will start, see below for details) will start on the falling edge of the BSAVETX SPI GROUP 5 bit.

Either scenario will activate the A2DIGL's GPADC delay acquisition timer which has a programmable duration specified by the GPD[15:0] bits in the memory-mapped MTR register. When the timer expires, the acquisition sequence will begin: the 'convert/mux timing logic' block on Neptune will assert a2di_a2d_sync to Seaweed to reset it's muxing, enable the GPADC digital logic by asserting a2di_en_gpadc, enable the GPADC analog section by pulling the a2di_pwd signal low, generate a convert signal (a2di_convert) to the ADC on Neptune, and generate mux select signals (a2di_mux_select[1:0]) for the GPADC. There are a couple of ways in which the a2di_pwd signal is controlled as specified by the APWD and PWD register bits. After each conversion, the A/D value is stored in the data register file[1]. After all 6 conversions are completed, a maskable interrupt can be generated to the MCU.

The following diagrams will explain the Normal Mode operation in more detail

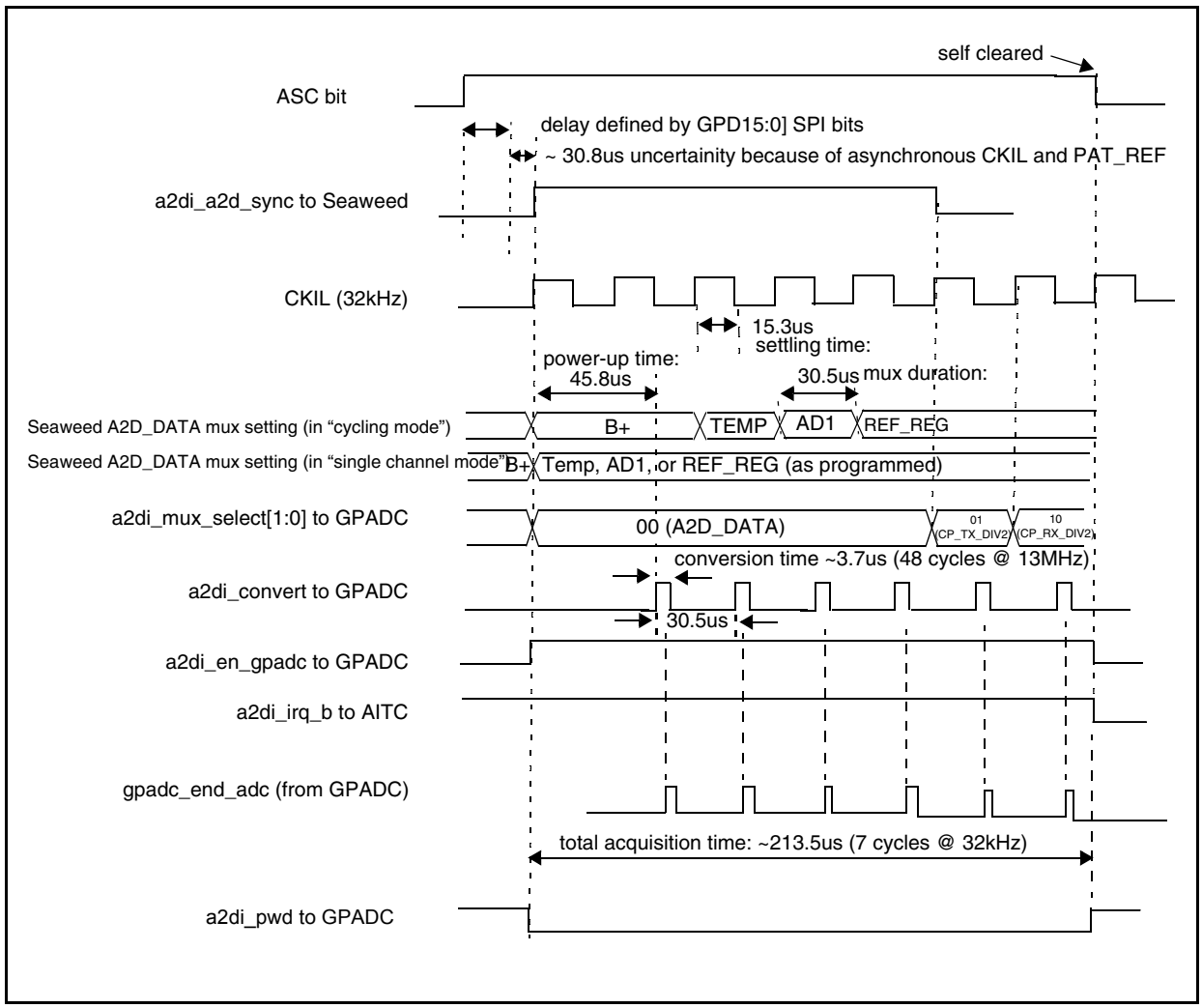


Figure 36-22. Normal Mode Timing using ASC & with APWD = 1, PWD=0/1, DFI=1

In the Figure 36-22 the CKIL clock can have a duty cycle other than 50%. Even in that case the timing requirements shown in the figure would be matched. The logic to assert the a2di_convert is generated with respect to the posiive edge of the first CKIL after the GPD timer expiration. Therefore, the timing betwen the first conversion of 45.8 and then 30.5 us for the remainder would remain constant irrespective of the

1. Refers to registers B+, TEMP, AD1, REF_REG, TXCPDIV2, and RXCPDIV2.

frequency or duty cycle of the CKIL clock. The DFS bit is set when the a2di_en_gpadc goes low after the last conversion finishes (and an interrupt is generated only if the DFI was programmed to 1). Self clearing of ASC/CNVE bits also happens at the same time. The a2di_convert, a2di_en_gpadc and a2di_pwd also goes to the GPIO. Also note that for the ASC conversion the a2di_pwd is essentially the inverted version of the a2di_en_gpadc.

Some more examples of the conversion sequence is detailed in the following figures.

.

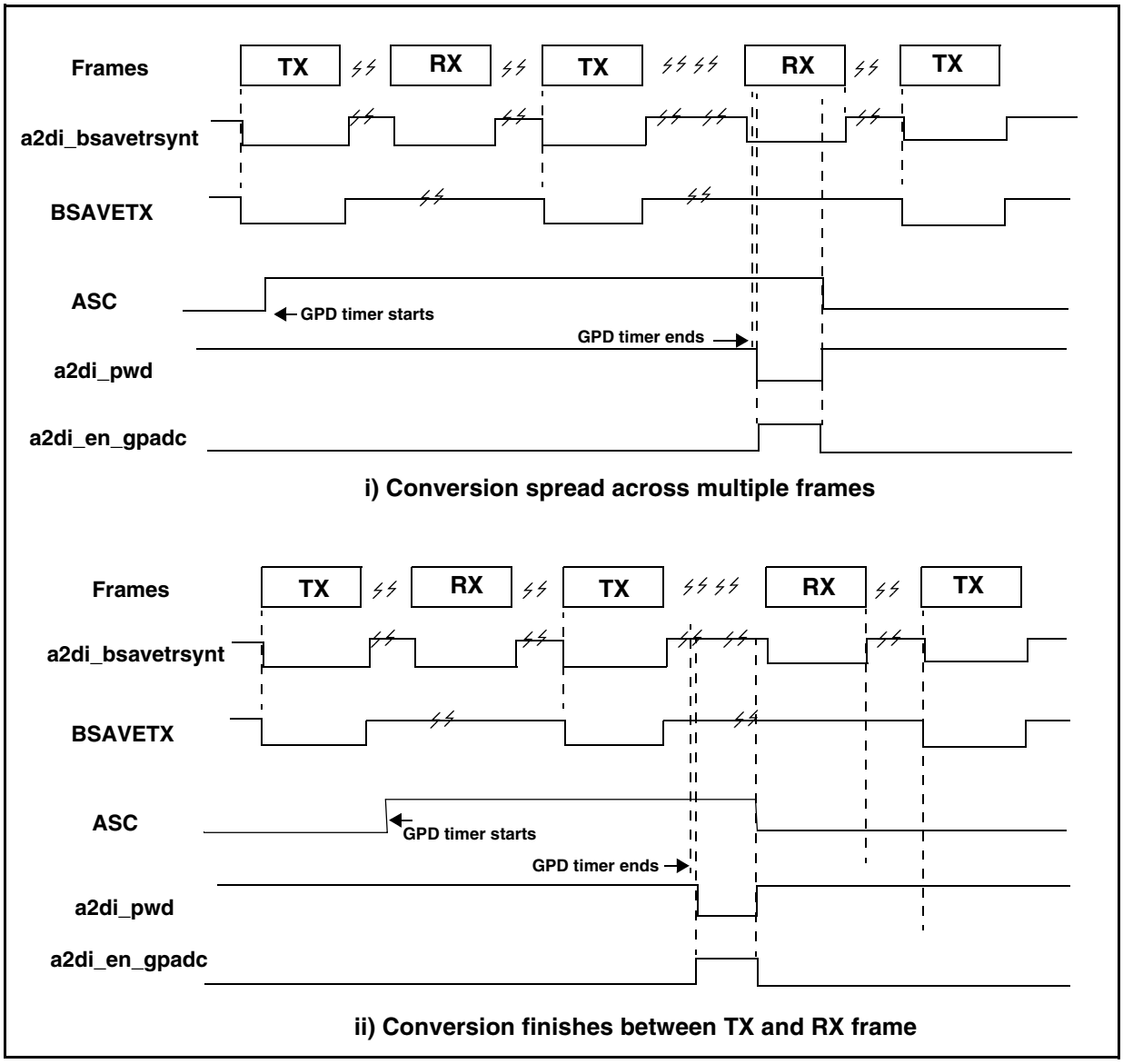


Figure 36-23. More Examples of ASC based conversions with APWD =1, PWD=0/1

NOTE : Using the ASC bit for conversion can potentially cause the a2di_pwd to transition in the middle of the transmit/receive slots. To avoid this use the CNVE bit for conversion sequence.

Next consider the scenario, where the CNVE bit is used to start the conversion process. This conversion is used to start the GPD timer with respect to the falling edge of the BSAVETX bit. If software latency causes the CNVE to get asserted after the BSAVETX goes low, the conversion process (GPD timer) will not start untill the next time when BSAVETX goes low again. This is shown in Figure 36-24 (Duration between conversions/samples are not shown here for simplicity and are the same as in Figure 36-22).

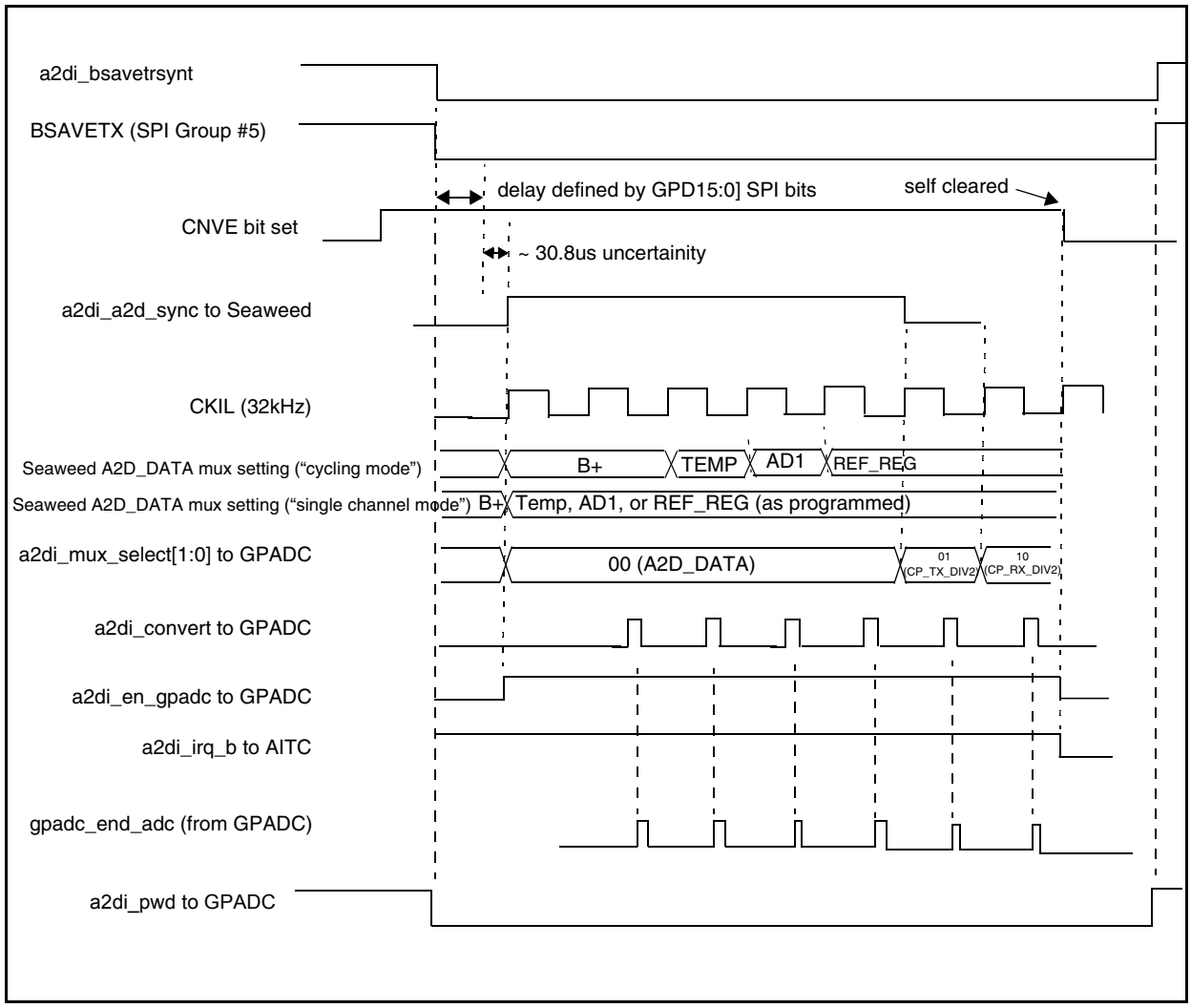


Figure 36-24. Normal Mode Timing using CNVE & with APWD=1, PWD=0/1, DFI=1

As can be seen the a2di_pwd remains low (powering up the GPADC analog sections) for the whole duration of the a2di_bsavetrsynt; well before and after the assertion of the a2di_en_gpadc.

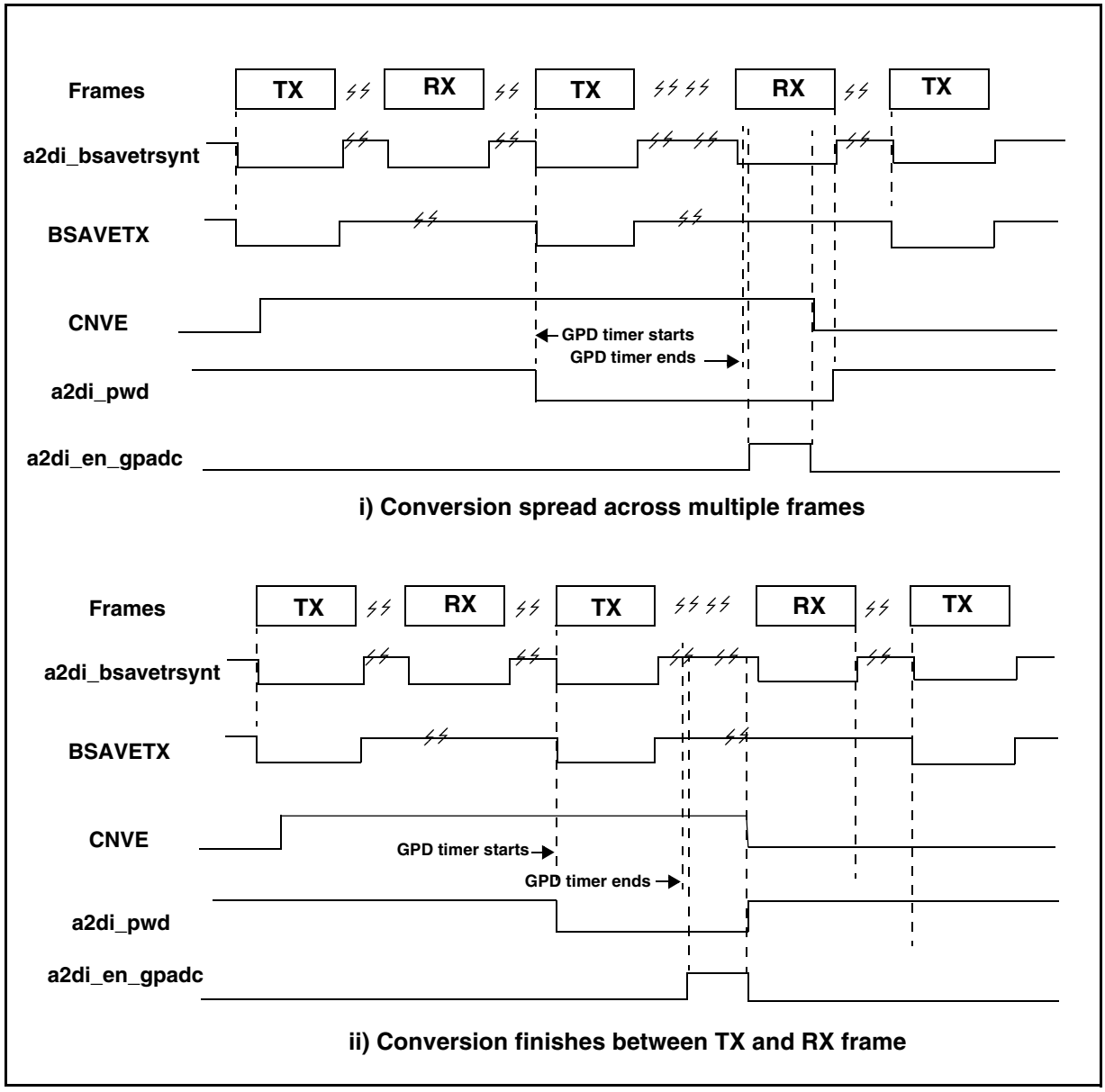


Figure 36-25. More Examples of CNVE based Conversion with APWD = 1, PWD=0/1

As can be seen from the Figure 36-25, the a2di_pwd will extend to the end of the frame (a2di_bsavetrsynt) in the case where the conversion finishes in the middle of the frame otherwise it will go high whenever the a2di_gpadc_en goes low. This behavior of the a2di_pwd is only true for the CNVE based conversion, for the ASC enabled sequence the a2di_pwd is simply the inverted version of a2di_gpadc_en. Both the ASC and the CNVE bits self clear at the end of conversion sequence.

An existing conversion can be terminated by clearing the APWD and setting of the PWD bits. This completely shuts down the A2DIGL - GPADC interface immediately. It also clears the ASC/CNVE bits if they were programmed high. DFS remains unaffected

36.5.2.2.2 Deep Sleep Mode

For deep sleep, Neptune will only sample the A2D_DATA signal (from the Seaweed, charge pump signals are not sampled in deep sleep), and Seaweed will not be required to cycle through all 4 A/D channels. Neptune will need to program Seaweed prior to entering deep sleep mode to select a single A/D channel which will be sampled 6 times. For deep sleep mode, the conversion sequence will be initiated by the rising edge of STANDBY. The delay between conversions will be programmable from 24 to 16384 cycles of the 32kHz clock, as defined by MTR register bits MTR[13:0]. This gives a resolution of 30.5us, a minimum time between conversions of 732us, a maximum time between conversions of 0.5 seconds, and a maximum time of 3.5s over which the seven conversions can be spaced. After each conversion (6 total), the A/D value is stored in the data register file. An interrupt is not generated at the end of the conversions in deep sleep mode; the MCU can read the register file when it awakes from the deep sleep mode. Though there is also a programmable option where each A/D sample can be compared to a high and low threshold, and an interrupt generated if the sample exceeds the thresholds. (see Figure 36-26). The GPADC will be powered down between conversions to save power.

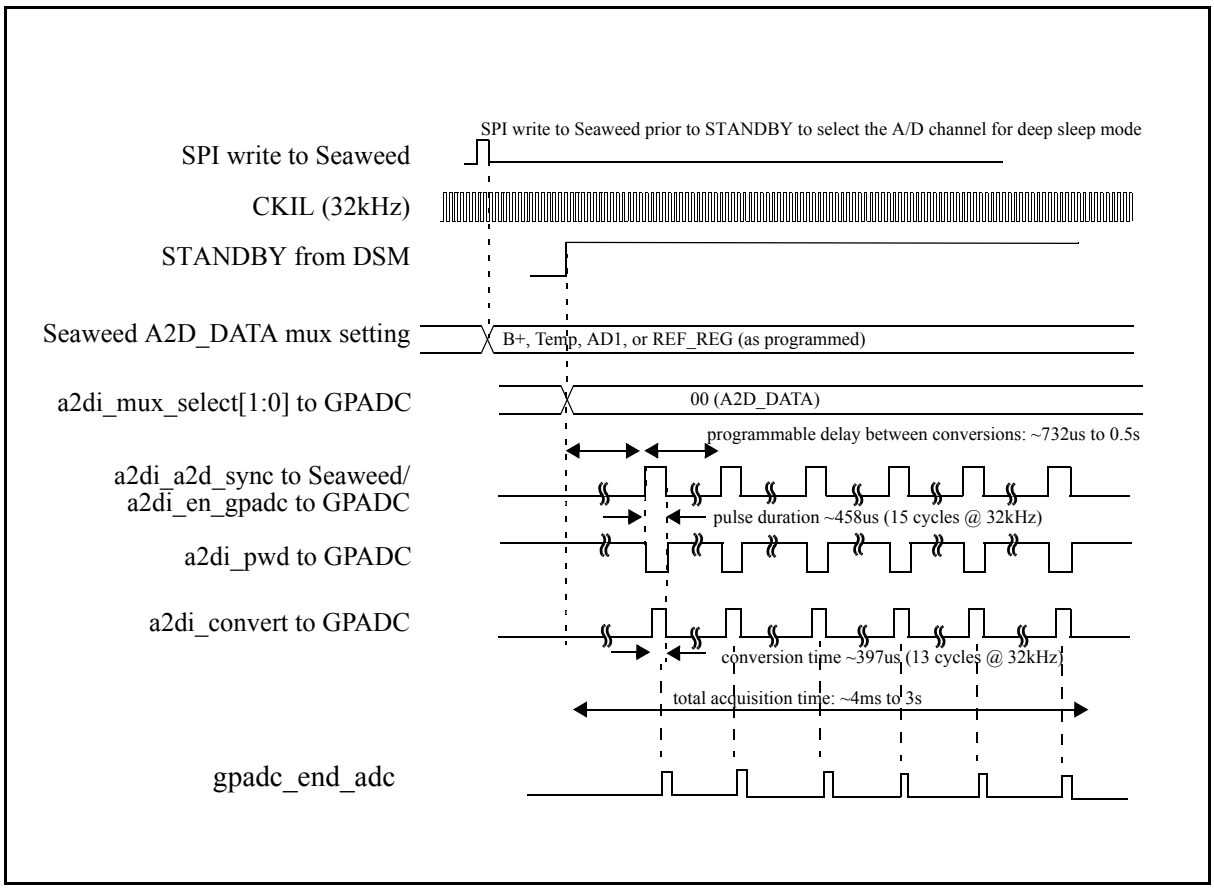


Figure 36-26. Deep Sleep Mode Behavior when APWD = 1, PWD=0/1

Table 36-24. General Purpose ADC Timing

Parameter	Min	Typ	Max
delay from acquisition timer expiration to a2di_a2d_sync high	0		30.8us
delay from a2di_a2d_sync to 1st conversion	45.8us		
total mux/low-pass-filter settling time			15.3us
delay from mux switch to begin convert	15.3us		
mux duration/delay between conversions (for non-standby mode)	30.5us		

36.5.2.2.3 Test Mode

The A2DIGL GPADC control logic also supports a test mode, which is entered when the CTRL register's TST bit is set. In this mode, a single acquisition from the A2D_DATA line will be performed on each rising edge of the gpio_t_convert signal. gpio_t_convert is input to the A2DIGL from an external pin via the GPIO port, and is used as the convert signal going to the GPADC module. The GPADC's data output, as well as it's gpadc_end_adc strobe signal, are routed to the GPIO, and it is intended that this is how the GPADC result will be read by the tester; however, the data value will also be stored by the A2DIGL in the B+ data register in case this value needs to be read by software. The timing for the test mode is shown below in Figure 36-27.

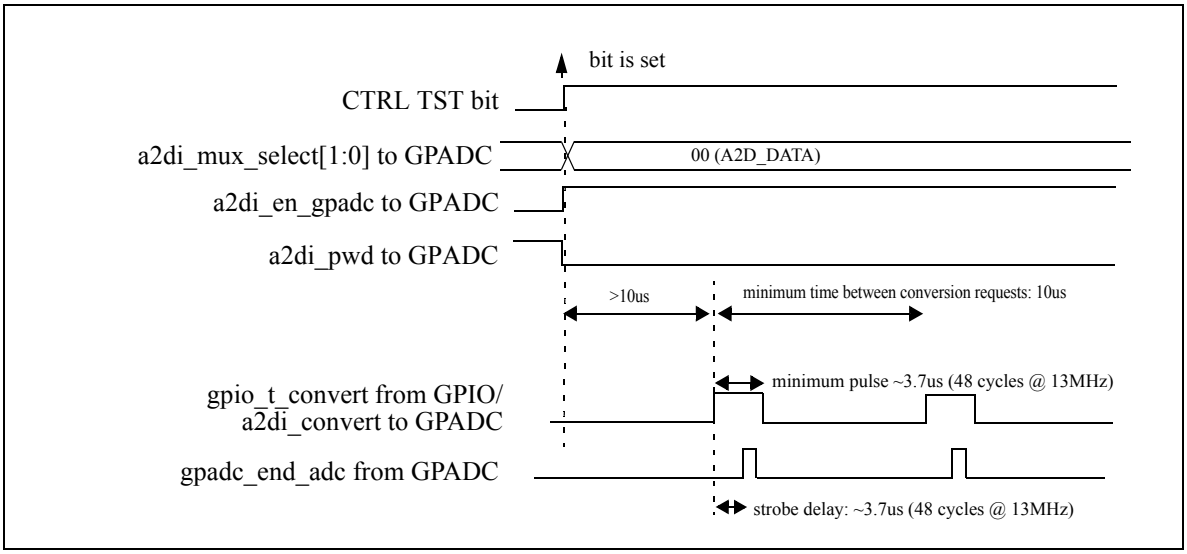


Figure 36-27. Test Mode Behavior, with APWD =1, PWD = 0/1

Note : The a2di_pwd, a2di_en_gpadc remains low and high respectively as long as TST bit is high. This behaviour can also be achieved in the bist mode (irrespective of the programmed value of the TST bit) when the tcm_mbist_en (TCM's tcm_mbist_mode_en) is asserted.

36.5.2.3 RTL Sychronization Issues

The design involves multiple clock domains CKIL, PAT_REF, IPG_CLK. This requires careful design to prevent static timing hazards and metastability issues. Consider the Normal mode operation when the GPD counter is based on PAT_REF clock , the ASC/CNVE bits are based on IPG_CLK and the signals (a2di_en_gpadc, a2di_convert, a2di_a2d_sync, a2di_mux_select) generated based on the GPD counter has to be syncrhonized to the CKIL domain. Instead of using a traditional 2-flop syncrhonizer which could have resulted in additional delay of (30.5 us) for asserting these signals, a different approach is used to generate these a2di_* signals with respect to the first CKIL positive edge. This is shown in the Figure 36-28

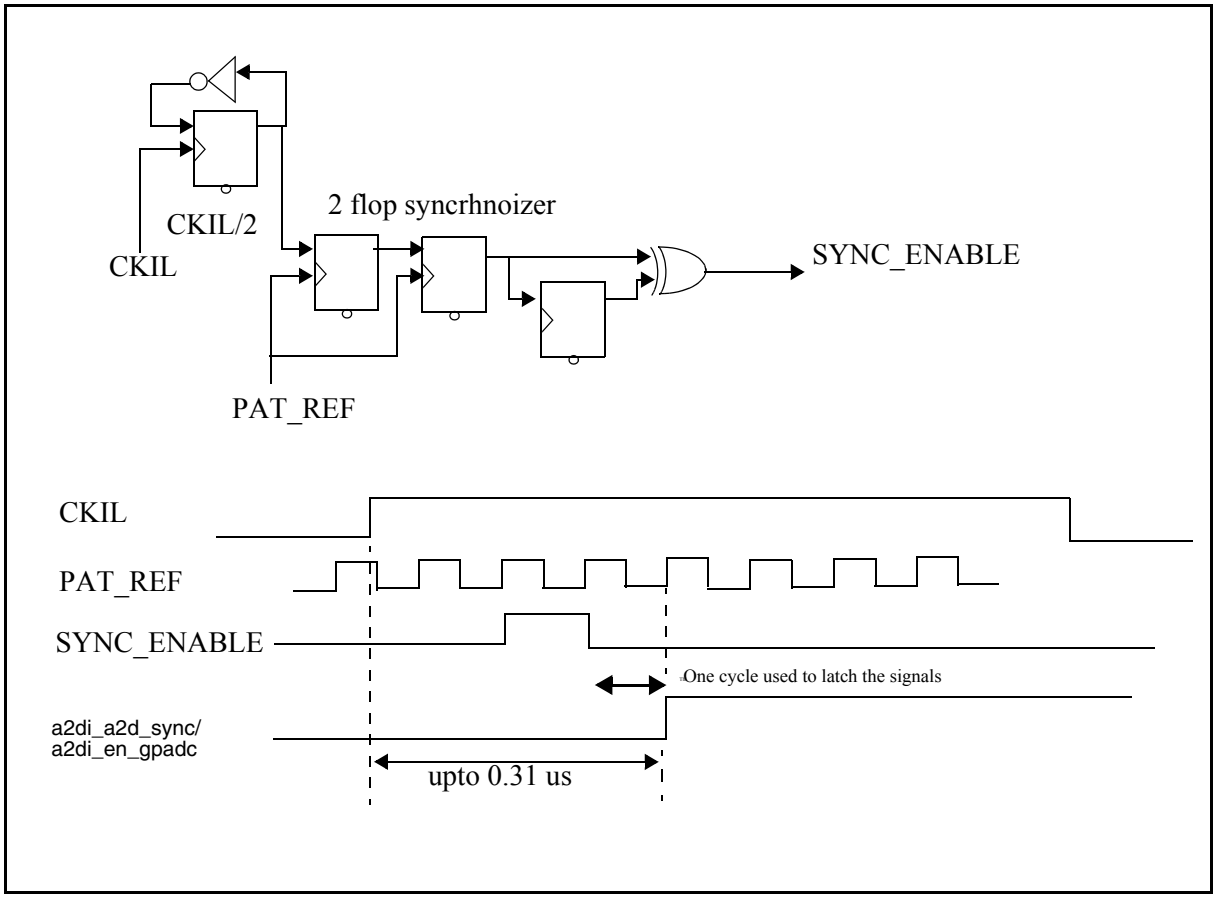


Figure 36-28. Pseudo Synchronization from PAT_REF to CKIL clock domain

In this approach a CKIL positive edge, synhronized SYNC_ENABLE signal is first generated, which is then used to trigger the a2di_* signals. This sync pulse is used also in the design to count the number of CKIL clocks so that after seven clocks the a2di_en_gpadc can be pulled low.

The only disadvantage to this approach is that the a2di_*signals are asserted upto 4 PAT_REF cycles after the CKIL. This should not be a problem as difference in period between the CKIL and PAT_REF is very high. The gpadc_end_adc is also synchronized similarly to generate a pulse that perform threshold comparison in the Deep Sleep mode.

The Normal mode and the Deep sleep mode signals to the GPADC goes out through normal muxes, which could result in glitch outputs.

36.5.2.4 GPADC Interface Register Summary

The GPADC Interface memory-mapped registers are described below:

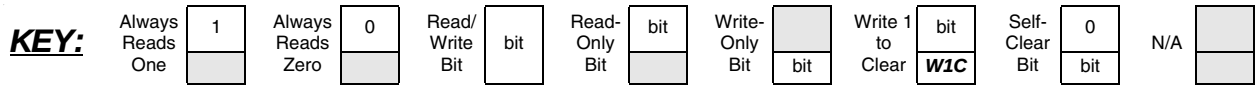


Table 36-25. GPADC Data Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B+ $2484_6010	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	B+[9:0]									
	W																
TEMP $2484_6014	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	TEMP[9:0]									
	W																
AD1 $2484_6018	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	AD1[9:0]									
	W																
REF_REG $2484_601C	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	REF_REG[9:0]									
	W																
TXCPDIV2 $2484_6020	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	TXCPDIV2[9:0]									
	W																
RXCPDIV2 $2484_6024	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	RXCPDIV2[9:0]									
	W																

Table 36-26. GPADC Control/Status Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRL $2484_6028	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0		APWD	CNVE	TST	PWD	ASC	MTR2	DFI	DFS	THHS	THLS
	W														W1C	W1C	W1C
WHIGH $2484_602C	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	WHIGH[9:0]									
	W																
WLOW $2484_6030	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	WLOW[9:0]									
	W																

Table 36-26. GPADC Control/Status Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MTR $2484_6034	R	GPD[15:0]															
	W																
	R	0	0	MTR[13:0]													
	W																

36.5.2.5 GPADC Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the GPADC memory-mapped registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and is cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

B+ B+ Register **Addr $2484_6010**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
							B+[9:0]									
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-27. B+ Description

Name	Description	Settings
B+ Bits 9-0	**B+ —** In the “cycling mode”, this register holds the B+ A/D channel samples. In the other modes, this register holds the samples of the single A/D channel that has been selected.	

TEMP TEMP Register **Addr $2484_6014**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
							TEMP[[9:0]									
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-28. TEMP Description

Name	Description	Settings
TEMP Bits 9-0	**TEMP —**In the “cycling mode”, this register holds the TEMP A/D channel samples. In other modes, this register holds the samples of the single A/D channel that has been selected.	

AD1 AD1 **Addr $2484_6018**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
							AD1[9:0]									
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-29. AD1 Description

Name	Description	Settings
AD1 Bits 9-0	**AD1 —** In the “cycling mode”, this register holds the AD1 A/D channel samples. In other modes, this register holds the samples of the single A/D channel that has been selected. .	

REF_REG REF_REG Register **Addr $2484_601C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
							REF_REG[9:0]									
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-30. REF_REG Description

Name	Description	Settings
REF_REG Bits 9-0	**REF_REG —** In the “cycling mode”, this register holds the REF_REG A/D channel samples. In other modes, this register holds the samples of the single A/D channel that has been selected.	

TXCPDIV2 TXCPDIV2 Register **Addr $2484_6020**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
							TXCPDIV2[9:0]									
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-31. TXCPDIV2 Description

Name	Description	Settings
TXCPDIV2 Bits 9-0	**TXCPDIV2 —** In the “cycling mode” and “non cycling mode” this register holds the TXCPDIV2 A/D channel samples. In deep sleep mode, this register holds the samples of the single A/D channel that has been selected.	

RXCPDIV2 RXCPDIV2 Register **Addr $2484_6024**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
							RXCPDIV2[9:0]									
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-32. RXCPDIV2 Description

Name	Description	Settings
RXCPDIV2 Bits 9-0	**RXCPDIV2 —** In the “cycling mode” and “non cycling mode” this register holds the RXCPDIV2 A/D channel samples. In deep sleep mode, this register holds the samples of the single A/D channel that has been selected.	

CTRL CTRL Register **Addr $2484_6028**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
							APWD	CNVE	TST	PWD	ASC	MTR2	DFI	DFS	THHS	THLS
TYPE	r	r	r	r	r	r	rw	rwm	rw	rw	rwm	rw	rw	w1c	w1c	w1c
RESET	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Table 36-33. CTRL Description

Name	Description	Settings
THLS Bit 0	**Threshold Low Status —** If set, indicates that an A/D sample taken in STANDBY mode was lower than the WLOW threshold. It is the programmer's responsibility to write one to clear this bit. Comparison takes place only when MTR2 is set	
THHS Bit 1	**Threshold High Status —** If set, indicates that an A/D sample taken in STANDBY was higher than the WHIGH threshold. It is the programmer's responsibility to write one to clear this bit. Comparison takes place only when MTR2 is se	
DFS Bit 2	**Data Full Status —** If set, indicates that all A/D samples have been completed. This bit will not be set in STANDBY mode. It is the programmer's responsibility to write one to clear this bit.	
DFI Bit 3	**Data Full Interrupt Enable —** If set, A2DIGL will generate an interrupt when the DFS bit is set	
MTR2 Bit 4	**MTR2 —** If this bit is set, the A/D samples in standby mode are compared to the thresholds WHIGH and WLOW, and an interrupt will be generated if thresholds are exceeded.	
ASC Bit 5	**A/D Start Conversion —** A/D Start Conversion for Seaweed channels. When this bit is set, the A2DIGL GPADC delay acqusition timer will start immediately. This bit will self-clear when the conversion is completed. It will also self clear when the APWD=0 and PWD is changed from 0 to 1 in the middle of conversion.	Setting of TST,ASC and CNVE should be mutually exclusive.

Table 36-33. CTRL Description (Continued)

Name	Description	Settings
PWD Bit 6	**Power Down —** When the APWD bit is set, the programming of PWD bit does not have any effect; the GPADC will be powered on and off automatically as needed. When the APWD bit is cleared, the PWD bit behaves as follows: when PWD is set, the GPADC module will be turned off (a2di_pwd signal will be held high, disabling the analog sections of the GPADC), and the A/D conversion sequence will be disabled (applies to both deep sleep,. normal, and test modes). When PWD is cleared, the analog section will stay enabled (a2di_pwd signal will be held low).	Setting APWD=0 PWD=1 powers down the A2DIGL outputs to GPADC in all the modes of operation Normal, Standby and Test.
TST Bit 7	**TST —** If set, the A2DIGL GPADC will enter a special test mode where T_CONVERT will be used as the convert signal to the GPADC module.	Setting of TST,ASC and CNVE should be mutually exclusive. NOTE : In the BIST mode, equivalent functionality is obtained without the need for setting the TST bit.
CNVE Bit 8	**CNVE —** Convert enable. If set, the A2DIGL GPADC delay acquisition timer (GPD) will start on the next falling edge of BSAVETX Group 5 bit. This bit will self-clear when the conversion is complete. Any falling edge of BSAVETX which occurs before the timer expires or the acquisition sequence completes will be ignored. It will self clear when the APWD=0 and PWD is changed from 0 to 1 in the middle of the conversion.	Setting of TST,ASC and CNVE should be mutually exclusive.
APWD Bit 9	**APWD —** Automatic Power Down. If set, the a2di_pwd signal to the GPADC will be generated automatically by A2DIGL so that the GPADC analog section is enabled only when needed for A/D acquistions (the a2di_pwd signal will be an inverted version of the a2di_en_gpadc signal if ASC controlled conversion is performed). If cleared, the a2di_pwd signal to the GPADC will follow the programming of the PWD bit in this register. The figures in the A2DIGL chapter reflect the timing for the default case where APWD is set.	

WHIGH WHIGH Register **Addr $2484_602C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
							WHIGH[9:0]									
TYPE	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-34. WHIGH Description

Name	Description	Settings
WHIGH Bits 9-0	**WHIGH —** Upper threshold compare.	

WLOW WLOW Register **Addr $2484_6030**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
							WLOW[9:0]									
TYPE	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-35. WLOW Description

Name	Description	Settings
WLOW Bits 9-0	**WLOW —** Low threshold compare. Interrupt enable on register file full.	

MTR MTR Register **Addr $2484_6034**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
	GPD[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
			MTR[13:0]													
TYPE	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-36. MTR Description

Name	Description	Settings
MTR Bits 13-0	**MTR —** Programmable delay between conversions in standby mode. The delay will be MTR+1 cycles of the 32kHz clock. The minimum allowed value for MTR is 23.	If the value programmed is less than 23, A2DIGL will use 23 (24 cycles) internally for the Deep Sleep conversions.
GPD Bits 31-16	**GPD —** Programmable delay prior to conversion in normal mode. The delay will be GPD (to be exact GPD+1) cycles of the PAT_REF clock. For a 13MHz PAT_REF, the maximum delay is ~5ms.	

36.5.3 A2DIGL Test Configuration in BIST Mode

To support the test of the receive, transmit, and GPADC modules while in BIST mode, A2DIGL will automatically perform the following configurations when BIST is enabled:

- To enable testing the receive and transmit modules during BIST:
  - The A2DIGL will act as though the MUXCTL register SSEL bit is set, to allow external SPI signals to drive the A2DIGL SPI inputs
  - The A2DIGL will act as though the MUXCTL register TSEL bit is set, to allow external RX_ACQ and DMCS inputs to A2DIGL
  - The A2DIGL will act as though the MUXCTL register TxSEL2 bit is set, to allow external TX data to be input to A2DIGL (which is then routed on to the TX module)
- To enable testing the GPADC during BIST, the A2DIGL will act as though the CTRL register TST bit is set.

Chapter 37
MCU-DSP Interface (MDI)

Revision History

Revision	Date	Author	Changes
1.0	10/8/99		Initial Release.
1.1	4/10/2000	Janet King	Added notes about DSP clocks gated off during MCU STOP mode, and shared memory clocking requirements during DSP STOP mode.
1.2	9/26/2000	Brian Einloth	Revised for Neptune
1.3	12/4/2000	John VanderMeer	Cleaned up old MCore and PIG references; updated memory size and block size to current design; Fixed minor format issues.
1.4	4th January, 2001	Alok Mittal	Updated the Pin Diagram and the Pin list to reflect the changes required because of DFT methodology and the change of PMBIF to SPMBIF; Fixed minor format issues.
1.5	15th February, 2001	Shannon Osgood	Updated Pin List table format
1.6	27th February, 2001	Rohit Mishra	Added two ports - pivsse5 and mirq5_b to the Pin Diagram and Pin list
1.7	21st March, 2001	Rohit Mishra	Corrected the descriptions for MCR (MRIE1) and DCR (DRIE1).
1.8	8th May,20001	Rohit Mishra	Added clarification for the state of shared memory after MCU resets the MDI and DSP using DHR bit.
1.9	9th May, 2001	Rohit Mishra	Correction of general Typos errors reported in DDTS.
2.0	31st May, 2001	Alok Mittal	Clarification added for generation of mtea during MCU accesses to reserved locations, read-only registers and write-only registers.

Revision History

Revision	Date	Author	Changes
2.1	26th December, 2001	Sachin Borole	Changed the pin list according to Patriot Hip7 requirements and modified some pin names. Changed pin names are res_b to res, bdis0/1 to mcdis0/1, vab to vab_rd, wclk_per_clk to mcu_clk, mirq_ to mirq_b, mnmirq_ to mnmirq_b, p3a_msel_b1 to p3a_msel_b11, wclk_dsp_res_b to wclk_dsp_reset_b, p3a_md_z to md_z, p3a_mtea_b_z to mtea_b_z p3a_mta16_b_z to mta16_b_z rlcont_ to rlcont_b Removed scan related signals (except scan_mode and scan_en) according to DFT methodology for Hip7 and also removed idreq_b and njdreq_b (These pins were used for waking DSP from STOP. But the same is used in DCLKG). Added pins (mcu_clk_b, gclkw_b, mig_scan_tristate_disable_b, spmbif_scan_tri_disable_b0 and spmbif_scan_tri_disable_b1) to the pin list and pin diagram. Updated the schematic block diagram. Added clarification for MCU access to shared memory when DSP goes in STOP (page No. 43). Updated description about bits in MSR (MEP), MCVR (MC) and DSR (MCP). The actual MCU access timings added. The name Neptune changed to Patriot Hip7 and ARM7 replaced by MCORE.
2.2	16th March, 2002	Sachin Borole	Added the idreq_b and njdreq_b to pin diagram and port list. Modified the DSP side Memory mapping diagram (Figure 24-3) and added description about generation of the mtea signal when MCU access to the lower 1K DSP memory since it is not accessible by MDI according to new memory architecture. The MEP bit description clarified more. (added doze and wait mode to the description of stop mode in MEP) Modified the description in the example of the MCU-low power modes section. Added description related to DEP getting stuck to the DSP low power modes section. Modified the foot node in the DSP low power modes section according to the design. Modified the description of the MDI Reset section. (The shared memory gets both the DHR reset and system reset according to the design)

Revision History

Revision	Date	Author	Changes
0.1	19th March, 2002	Sachin Borole	Starting point of this specifications is Patriot Indy MDI. Preliminary specifications released for Neptune LTS. The PIG interface replaced by IP bus and changes done in the pin diagram and pin list accordingly.
0.2	18th April, 2002	Sachin Borole	Changed the names of the ports ips_module_en, scan_en and scan_mode.
0.3	23th May, 2002	Sachin Borole	- Added the frequency ratio restriction in Section 37.7.1.1, “MCU and DSP Frequency Ratio Dependency.” - Added the TRNM bit restriction in Section 37.7.3.5 The cross link is made from TRNM bit in DSR to this description. - Removed the line stating that the output reset to DSP from MDI resets the DSP shared memory from the section MDI Reset. (The DSP memory do not get MDI reset in Indy. Since same memory is used in LTS this line is deleted.) - Added the new MCU access timings (changed due to IP bus interface) for the LTS (see Table 37-20 on page 37-49) - In Section 24-6-2, the point 2. Access type changed according to the IP bus changes. - Added the Section 37.6.2.1, “MDI Timing diagram for shared memory Read and Write.”
0.4	05/07/03		Updated for LTE specification release.

37.1 Introduction

The MCU-DSP interface module (MDI) consists of two independent sub-blocks: a dual-access memory (with read/write access for both processors), and a messaging unit that provides status and messaging control. Following is a list of the main features of the MDI:

- The shared 16-bit memory size is defined in the Chip Configuration chapter.
- Messaging control by interrupts or by polling.
- Flexible support of software controlled messaging protocols.
- The MCU can trigger any interrupt vector in the DSP, as a regular interrupt or as a Non-Maskable one (Command Vector).
- The MCU can wake the DSP from all low-power modes;
  The DSP can wake the MCU from Wait and Doze low power modes.
- The MCU can issue a hardware reset to the DSP.

The basic block diagram of the MDI is given in Figure 37-1.

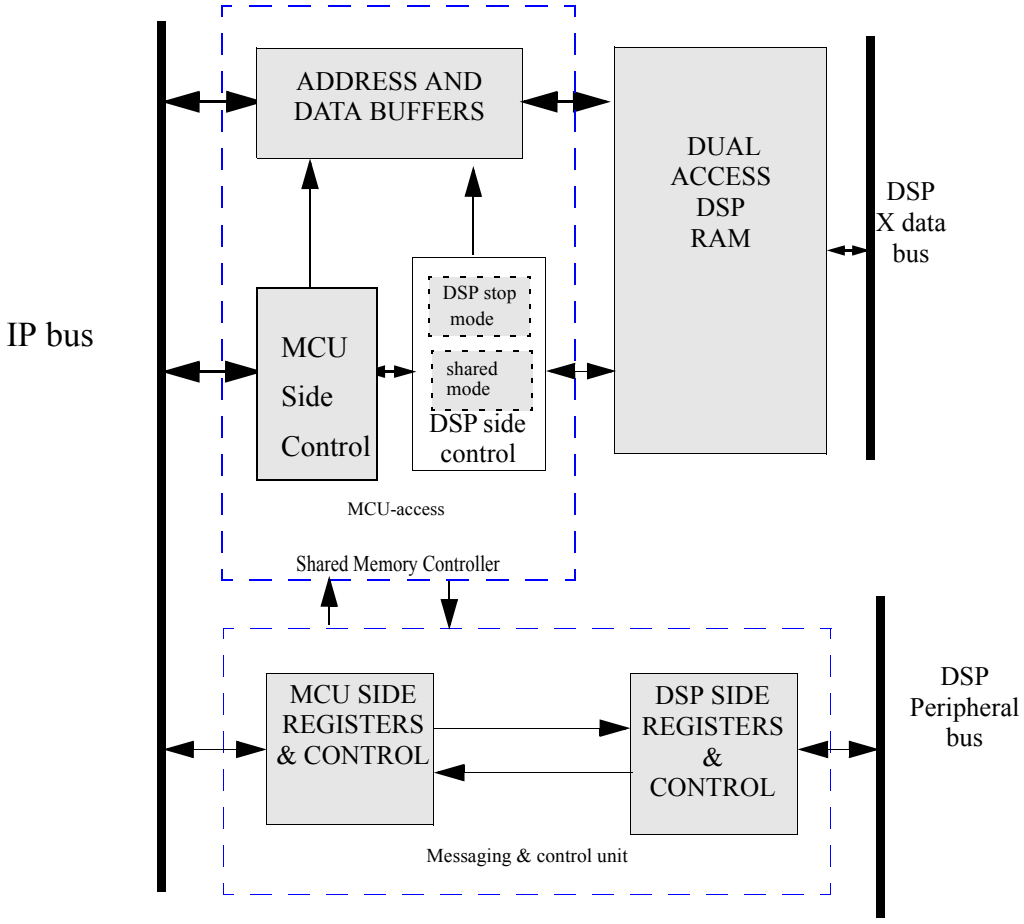


Figure 37-1. MDI Schematic Block Diagram.

The basic pin diagram is given below.

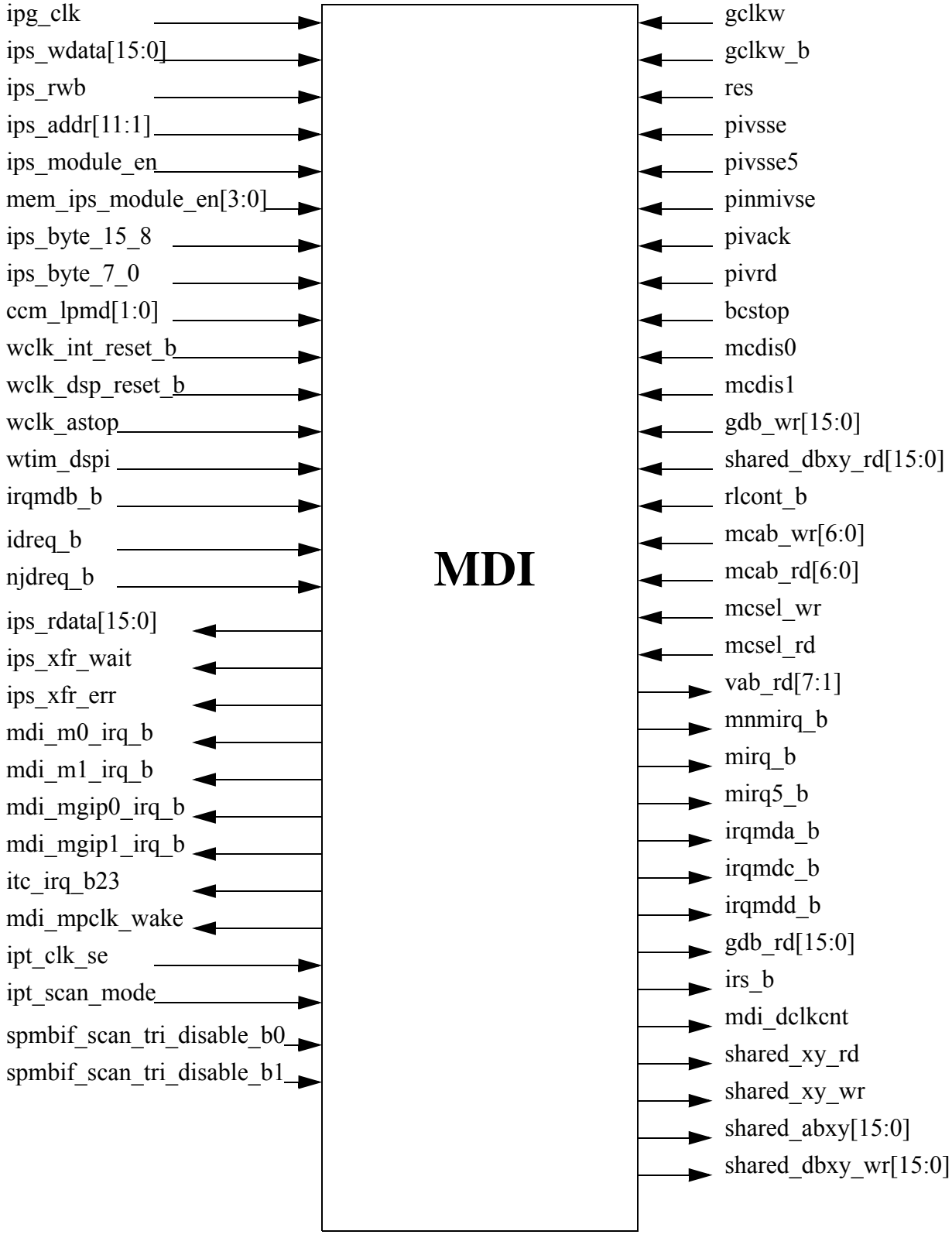


Figure 37-2. Pin Diagram

37.1.1 MDI Module Pin List

Table 37-1 is a list of the MDI module pins.

Table 37-1. MDI Module Pin List

Pin Name	Direction	Description
ipg_clk	in	input clock from clock control
ips_wdata[15:0]	in	16-bit Write data bus from IP bus
ips_rwb	in	Input from IP bus indicates a read transfer when asserted. Indicates a write transfer when negated.
ips_addr[11:1]	in	Address bus from IP bus
ips_module_en	in	Module enable from IP bus indicates MDI is selected and data transfer is in progress
mem_ips_module_en[3:0]	in	Module select for shared memory access by MCU indicates that the MDI is selected and data transfer is in progress. The mem_ips_module_en[0] bit represents lowest bank and mem_ips_module_en[3] -highest bank
ips_byte_15_8	in	Byte select used to select the active [15:8] data bus byte.
ips_byte_7_0	in	Byte select used to select the active [7:0] data bus byte.
ccm_lpmd[1:0]	in	Power mode status of MCU from CCM
wclk_int_reset_b	in	Input reset from clock control
wclk_dsp_reset_b	in	Input reset from clock control
wclk_astop	in	input from clock control
wtim_dspi	in	input from L1 Timer
irqmdb_b	in	input from GPIO
idreq_b	in	DSP Debug Signal from GPIO
njdreq_b	in	input from DSP
ips_rdata[15:0]	out	16-bit Read data bus to IP bus
ips_xfr_wait	out	Wait signal to IP bus. Indicates module does not wish to complete transfer in the current cycle.
ips_xfr_err	out	Error signal to the IP bus when illegal access is attempted.
mdi_m0_irq_b	out	MDI or'd rx0/tx0 irq
mdi_m1_irq_b	out	MDI or'd rx1/tx1 irq

Table 37-1. MDI Module Pin List

Pin Name	Direction	Description
mdi_mgip0_irq_b	out	MDI General0 irq
mdi_mgip1_irq_b	out	MDI General1 irq
itc_irq_b23	out	Or of 4 MDI intrpt sources
mdi_mpclk_wake	out	Output to clock control
ipt_clk_se	in	Scan enable
ipt_scan_mode	in	Scan mode
spmbif_scan_tri_disable_b0	in	Allows tri-states to be disabled from top level during scan in SPMBIF
spmbif_scan_tri_disable_b1	in	Allows tri-states to be disabled from top level during scan in SPMBIF
gclkw	in	Raw clock for New Custom Module
gclkw_b	in	Raw clock for New Custom Module
res	in	Hardware reset
pivsse	in	Interrupt Vector Source Select
pivsse5	in	Interrupt Vector source select for DSP interrupt 5.
pinmivse	in	Non-maskable interrupt vector source select
pivack	in	Peripheral Module Interrupt Acknowledge
pivrd	in	Interrupt vector Read Storbe
bcstop	in	This signal indicates that the core is in stop mode.
mcdis1	in	Core pipeline stall indication
mcdis0	in	Core pipeline stall indication
gdb_wr[15:0]	in	Core Write Data Bus
shared_dbxy_rd[15:0]	in	MDI shared memory Read data bus
rlcont_b	in	input/output from/to MDI shared memory
mcab_wr[6:0]	in	Core Write Address Bus
mcab_rd[6:0]	in	Core Read Address Bus
mcsel_wr	in	Core Write Select
mcsel_rd	in	Core Read Select
vab_rd[7:1]	out	Interrupt Vector Bus
mnmirq_b	out	MCU Non-Maskable interrupt request to DSP

Table 37-1. MDI Module Pin List

Pin Name	Direction	Description
mirq_b	out	Interrupt request to DSP
mirq5_b	out	Interrupt request 5 to DSP
irqmda_b	out	Interrupt request to DSP
irqmdc_b	out	Interrupt request to DSP
irqmdd_b	out	Interrupt request to DSP
gdb_rd[15:0]	out	Core Read Data Bus
irs_b	out	Output reset to DSP
mdi_dclkcnt	out	Output to clock control
shared_xy_rd	out	Read request to MDI shared memory
shared_xy_wr	out	Write request to MDI shared memory
shared_abxy[15:0]	out	MDI shared memory address bus
shared_dbxy_wr[15:0]	out	MDI shared memory Write data bus

37.2 DSP Side Memory Mapping

The shared RAM is mapped to the X data memory space, at the bottom of the internal XRAM starting from the location $0400 to $1FFF. The lower 1K ($0000 to $03FF) is not accessible by MDI and if MDI happen to access this lower 1K ($0000 to $03FF) memory space then the MDI generates module transfer error acknowledge (ips_xfr_err) signal to MCU. From a functional point of view of the DSP, the shared memory is indistinguishable from regular data RAM. A parallel data path allows the shared memory to be written from the MCU, without restricting or stalling the DSP accesses in any way (In case of simultaneous access from both the MCU and the DSP to the same block (7K dsp memory divided into 8 block - so the one block equal to 7K/8), the DSP access has precedence - see Section 37.5, “Simultaneous Access of the DSP and MCU to the Memory.”). The DSP programmer must be aware, however, that data written to that area may be changed by the MCU.

The messaging control and status registers of the MDI will be mapped to the X I/O memory, as a regular peripheral (accessible with the specialized I/O instructions). The memory mapping of the MDI is illustrated in Figure 37-3. The base addresses for the control section and memory section are programmable at production.

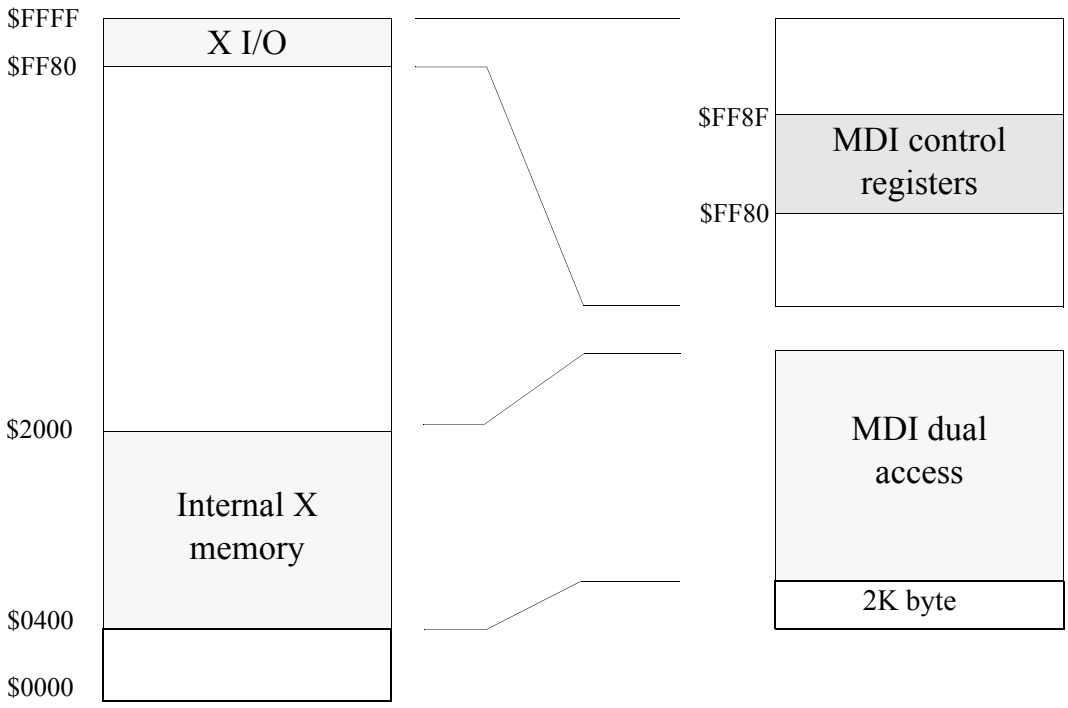


Figure 37-3. MDI Memory Mapping in the DSP Side.

37.2.1 MCU Side Memory Mapping and Access Type Support

The MDI is connected as a peripheral under the IP bus, which buffers the peripherals from the MCU memory bus. A module under the IP bus is allocated 4K addresses (byte wide) in the MCU memory map. The “Messaging And Control Unit” (Figure 37-1) is mapped within one 4K byte bank in the MCU address space and the “MCU-access Shared Memory Controller” (Figure 37-4) is mapped within 4 other 4K byte banks in the MCU address space. Out of this four 4K byte banks of the MCU-access Shared Memory Controller, the lowest 4K bank is not completely accessible for MDI. In this lowest 4K byte bank, only upper 2K byte bank is accessible and if the access happens to the lower 2K byte bank module transfer error acknowledge (ips_xfr_err) signal is given to MCU by MDI. The general MCU memory configuration is given in Figure 37-4. Only data busses are illustrated.

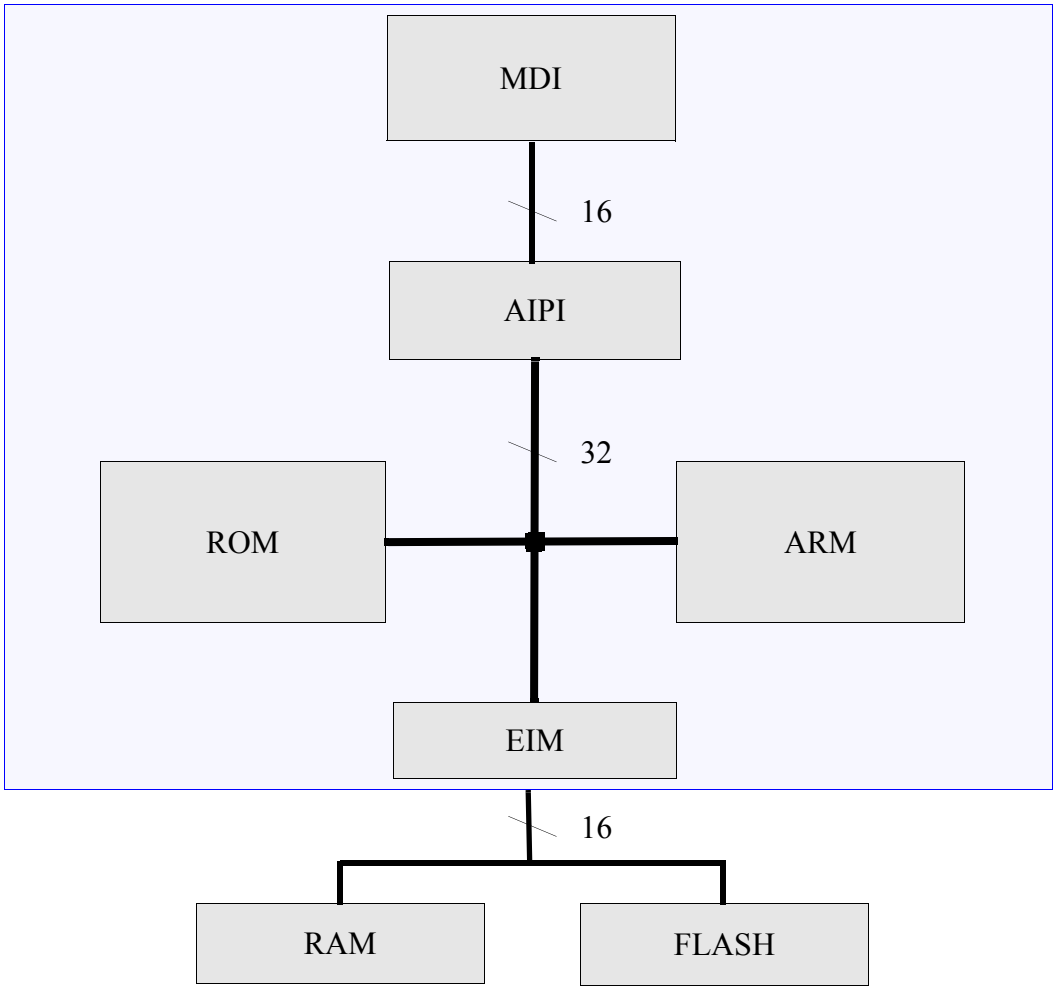


Figure 37-4. MCU Memory Configuration

The IP data bus is 16-bits wide inside the module. In 16-bit accesses, the IP bus will multiplex the appropriate high or low half of the AHB bus to the IP bus. Thus the MDI supports contiguously 16-bit accesses (even aligned). On 32-bit reads, the AIPI bus will drive zeros on the high 16-bits. In 32-bit write, the high 16-bits will be ignored. In every 32-bit access, only one 16-bit register will be accessed in the MDI, therefore only half of the registers will be accessible in aligned 32-bit accesses (see Figure 37-5). 8-bit reads are supported, however in 8-bit writes, there is a difference between the shared memory and the MDI registers. 8-bit writes to the MDI registers at the messaging section are supported. 8-bit writes to the shared memory are not supported, since the shared memory is based on a standard DSP 16-bit memory array module. In 8-bit writes the data on the unused byte is not valid, and the shared memory location may be written with bad data. The programmer should avoid such accesses. On every read, the MDI will drive all 16-bits, so 8-bit reads will produce valid data on the bus. Note that an 8-bit access to a transmit or receive register will have the same effect on status update as a 16-bit access.

The mapping of 16-bit and 32-bit accesses is illustrated in Figure 37-5. To ensure valid operation, the user should access the MDI shared memory only with 16-bit accesses.

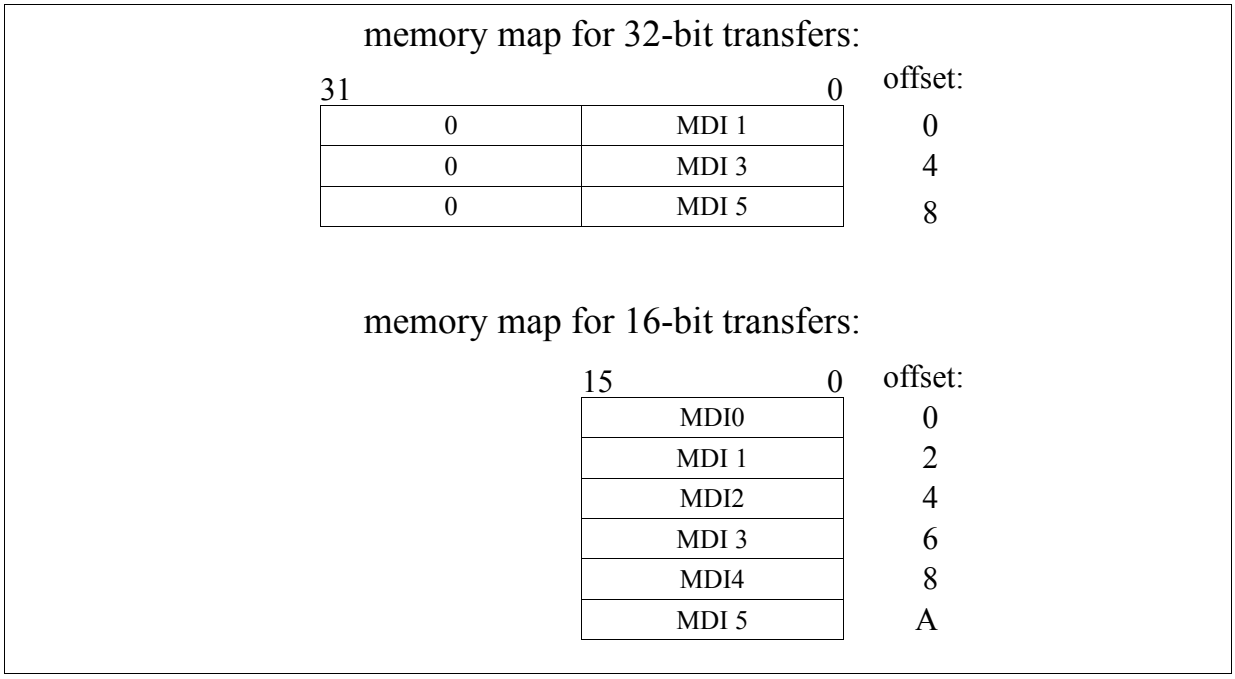


Figure 37-5. MDI Register Mapping in the MCU Address Space.

The MDI access type support is summarized in Table 37-2.

Table 37-2. MDI Access Type Support at MCU Side

Access	Direction	Shared Memory	Control, Status & Messaging Registers
word (32 bits)	reads and writes	unsupported for every second half-word. only one half word accessed in a word access. other half-words are not accessible.	
half word (16-bits)	reads and writes	supported	supported
byte (8-bits)	reads	supported	supported
	writes	not supported	supported

The offset conversion formula between the MDI internal address offset (which is also equal to the DSP offset) and the 16-bit ARM addresses offset is therefore:

$$OFF_{ARM} = OFF_{INT} * 2$$

The MDI control and status registers (MDR) is mapped at the top of the 4K memory space bank allocated to it (from displacement $FF0). The MDI shared memory (MDM) occupies four 4K Bytes memory space banks allocated. The memory mapping of addresses in the MDI is illustrated in Figure 37-6.

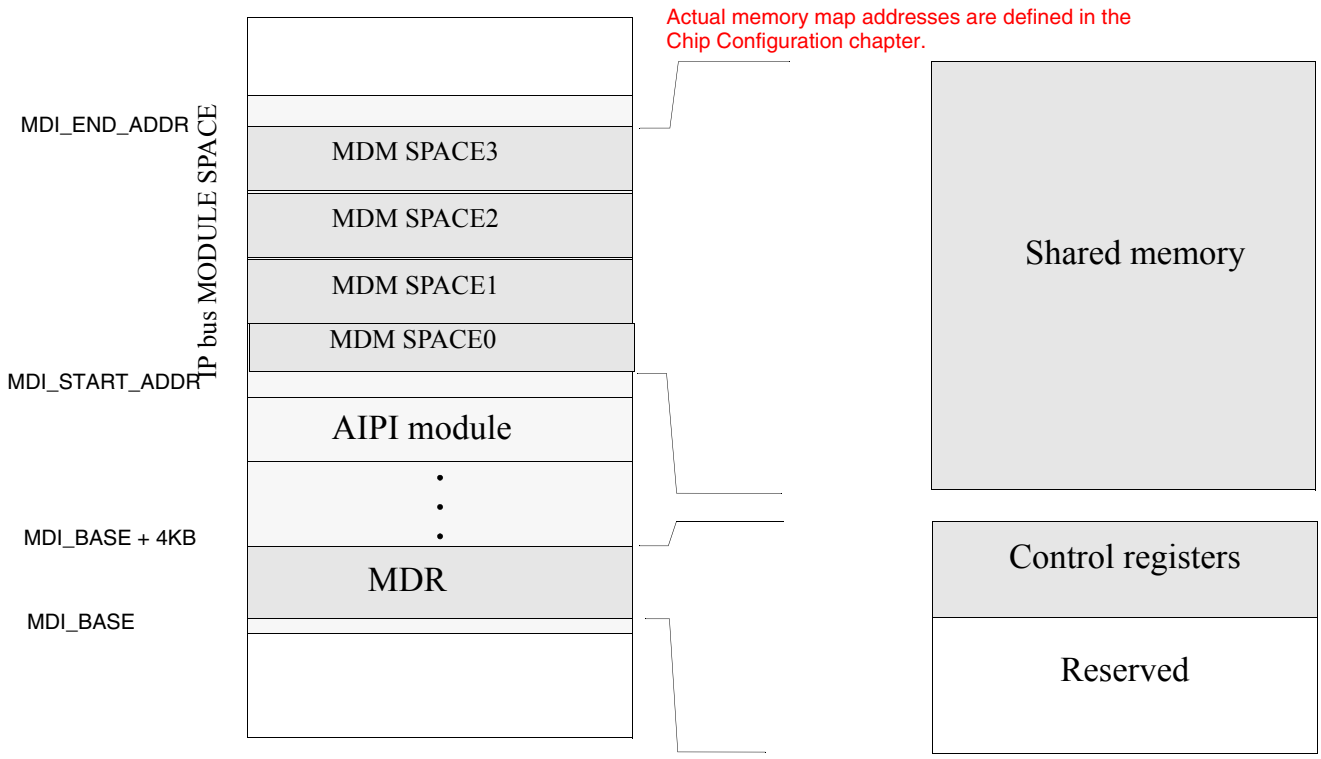


Figure 37-6. MDI Memory Mapping in the MCU Side

A read/write access to any reserved location of MDI would generate the module transfer error acknowledge (ips_xfr_err) to MCU. A write access to read-only registers (MRR0, MRR1) or a read access to write-only registers (MTR0, MTR1) would also generate mtea.

37.2.2 Programmer's Model

The MDI can be viewed as a shared memory array plus a separate messaging block. The size of the memory array is defined in the "Chip Configuration and Memory Maps" chapter. This array is accessible for reads/writes from both DSP and MCU without restriction. Access contentions are resolved in hardware, in which case the DSP access has precedence, and the MCU may stall until the access can be completed.

The messaging logic is independent of the memory array. The user is supplied with various messaging means, which with easy software control can implement a messaging protocol. Some of these messages could mean "I have just written a message of N words, starting at offset X in the memory", or "I have just finished reading the previous data block that was sent". Having the messaging logic independent from the memory array does not restrict the user to a predefined hardware protocol. On the other hand, the software needed to manage the messaging is short and straight forward.

Most of the messaging mechanisms are symmetric, meaning, they are duplicated and available on both DSP side and MCU side. These mechanisms are:

1. Two 16-bit write-only transmit registers, that are each reflected in two read-only receive registers in the other processor's side. These registers could be used to transfer 16-bit word messages, or frame information of messages written to the shared memory (number of words, initial address, message type code, etc.).

2. A write to each of the transmit registers at the transmitter side clears a “transmitter empty” bit in the Status Register at the transmitter side and set a “receiver full” bit in the Status Register at the receiver side. The setting of the bit at the receiver side can optionally trigger an interrupt at the receiver side (maskable receive interrupt).
3. A read of one of the receive registers at the receiver side clears the “receiver full” bit in the Status Register at the receiver side and sets the “transmitter empty” bit in the Status Register at the transmitter. The setting of the “transmitter empty” bit can optionally trigger an interrupt at the transmitter side (maskable transmit interrupt).
4. 3 general purpose flags that are reflected in the Status Register at the receiver side.

Special purpose control functions are also available:

1. Each processor’s power mode is reflected in the other processor’s Status register.
2. The MCU may issue an interrupt to wake the DSP from any of its low-power modes. The DSP may issue an interrupt to wake the MCU from its Wait or Doze low-power modes.
3. The MCU may issue a command interrupt to the DSP. The vector address of this interrupt could be written by software into a register at the MCU side. The command interrupt may be defined as a maskable or non-maskable interrupt.
4. The DSP may issue two general purpose interrupt requests to the MCU, by setting a bit at the DSP side. The interrupt at the MCU side is user maskable.
5. The MCU may issue a hardware reset to the DSP.

The MCU-side signalling and control registers are listed in Table 37-3.

Table 37-3. MCU-Side Signalling & Control Registers

Name	Function	MDI Base Address Offset
MRR0	MCU Receive Register 0	$FFE
MRR1	MCU Receive Register 1	$FFC
MTR0	MCU Transmit Register 0	$FFA
MTR1	MCU Transmit Register 1	$FF8
MSR	MCU-side Status Register	$FF6
MCR	MCU-side Control Register	$FF4
MCVR	MCU-side Command Vector Register	$FF2

The DSP-side signalling and control registers are listed in Table 37-4.

Table 37-4. DSP-Side Signalling & Control Registers

Name	Function	Address
DRR0	DSP Receive Register 0	$FF8F
DRR1	DSP Receive Register 1	$FF8E
DTR0	DSP Transmit Register 0	$FF8D
DTR1	DSP Transmit Register 1	$FF8C

Table 37-4. DSP-Side Signalling & Control Registers

Name	Function	Address
DSR	DSP-side Status Register	$FF8B
DCR	DSP-side Control Register	$FF8A

A schematic illustration of the MDI status and control registers is given in Figure 37-7.

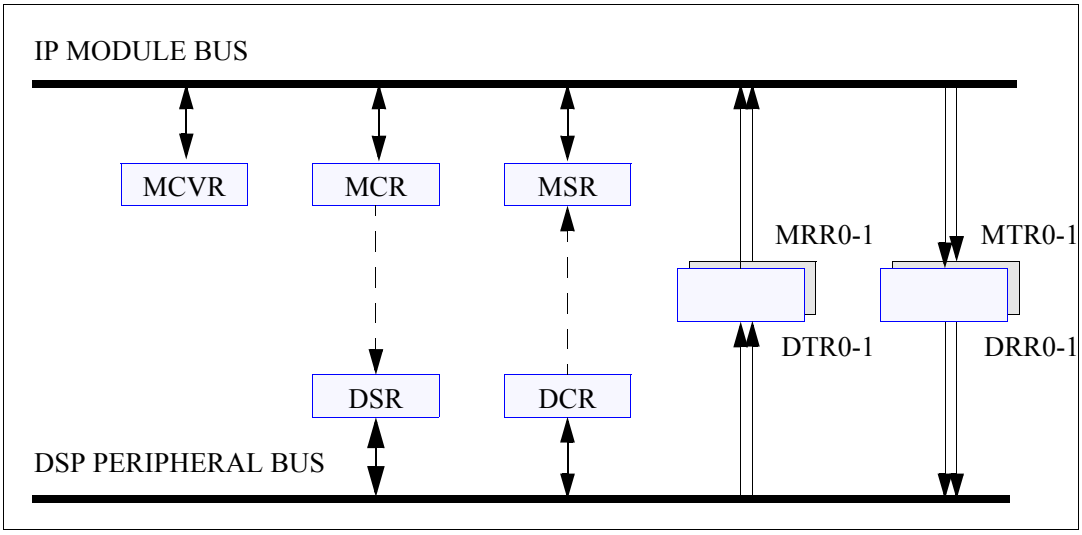


Figure 37-7. MDI Registers.

37.2.3 MDI Register Summary

KEY: Always Reads One (1) | Always Reads Zero (0) | Read/Write Bit (bit) | Read-Only Bit (bit) | Write-Only Bit (bit) | Write 1 to Clear (bit / *W1C*) | Self-Clear Bit (0 / bit) | N/A

Table 37-5. MDI Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCVR ($2485_BFF2)	R	0	0	0	0	0	0	0	MC[1]	MCV[6:0]							MNMI
	W																
MCR ($2485_BFF4)	R	MRIE 0	MRIE 1	MTIE 0	MTIE 1	MGIE0	MGIE1	0	0	DHR	0	0	0	0	MDF[2:0]		
	W										MDIR						
MSR ($2485_BFF6)	R	MRF0	MRF1	MTE0	MTE1	MGIP0	MGIP1	MTIR	DWS[1]	DRS	MSMP	DPM	MEP	0	MF[2:0]		
	W					*W1C*	*W1C*										
MTR1 ($2485_BFF8)	R																
	W	MTR1[15:0]															
MTR0 ($2485_BFFA)	R																
	W	MTR0[15:0]															
MRR1 ($2485_BFFC)	R	MRR1[15:0]															
	W																
MRR0 ($2485_BFFE)	R	MRR0[15:0]															
	W																
DCR (X:$FF8A)	R	DTIE0	DTIE1	DRIE 0	DRIE 1	0	0	0	MCIE	0	0	0	0	0	DMF[2:0]		
	W																
DSR (X:$FF8B)	R	DTE0	DTE1	DRF0	DRF1	DGIR0[1]	DGIR1[1]		MCP		MPM[1:0]		DEP	TRNM	DF[2:0]		
	W							DTIC		DWSC							
DTR1 (X:$FF8C)	R																
	W	DTR1[15:0]															
DTR0 (X:$FF8D)	R																
	W	DTR0[15:0]															
DRR1 (X:$FF8E)	R	DRR1[15:0]															
	W																
DRR0 (X:$FF8F)	R	DRR0[15:0]															
	W																

1. The MC bit in MCVR, the DWS bit in MSR, and the DGIR0 and DGIR1 bits in DSR may be read, or written as 1 to set only (write as 0 is ignored).

37.2.4 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various MDI registers. The following definitions serve as a key for these figures:

- Grey bit: unimplemented bit. Always reads as zero; Writing has no effect.
- TYPE: Type of register bit. Defines register bit's behavior. Possible values:
  - — **r**: read only. Writing this bit has no effect.
  - — **w**: write only.
  - — **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - — **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.

  - — **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - — **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - — **0**: Will reset to a logic 0
  - — **1**: Will reset to a logic 1
  - — **?**: The reset state is unknown.
  - — **u**: Unaffected by reset

The MCU-side registers are described in the following paragraphs.

MTR0 MDI MCU Transmit Register 0 **Addr $2485_BFFA**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	MTR0[15:0]															
TYPE	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-6. MTR0 Description
Table 37-8.

Name	Description	Settings
MTR0 Bits 15-0	**MCU Transmit Register 0--** A 16-bit write-only register. Data written to it is reflected at the DSP side, in DSP Receive register 0 (DRR0). MTR0 and DRR0 are not double buffered - a write to MTR0 overrides the data readable at DRR0. A write to MTR0 also clears the MCU Transmit register 0 Empty bit (MTE0) in the MSR, and sets the DSP Receive register 0 Full bit (DRF0) in the DSR (optionally triggering a receive interrupt at the DSP side). A single 8-bit write to MTR0 will update all status information	N/A

MTR1 MDI MCU Transmit Register 1 **Addr $2485_BFF8**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	MTR1[15:0]															
TYPE	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-7. MTR1 Description
Table 37-9.

Name	Description	Settings
MTR1 Bits 15-0	**MCU Transmit Register 1--** A 16-bit write-only register. Data written to it is reflected at the DSP side, in DSP Receive register 1(DRR1). MTR1 and DRR1 are not double buffered - a write to MTR1 overrides the data readable at DRR1. A write to MTR1 also clears the MCU Transmit register 1Empty bit (MTE1) in the MSR, and sets the DSP Receive register 1Full bit (DRF1) in the DSR (optionally triggering a receive interrupt at the DSP side). A single 8-bit write to MTR1 will update all status information. Analogous functionality to MTR0, completely independent.	N/A

MRR0 MDI MCU Receive Register 0 **Addr $2485_BFFE**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	MRR0[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-8. MRR0 Description
Table 37-10.

Name	Description	Settings
MRR0 Bits 15-0	**MCU Receive Register 0--**A 16-bit read-only register, which reflects the data written to the DSP Transmit Register 0 (DTR0). Reading MRR0 clears the MCU Receive register 0 Full bit (MRF0) in the MSR, and sets the DSP transmit register 0 Empty bit (DTE0) in the DSR (optionally triggering a transmit interrupt at the DSP side). A single 8-bit read from MRR0 will update all status information.	N/A

MRR1 MDI MCU Receive Register 1 **Addr $2485_BFFC**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	MRR1[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-9. MRR1 Description
Table 37-11.

Name	Description	Settings
MRR1 Bits 15-0	**MCU Receive Register 1--**A 16-bit read-only register, which reflects the data written to the DSP Transmit Register 1 (DTR1). Reading MRR1 clears the MCU Receive register 1 Full bit (MRF1) in the MSR, and sets the DSP transmit register 1 Empty bit (DTE1) in the DSR (optionally triggering a transmit interrupt at the DSP side). A single 8-bit read from MRR1 will update all status information. Analogous functionality to MRR0, completely independent.	N/A

The MSR is a 16-bit read-write register used to show information regarding interrupt status from the DSP, general purpose flags, the DSP power mode and dual function control-status bits. Some dual-purpose bits are set by the MDI logic, and cleared by the MCU-side programmer; another is set by the programmer, and cleared by the MDI logic. Refer to Table 37-10, MSR Description.

MSR MDI MCU Status Register **Addr $2485_BFF6**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	MRF0	MRF1	MTE0	MTE1	MGIP 0	MGIP 1	MTIR	DWS	DRS	MSM P	DPM	MEP		MF[2-0]		
TYPE	r	r	r	r	w1c	w1c	r	rw	r	r	r	r	r	r	r	r
RESET	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0

1. The DWS bit in MSR may be read, or written as 1 to set only (write as 0 is ignored).

Table 37-10. MSR Description

Name	Description	Settings
MRF0 Bit 15	**MCU Receive Full 0--** This read-only bit signals the MCU side that new data was written by the DSP to DTR0, and ready to be read by the MCU in MRR0. MRF0 is set when DTR0 was written at the DSP side. It is cleared when MRR0 was read. When this bit is set, a receive 0 interrupt may be issued at the MCU side, if the MRIE0 bit in the MCR is set. MRF0 is cleared at MDI reset.	0 = MRR0 not full (default) 1 = MRR0 full
MRF1 Bit 14	**MCU Receive Full 1—** This read-only bit signals the MCU side that new data was written by the DSP to DTR1, and ready to be read by the MCU in MRR1. MRF1 is set when DTR1 was written at the DSP side. It is cleared when MRR1 was read. When this bit is set, a receive 1 interrupt may be issued at the MCU side, if the MRIE1 bit in the MCR is set. MRF1 is cleared at MDI reset.	0 = MRR1 not full (default) 1 = MRR1 full
MTE0 Bit 13	**MCU Transmit Empty 0 —** This read-only bit, when cleared, signals the MCU side that the DSP did not read DRR0 yet, so MTR0 is not ready to be written at the MCU side. MTE0 is cleared when MTR0 was written at the MCU side. It is set when DRR0 was read at the DSP side. When this bit is set, a transmit 0 interrupt may be issued at the MCU side, if the MTIE0 bit in the MCR is set. MTE0 is set at MDI reset.	0 = MTR0 not empty 1 = MTR0 empty (default)
MTE1 Bit 12	**MCU Transmit Empty 1—** This read-only bit, when cleared, signals the MCU side that the DSP did not read DRR1 yet, so MTR1 is not ready to be written at the MCU side. MTE1 is cleared when MTR1 was written at the MCU side. It is set when DRR1 was read at the DSP side. When this bit is set, a transmit 1 interrupt may be issued at the MCU side, if the MTIE1 bit in the MCR is set. MTE1 is set at MDI reset.	0 = MTR1 not empty 1 = MTR1 empty (default)

Table 37-10. MSR Description (Continued)

Name	Description	Settings
MGIP0 Bit 11	**MCU General Interrupt 0 Pending —** The MGIP0 bit signals the MCU that the DGIR0 bit in the DSR at the DSP side was set from zero to one. If the MGIE0 bit in the MCR is set, a general interrupt 0 request is issued. The MGIP0 bit is cleared by writing it back as one. Writing zero, or writing one when the bit is clear is ignored. This feature should be used in the interrupt routine, where the MGIP0 bit should be cleared in order to de-assert the interrupt request source at the interrupt controller. The proper bit clearing sequence is to clear an MCU register, set the desired bit in it, and write it to the MSR, thus clearing the MGIP0 bit. MGIP0 is cleared at MDI reset.	0 = MCU General interrupt 0 not pending (default). 1 = MCU General interrupt 0 pending
MGIP1 Bit 10	**MCU General Interrupt 1 Pending —** The MGIP1 bit signals the MCU that the DGIR1 bit in the DSR at the DSP side was set from zero to one. If the MGIE1 bit in the MCR is set, a general interrupt 1 request is issued. The MGIP1 bit is cleared by writing it back as one. Writing zero, or writing one when the bit is clear is ignored. This feature should be used in the interrupt routine, where the MGIP1 bit should be cleared in order to de-assert the interrupt request source at the interrupt controller. The proper bit clearing sequence is to clear an MCU register, set the desired bit in it, and write it to the MSR, thus clearing the MGIP1 bit. MGIP1 is cleared at MDI reset.	0 = MCU General interrupt 1 not pending (default). 1 = MCU General interrupt 1pending

Table 37-10. MSR Description (Continued)

Name	Description	Settings
MTIR Bit 9	**MCU Layer 1 Timer wake DSP from STOP and Interrupt Request —** The MTIR bit is a read-only bit that is set when the Layer 1 Timer triggers the DSP interrupt (dsp_wake event), which performs two independent operations: asserts the IRQD input of the DSP, and asserts the IRQA DSP input. If the DSP was in stop and IRQD enabled, assertion of MTIR will wake the DSP and IRQD will be serviced after the DSP woke from stop. If the DSP was not in stop, only IRQD will be serviced. Thus IRQD may be used both as a general purpose interrupt or as an indicator to the DSP that the Layer 1 Timer woke it up out of stop mode. The MTIR bit is cleared by the DSP side by writing “1” to the DTIC bit in the DSR (it is the DSP programmer’s responsibility to explicitly clear this bit in the IRQD service routine). For proper operation of MTIR, IRQD should be defined as a level triggered interrupt in the IPRC in the DSP, and should be in the enabled state when the DSP enters stop. See Section 37.3.2, “DSP Low-Power Modes,” for details. Asserting MTIR a second time before it was cleared will result in the loss of the second assertion. The user should verify that the MTIR bit is cleared before attempting the MDIR operation - see Section 37.4, “MDI Reset.” . The MTIR bit is cleared at MDI reset.	0 = Layer 1 Timer wake DSP from STOP and Interrupt Request not asserted (default) 1 = Layer 1 Timer wake DSP from STOP and Interrupt Request asserted

Table 37-10. MSR Description (Continued)

Name	Description	Settings
DWS Bit 8	**DSP Wake from STOP and Interrupt Request —** Setting the DWS performs two independent operations: asserting the IRQC input of the DSP, and asserting the IRQA DSP input. If the DSP was in stop and IRQC enabled, assertion of DWS will wake the DSP and IRQC will be serviced after it woke. If the DSP was not in stop, only IRQC will be serviced. Thus IRQC may be used both as a general purpose interrupt or as an indicator to the DSP that the MDI woke it up of stop mode. The DWS bit could only be written as "1", and only if it was clear to begin with. Writing it as zero, or trying to re-set it will be ignored. The DWS is cleared by the DSP side by writing "1" in software to the DWSC bit in the DSR (it is the DSP programmer's responsibility to explicitly clear this bit in the IRQC service routine). Therefore the DWS bit can serve as an indicator at the MCU side that IRQC was serviced at the DSP side. For proper operation of this mechanism, IRQC should be defined as a level triggered interrupt in the IPRC in the DSP, and should be in the enabled state when the DSP enters stop. See Section 37.3.2, "DSP Low-Power Modes," for details. The user should verify that the DWS bit is cleared before attempting the MDIR operation - see Section 37.4, "MDI Reset." . The DWS bit is cleared at MDI reset.	0 = DSP Wake from STOP and Interrupt Request not asserted (default) 1 = DSP Wake from STOP and Interrupt Request asserted
DRS Bit 7	**DSP Reset State —** The DRS is a read-only bit that reflects if the DSP-side of the MDI is in reset state or not. If the DRS bit is set, then the DSP-side of the MDI is still in the reset state. This could be because all the DSP is still in reset (after a system reset, or a DSP hardware reset), or if only the DSP-side of the MDI is in reset (after an MDI reset). If the DRS bit is clear, then both the DSP and the DSP-side of the MDI are out of reset. The DRS bit is set at MCU system reset, at DSP hardware reset (caused by setting the DHR bit in the MCR) or at MDI reset (caused by setting the MDIR bit at the MCR). The DRS bit is cleared when the reset sequence at the DSP-side of the MDI ends. After issuing any of the three reset events mentioned, the programmer should verify that the DRS bit is clear before accessing the MDI shared memory.	0 = DSP not in reset state 1 = DSP in reset state

Table 37-10. MSR Description (Continued)

Name	Description	Settings
MSMP Bit 6	**MCU Shared Memory access Pending —** The MSMP is a read only bit that is set if an MCU access to the shared memory is pending. The only way this bit could be read as one is after an MCU write to the shared memory, since a read access is not finished before the access ends, while the write continues to execute after the IP bus access ends. MSMP should be zero before attempting an MDI reset (with the MDIR or DHR bits in the MCR), otherwise the pending write could be lost.	0 = MCU Shared Memory access not pending (default) 1 = MCU Shared Memory access pending
DPM Bit 5	**DSP Power Mode —** The DPM is a read-only bit that reflects the DSP power mode. If the DSP is in Normal mode, the DPM bit is clear. If the DSP is in Stop mode, the DPM bit is set. The transition period from Stop mode or Normal mode when the DSP waits a predefined number of clock cycles to let the clock stabilize is considered part of Stop mode although the clocks are active.	0 = DSP in Normal Mode (default) 1 = DSP in STOP mode
MEP Bit 4	**MCU-side Event Pending —** The MEP is a read-only bit that is set when the MCU-side mechanism sends an event update request to the DSP side, and is cleared when the event update acknowledge was received. An “event” is any hardware message that should be reflected in the DSR at the DSP-side (for example, “transmit register 0 written”). During normal operation the user should not worry about the state of the bit since the event update mechanism works automatically. The user should verify that this bit is cleared only before attempting to enter Stop or wait or doze mode. If MEP is set, the user should wait and continue to poll it before entering Stop or wait or doze. Reading the MSR (to check the MEP bit) should be the last access to the MDI the user should perform before entering Stop or wait or doze, otherwise the MEP may be set as a result of that additional action. Avoiding proper verification of the MEP value may cause some events (including the MCU Stop or wait or doze power mode) not to be reflected at the DSR during the time the MCU is in Stop or wait or doze. The MEP bit is cleared at MDI reset.	0 = MCU-side event not pending (default) 1 = MCU-side event pending
Bit 3	**Reserved bit —** Read as 0, should be written as 0 for future compatibility	N/A
MF[2:0] Bits 2-0	**MCU-side Flag 0-2 —** Each of these bits reflects the values of DMF0-2, respectively.	N/A

The MCVR is a 16-bit read-write register used to control the Command Vector Interrupt, by which the MCU may trigger any interrupt at the DSP side by supplying the interrupt vector offset. This interrupt could be defined as a regular interrupt or as a non-maskable interrupt. The register also reflects whether the interrupt request was serviced at the DSP side. The MCVR could be written only if the MC bit is cleared (meaning there is no pending command interrupt). A write to the MCVR while the MC bit is set is ignored by the hardware. Refer to Table 37-11, MCVR Description.

MCVR MDI MCU Command Vector Register **Addr $2485_BFF2**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
								MC	MCV[6:0]							MNMI
TYPE	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

1. The MC bit in MCVR may be read, or written as 1 to set only (write as 0 is ignored).

Table 37-11. MCVR Description

Name	Description	Settings
Bits 15-9	**Reserved bits —** Read as 0, should be written as 0 for future compatibility	N/A
MC Bit 8	**MCU Command —** Setting the MC bit sets the MCP bit in the DSR at the DSP side. If the MNMI bit in MCVR is set, a non-maskable MCU Command interrupt will be issued at the DSP side. If the MNMI bit is cleared, a maskable interrupt request will be issued at the DSP side if the MCIE bit in the DCR is set. The MC bit will remain set until the Command interrupt is serviced at the DSP side. This way the MCU can monitor if the interrupt was serviced. The MC bit cannot be cleared by software at the MCU side (the bit could only be written to as 1). The MC bit is cleared when the MCP bit in DSR at DSP side is cleared indicating that Command interrupt is serviced. The MCVR could not be written to while the MC bit is set. Before writing to the MCVR, the user should check if the MC bit is cleared. The MC bit is cleared at MDI reset.	0 = MCU Command Vector not asserted (default) 1 = MCU Command Vector asserted
MCV[6-0] Bits 7-1	**MCU Command Vector—** The MCV bits determine the vector address displacement that will be supplied by the DSP during the service of the Command interrupt, thus enabling the MCU to activate any interrupt from the DSP interrupt table.The vector used is the value of MCV6-0 multiplied by 2, so the values encoded by these bits are all the even numbers between 0 and 255 (The DSP interrupts must start at an even address). The MCV bits could only be written to if the MC bit is cleared. At MDI reset, the MCV bits are set to a default value. For Patriot Hip7, this value is $30 to give the interrupt vector at $60.	N/A

Table 37-11. MCVR Description (Continued)

Name	Description	Settings
MNMI Bit 0	**MCU Non-Maskable Interrupt—** If the MNMI bit is set, the Command interrupt triggered by the MC bit will be non-maskable. The MCIE bit in the DCR at the DSP side will be ignored. If MNMI is cleared, the Command interrupt will be enabled at the DSP side only if the MCIE bit at the DCR is set, and also will be maskable by the DSP as a regular interrupt. The MNMI bit could only be written to if the MC bit is cleared. The MNMI bit is cleared at MDI reset.	0 = Command vector interrupt is maskable (default). 1 = Command vector interrupt is non-maskable

The MCR is a 16-bit read-write register used to enable the MDI interrupts at the MCU side, and trigger events at the DSP side (wake from stop, hardware reset, flag update). Refer to Table 37-12, MCR Description.

MCR MDI MCU Control Register **Addr $2485_BFF4**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	MRIE 0	MRIE 1	MTIE 0	MTIE 1	MGIE 0	MGIE 1			DHR	MDIR				MDF[2:0]		
TYPE	rw	rw	rw	rw	rw	rw	r	r	rw	slfclr	r	r	r	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0[1]	0	0	0	0	0	0	0

1. The DHR bit does not return to reset value at software MDIR set.

Table 37-12. MCR Description

Name	Description	Settings
MRIE0 Bit 15	**MCU Receive Interrupt Enable 0 —** The MRIE0 is a read-write bit that enables the MCU receive interrupt 0. If this bit is set, then the receive interrupt 0 request will be issued if the MRF0 bit is set in the MSR. If MRIE0 is clear, the value of the MRF0 bit will be ignored and no receive interrupt request 0 will be issued. The MRIE0 bit is cleared at MDI reset.	0 = MCU Receive Interrupt 0 not enabled (default) 1 = MCU Receive Interrupt 0 enabled
MRIE1 Bit 14	**MCU Receive Interrupt Enable1—** The MRIE1 is a read-write bit that enables the MCU receive interrupt 1. If this bit is set, then the receive interrupt 1 request will be issued if the MRF1 bit is set in the MSR. If MRIE1 is clear, the value of the MRF1 bit will be ignored and no receive interrupt request 1 will be issued. The MRIE1 bit is cleared at MDI reset.	0 = MCU Receive Interrupt 1not enabled (default) 1 = MCU Receive Interrupt 1enabled
MTIE0 Bit 13	**MCU Transmit Interrupt Enable 0 —** The MTIE0 is a read-write bit that enables the MCU transmit interrupt 0. If this bit is set, then the transmit interrupt 0 request will be issued if the MTE0 bit is set in the MSR. If MTIE0 is clear, the value of the MTE0 bit will be ignored and no transmit interrupt request 0 will be issued. The MTIE0 bit is cleared at MDI reset.	0 = MCU Transmit Interrupt 0 not enabled (default) 1 = MCU Transmit Interrupt 0 enabled
MTIE1 Bit 12	**MCU Transmit Interrupt Enable1—** The MTIE1 is a read-write bit that enables the MCU transmit interrupt 1. If this bit is set, then the transmit interrupt 1 request will be issued if the MTE1 bit is set in the MSR. If MTIE1 is clear, the value of the MTE1 bit will be ignored and no transmit interrupt request 1 will be issued. The MTIE1 bit is cleared at MDI reset.	0 = MCU Transmit Interrupt 1not enabled (default) 1 = MCU Transmit Interrupt 1 enabled

Table 37-12. MCR Description (Continued)

Name	Description	Settings
MGIE0 Bit 11	**MCU General Interrupt Enable 0 —** The MGIE0 is a read-write bit that enables the MCU general interrupt 0. If this bit is set, then the general interrupt 0 request will be issued if the MGIP0 bit in the MSR is set. If MGIE0 is clear, the value of the MGIP0 bit will be ignored and no general interrupt request 0 will be issued. The MGIE0 bit is cleared at MDI reset.	0 = MCU General interrupt 0 not enabled (default) 1 = MCU General interrupt 0 enabled
MGIE1 Bit 10	**MCU General Interrupt Enable 1 —** The MGIE1 is a read-write bit that enables the MCU general interrupt 1. If this bit is set, then the general interrupt 1 request will be issued if the MGIP1 bit in the MSR is set. If MGIE1 is clear, the value of the MGIP1 bit will be ignored and no general interrupt request 1 will be issued. The MGIE1 bit is cleared at MDI reset.	0 = MCU General interrupt 1not enabled (default) 1 = MCU General interrupt 1enabled
Bits 9-8	**Reserved bits —** Read as 0, should be written as 0 for future compatibility	N/A
DHR Bit 7	**DSP Hardware Reset —** The DHR is a read-write bit used to assert and de-assert the DSP hardware reset input. Setting the DHR bit issues a hardware reset to the DSP. Clearing the DHR bit de-asserts the DSP hardware reset input. The DHR bit should be kept asserted for a minimum time according to the chip's data-sheet. The delay between asserting and de-asserting the DSR bit should be determined by the programmer in software, according to the current frequency ratio between the processors. Strobe setting the DHR bit will also cause MDI reset, returning all MDI control and status bits to their default values (except the DHR bit itself). After clearing the DHR bit, the programmer should monitor the value of the DRS bit at the MSR to know when the reset sequence at the DSP ended, before attempting an access to the shared memory.	0 = Deassert DSP hardware reset 1 = Assert DSP hardware reset

Table 37-12. MCR Description (Continued)

Name	Description	Settings
MDIR Bit 6	**MDI Reset—**Setting the MDIR bit resets the MDI module, on both DSP and MCU sides, both messaging section and shared memory section. The MDIR bit can only be written as one, and is always read as zero. Setting the MDIR causes all control and status registers to return to their default values (except the DHR bit in the MCR), and all internal states to be cleared. The data in the shared memory array itself remains intact; only the access control logic is affected. After setting the MDIR bit, the programmer should monitor the value of the DRS bit at the MSR to know when the reset sequence at the DSP-side ended. The user should not access the shared memory before verifying that the DRS bit is cleared. Setting the MDIR may affect the ongoing DSP program, therefore the user is advised to interrupt the DSP (with an NMI command) before the MDIR operation. See Section 37.4, “MDI Reset,” for details. The MDIR bit is cleared during the MDI reset sequence.	0 = N/A 1 = Assert MDI reset
Bits 5-3	**Reserved bits —** Read as 0, should be written as 0 for future compatibility	N/A
MDF[2:0] Bits 2-0	**MCU to DSP Flags 0-2—** The MDF0-2 are read-write flags, that are reflected in DF0-2 bits in the DSR at the DSP side. The MDF0-2 bits are cleared at MDI reset.	N/A

DSP-side registers are generally symmetrical to those on the MCU side.

DTR0 MDI DSP Transmit Register 0 **Addr X:$FF8D**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	DTR0[15:0]															
TYPE	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-13. DTR0 Description
Table 37-12.

Name	Description	Settings
DTR0 Bits 15-0	**DSP Transmit Register 0--** A 16-bit write-only register. Data written to it is reflected at the MCU side, in MCU Receive register 0 (MRR0). DTR0 and MRR0 are not dual buffered - a write to DTR0 overrides the data readable at MRR0. A write to DTR0 also clears the DSP Transmit register 0 Empty bit (DTE0) in the DSR, and sets the MCU Receive register 0 Full bit (MRF0) in the MSR (optionally triggering a receive interrupt at the MCU side).	N/A

DTR1 MDI DSP Transmit Register 1 **Addr X:$FF8C**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	DTR1[15:0]															
TYPE	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-14. DTR1 Description
Table 37-13.

Name	Description	Settings
DTR1 Bits 15-0	**DSP Transmit Register 1--** A 16-bit write-only register. Data written to it is reflected at the MCU side, in MCU Receive register 1 (MRR1). DTR1 and MRR1 are not dual buffered - a write to DTR1 overrides the data readable at MRR1. A write to DTR1 also clears the DSP Transmit register 1 Empty bit (DTE1) in the DSR, and sets the MCU Receive register 1 Full bit (MRF1) in the MSR (optionally triggering a receive interrupt at the MCU side). Analogous functionality to DTR0, completely independent.	N/A

DRR0 MDI DSP Receive Register 0 **Addr X:$FF8F**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	DRR0[15-0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-15. DRR0 Description
Table 37-14.

Name	Description	Settings
DRR0 Bits 15-0	**DSP Receive Register 0--**A 16-bit read-only register, which reflects the data written to the MCU Transmit Register 0 (MTR0). Reading DRR0 clears the DSP Receive register 0 Full bit (DRF0) in the DSR, and sets the MCU transmit register 0 Empty bit (MTE0) in the MSR (optionally triggering a transmit interrupt at the MCU side).	N/A

DRR1 MDI DSP Receive Register 1 **Addr X:$FF8E**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	DRR1[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-16. DRR1 Description
Table 37-15.

Name	Description	Settings
DRR1 Bits 15-0	**DSP Receive Register 1--**A 16-bit read-only register, which reflects the data written to the MCU Transmit Register 1 (MTR1). Reading DRR1 clears the DSP Receive register 1 Full bit (DRF1) in the DSR, and sets the MCU transmit register 1 Empty bit (MTE1) in the MSR (optionally triggering a transmit interrupt at the MCU side). Analogous functionality to DRR0, completely independent.	N/A

The DSR is a 16-bit read-write register used to show information regarding interrupt status from the MCU, general purpose flags, the MCU power mode and dual function control-status bits. Dual-purpose bits are set by the DSP-side programmer, and cleared by the MDI logic. For example the general interrupt request bits (DGIR0,1) - are set by writing them as “1”, and cleared by the MDI logic when the interrupt was acknowledged at the MCU side. Refer to Table 37-17, DSR Description.

DSR MDI DSP Status Register **Addr X:$FF8B**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	DTE0	DTE1	DRF0	DRF1	DGIR0 [1]	DGIR1 [1]	DTIC	MCP	DWSC	MPM[1:0]		DEP	TRNM	DF[2:0]		
TYPE	r	r	r	r	rw	rw	w	r	w	r	r	r	r	r	r	r
RESET	1	1	0	0	0	0	0	0	0	?	?	0	0	0	0	0

1. The DGIR0 and DGIR1 bits in DSR may be read, or written as 1 to set only (write as 0 is ignored).

Table 37-17. DSR Description

Name	Description	Settings
DTE0 Bit 15	**DSP Transmit Register 0 Empty--** This read-only bit, when cleared, signals the DSP side that the MCU did not read MRR0 yet, so DTR0 is not ready to be written at the DSP side. DTE0 is cleared when DTR0 was written at the DSP side. It is set when MRR0 was read at the MCU side. Setting this bit may issue a transmit 0 interrupt at the DSP side, if the DTIE0 bit in the DCR is set. DTE0 is set at MDI reset.	0 = DTR0 not empty 1 = DTR0 empty (default)
DTE1 Bit 14	**DSP Transmit Register 1 Empty —**.This read-only bit, when cleared, signals the DSP side that the MCU did not read MRR1 yet, so DTR1 is not ready to be written at the DSP side. DTE1 is cleared when DTR1 was written at the DSP side. It is set when MRR1 was read at the MCU side. Setting this bit may issue a transmit 1 interrupt at the DSP side, if the DTIE1 bit in the DCR is set. DTE1 is set at MDI reset.	0 = DTR1 not empty 1 = DTR1 empty (default)
DRF0 Bit 13	**DSP Receive Register 0 Full —** This read-only bit signals the DSP side that new data was written by the MCU to MTR0, and ready to be read by the DSP in DRR0. DRF0 is set when MTR0 was written at the MCU side. It is cleared when DRR0 was read. Setting this bit may issue a receive 0 interrupt at the DSP side, if the DRIE0 bit in the DCR is set. DRF0 is cleared at MDI reset.	0 = DRR0 not full (default) 1 = DRR0 full

Table 37-17. DSR Description (Continued)

Name	Description	Settings
DRF1 Bit 12	**DSP Receive Register 1 Full —**This read-only bit signals the DSP side that new data was written by the MCU to MTR1, and ready to be read by the DSP in DRR1. DRF1 is set when MTR1 was written at the MCU side. It is cleared when DRR1 was read. Setting this bit may issue a receive 1 interrupt at the DSP side, if the DRIE1 bit in the DCR is set. DRF1 is cleared at MDI reset.	0 = DRR1 not full (default) 1 = DRR1 full
DGIR0 Bit 11	**DSP-side General Interrupt Request 0 —** Writing this bit as one is reflected at the MGIP0 bit in the MSR at the MCU side. If the MGIE0 bit is set in the MCR at the MCU side, an interrupt request will result. The DGIR0 is cleared if the MGIP0 bit at the MCU side is cleared, thus signalling the DSP that the interrupt was accepted (cleared by software). The bit cannot be written as zero at the DSP side. To ensure proper operation, the user must verify that the DGIR0 bit is cleared (meaning that there is no pending interrupt) before setting it. DGIR0 is cleared at MDI reset.	0 = MCU General interrupt 0 not requested (default). 1 = MCU General interrupt 0 requested
DGIR1 Bit 10	**DSP-side General Interrupt Request 1—** Writing this bit as one is reflected at the MGIP1 bit in the MSR at the MCU side. If the MGIE1 bit is set in the MCR at the MCU side, an interrupt request will result. The DGIR1 is cleared if the MGIP1 bit at the MCU side is cleared, thus signalling the DSP that the interrupt was accepted (cleared by software). The bit cannot be written as zero at the DSP side. To ensure proper operation, the user must verify that the DGIR1 bit is cleared (meaning that there is no pending interrupt) before setting it. DGIR1 is cleared at MDI reset.	0 = MCU General interrupt 1 not requested (default). 1 = MCU General interrupt 1requested
DTIC Bit 9	**MCU Layer 1 Timer wake DSP from STOP and Interrupt Clear—** The DTIC is a write-only bit that is to be used by the Layer 1 Timer DSP interrupt (IRQD) service routine in order to clear the interrupt. Writing this bit as “1” signals the MCU side of the MDI to clear the MTIR bit in the MSR, thus de-asserting IRQD and IRQA and enabling the MTIR bit to receive another interrupt. The DTIC bit will always be read as zero, therefore it cannot be polled, and its only use can be as part of the IRQD interrupt service routine, in which it should only be written as “1”, once. The DTIC bit is cleared in MDI reset.	1 = Layer 1 Timer wake DSP from STOP and Interrupt cleared

Table 37-17. DSR Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>MCP
Bit 8</td><td>MCU Command Pending— The MCP is set if the MC bit in the MCVR at the MCU side was set from zero to one. This event may trigger an interrupt (maskable or non-maskable), depending on the state of the MCIE bit in the DCR, and the MNMI at the MCVR at the MCU side. If an interrupt is issued, MCP is cleared when the interrupt is serviced (interrupt acknowledge is received). If the interrupt is masked, MCP remains set, and could not be cleared by software. MCP is cleared at MDI reset.</td><td>0 = MCU Command bit not set (default)
1 = MCU Command bit set</td></tr>
<tr><td>DWSC
Bit 7</td><td>DSP Wake from Stop and Interrupt Clear— The DWSC is a write-only bit that is to be used by the MDI Wake from Stop & general interrupt (IRQC) service routine in order to clear the interrupt. Writing this bit as “1” signals the MCU to clear the DWS bit in the MSR, thus de-asserting IRQC and IRQA and enabling the DWS bit to receive another interrupt. The DWSC bit will always be read as zero, therefore it cannot be polled, and it’s only use can be as part of the IRQC interrupt service routine, in which it should only be written as “1”, once. The DWSC bit is cleared in MDI reset.</td><td>1 = DSP Wake from STOP and Interrupt cleared</td></tr>
<tr><td>MPM[1:0]
Bit 6-5</td><td>MCU Power Mode— MPM1,0 are read-only bits that reflect the MCU power mode, according to the table on the right.</td><td>Table 1: MPM bits
<table><tr><th>MPM1</th><th>MPM0</th><th>MCU power mode</th></tr><tr><td>0</td><td>0</td><td>Stop</td></tr><tr><td>0</td><td>1</td><td>Wait</td></tr><tr><td>1</td><td>0</td><td>Doze</td></tr><tr><td>1</td><td>1</td><td>Normal</td></tr></table></td></tr>
</table>

Table 37-17. DSR Description (Continued)

Name	Description	Settings
DEP Bit 4	**DSP-side Event Pending —** The DEP is a read-only bit that is set when the DSP-side mechanism sends an event update request to the MCU side, and is cleared when the event update acknowledge was received. An “event” is any hardware message that should be reflected in the MSR at the MCU-side (for example, “transmit register 0 written”). During normal operation the user should not worry about the state of the bit since the event update mechanism works automatically. The user should verify that this bit is cleared only before attempting to enter Stop mode, as part of the set of operations needed to be performed before entering stop - see Section 37.3.2, “DSP Low-Power Modes.” . Reading the DSR (to check the DEP bit) should be the last access to the MDI the user should perform before entering Stop, otherwise the DEP may be set as a result of that additional action. Due to DSP pipeline effects, 3 NOP operations (or their timing equivalent) should be given after an instruction that sets an event before the DEP bit will reflect this event (see Section 37.7.3.3, “DSP Pipeline Effects on Registers.”). Avoiding proper verification of the DEP value may result in possible shared memory accesses being lost and in some events possibly not being reflected at the MSR during the time the DSP is in Stop. The DEP bit is cleared at MDI reset.	0 = DSP-side event not pending (default) 1 = DSP-side event pending
TRNM Bit 3	**Transition Mode Bit—** The TRNM is a read-only bit that is set when the DSP core stop mode indication (bcstop) signal is asserted and is cleared after bcstop signal is negated and memory access from MCU is complete. When the DSP has woken from stop mode, the user should verify that this bit is cleared; only then can DSP perform shared memory access. If this bit is set, shared memory access by DSP is not allowed. (see Section 37.7.3.5, “TRNM bit restriction while coming out of stop.” for more about TRNM bit).	0 = Shared memory access allowed (default) 1 = Shared memory access not allowed
DF[2:0] Bits 2-0	**DSP-side Flag 0-2 —** These read-only flags reflect the state of the corresponding MDF0-2 bits, respectively, in the MCR, at the MCU side. The value of DF0-2 after DSP reset follows that of the MDF0-2 bits at the MCU side.	N/A

The DCR is a 16-bit read-write register used to enable the MDI interrupts at the DSP side, and trigger events at the MCU side (general purpose interrupts, flag update). Refer to Table 37-18, DCR Description.

DCR MDI DSP Control Register **Addr X:$FF8A**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	DTIE0	DTIE1	DRIE0	DRIE1				MCIE						DMF[2:0]		
TYPE	rw	rw	rw	rw	r	r	r	rw	r	r	r	r	r	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-18. DCR Description

Name	Description	Settings
DTIE0 Bit 15	**DSP Transmit Interrupt Enable 0—** The DTIE0 is a read-write bit that enables the DSP transmit interrupt 0. If this bit is set, then the transmit interrupt 0 request will be issued if the DTE0 bit is set in the DSR. If DTIE0 is clear, the value of the DTE0 bit will be ignored and no transmit interrupt request 0 will be issued. The DTIE0 bit is cleared at MDI reset.	0 = DSP Transmit Interrupt 0 not enabled (default) 1 = DSP Transmit Interrupt 0 enabled
DTIE1 Bit 14	**DSP Transmit Interrupt Enable1 —** The DTIE1 is a read-write bit that enables the DSP transmit interrupt 1. If this bit is set, then the transmit interrupt 1 request will be issued if the DTE1 bit is set in the DSR. If DTIE1 is clear, the value of the DTE1 bit will be ignored and no transmit interrupt request 1 will be issued. The DTIE1 bit is cleared at MDI reset.	0 = DSP Transmit Interrupt 1not enabled (default) 1 = DSP Transmit Interrupt 1 enabled
DRIE0 Bit 13	**DSP Receive Interrupt Enable 0—** The DRIE0 is a read-write bit that enables the DSP receive interrupt 0. If this bit is set, then the receive interrupt 0 request will be issued if the DRF0 bit is set in the DSR. If DRIE0 is clear, the value of the DRF0 bit will be ignored and no receive interrupt request 0 will be issued. The DRIE0 bit is cleared at MDI reset.	0 = DSP Receive Interrupt 0 not enabled (default) 1 = DSP Receive Interrupt 0 enabled
DRIE1 Bit 12	**DSP Receive Interrupt Enable1—** The DRIE1 is a read-write bit that enables the DSP receive interrupt 1. If this bit is set, then the receive interrupt 1 request will be issued if the DRF1 bit is set in the DSR. If DRIE1 is clear, the value of the DRF1 bit will be ignored and no receive interrupt request 1 will be issued. The DRIE1 bit is cleared at MDI reset.	0 = DSP Receive Interrupt 1not enabled (default) 1 = DSP Receive Interrupt 1enabled
Bits 11-9	**Reserved bits —** Read as 0, should be written as 0 for future compatibility	N/A

Table 37-18. DCR Description (Continued)

Name	Description	Settings
MCIE Bit 8	**MCU Command Interrupt Enable —** The MCIE is a read-write bit that enables the maskable Command Interrupt. If this bit is set, and the MCP bit in the DSR is set, and the MNMI bit in the MCVR is clear, a maskable command interrupt is issued. If the MNMI bit in the MCVR is set, the value of the MCIE bit is ignored. In such a case, if the MCP bit is set, a non-maskable interrupt will be issued. The MCIE bit is cleared at MDI reset.	0 = MCU Command interrupt not enabled (default) 1 = MCU Command interrupt enabled
Bits 7-3	**Reserved bits —** Read as 0, should be written as 0 for future compatibility	N/A
DMF[2:0] Bits 2-0	**DSP to MCU Flags 0-2—** The DMF0-2 are read-write flags, that are reflected in MF0-2 in the MSR at the MCU side. DMF0-2 are cleared at MDI reset.	N/A

37.2.5 Interrupt Sources and Priorities

At the DSP side, the MDI has five independent sets of interrupt interface lines:

1. Command vector interrupt,
2. Receive/transmit interrupt.
3. MDI DSP Wake From Stop/general purpose interrupt (connecting to IRQC DSP input).
4. L1 Timer DSP Wake From Stop/general purpose interrupt (connecting to IRQD DSP input).
5. External DSP Wake From Stop/general purpose interrupt (connecting to IRQB DSP input).

From the DSP point of view, these are five independent interrupt sources, that could be given different priorities at the Interrupt Priority Registers (IPRP/IPRC). The internal MDI priority between the receive/transmit interrupts (if more that one is pending) is fixed as follows:

1. Receive register 0 full
2. Receive register 1 full
3. Transmit register 0 empty
4. Transmit register 1 empty

The priority of the wake from stop interrupts is set solely in the IPRC and the DSP core hardware definition (see Chapter 6, “Chip Configuration and Memory Maps (MEMMAP).”). For proper operation of the IRQC and IRQD interrupts, they should be defined as level triggered and be in the enabled state when the DSP enters Stop mode.

IRQB is controlled directly by the GPIO module. For proper operation of its parallel wake from stop functionality, it should be defined as level triggered, and activated using a software protocol between the DSP and the external source, signalling the external device when to de-assert the interrupt.

At the MCU side, there are five interrupt request lines to the MCU interrupt controller. The first interrupt request line combines all interrupt sources into one request. Determining the interrupt source must be done by the interrupt routine reading the MSR and servicing the highest priority pending (and enabled) interrupt. Priority is completely determined by the interrupt service routine software, not by hardware.

Note that at the MCU side, the General Purpose Interrupt Pending bits (MGIP0,1) should be cleared by software as part of the interrupt service routine in order to de-assert the request to the interrupt controller.

When the MCU is in either the Wait or Doze low-power modes only three out of five interrupts are generated. These are, MCU general purpose interrupt0,MCU general purpose interrupt1 and MDI OR'd Interrupt.Triggering any of these interrupt will wake the MCU before servicing the interrupt.

The other four interrupt request lines are:

1. MCU general purpose interrupt0.
2. MCU general purpose interrupt1.
3. combined interrupt request of MCU receiver full0 and MCU transmitter empty0.
4. combined interrupt request of MCU receiver full1 and MCU transmitter empty1.

Note that when the MCU is in the low-power stop mode these interrupt sources can't be triggered (since the DSP clocks are gated off), and thus they can't take the MCU out of the low power mode.

37.2.6 Implementing Messaging Protocols

The programmer's model of the messaging protocol using transmit and receive registers is illustrated in Figure 37-16.

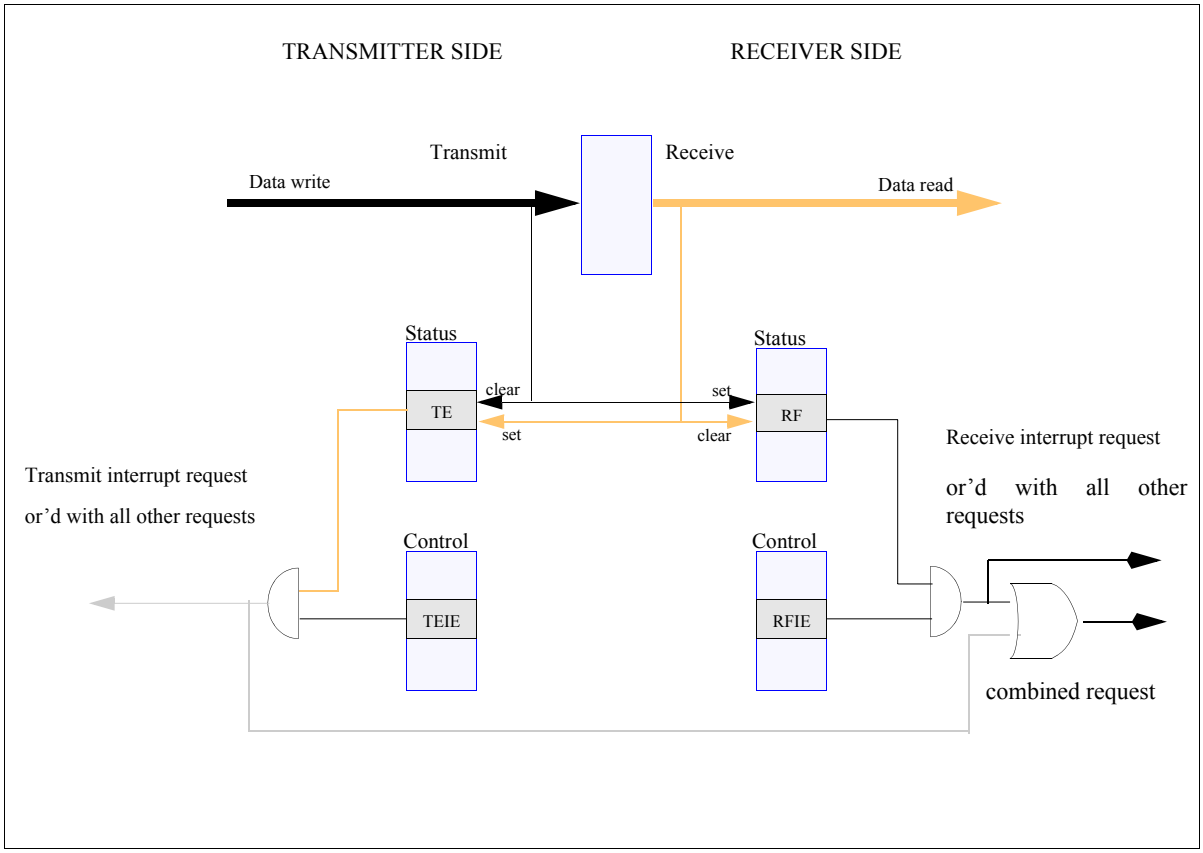


Figure 37-16. Messaging Model Using Transmit and Receive Registers.

The programmer's model of the messaging protocol using the general purpose interrupts (DSP triggers MCU interrupt) is illustrated in Figure 37-17. The same principle applies for the Command Interrupt (MCU triggers a DSP interrupt), with the following differences:

1. The interrupt pending bit in the DSP Status Register is cleared automatically when the interrupt is acknowledged.
2. The trigger bit at the MCU side is in the MCU Command Vector Register (MCVR).
3. In case of a non-maskable interrupt, the interrupt enable bit at the DSP side in ignored.

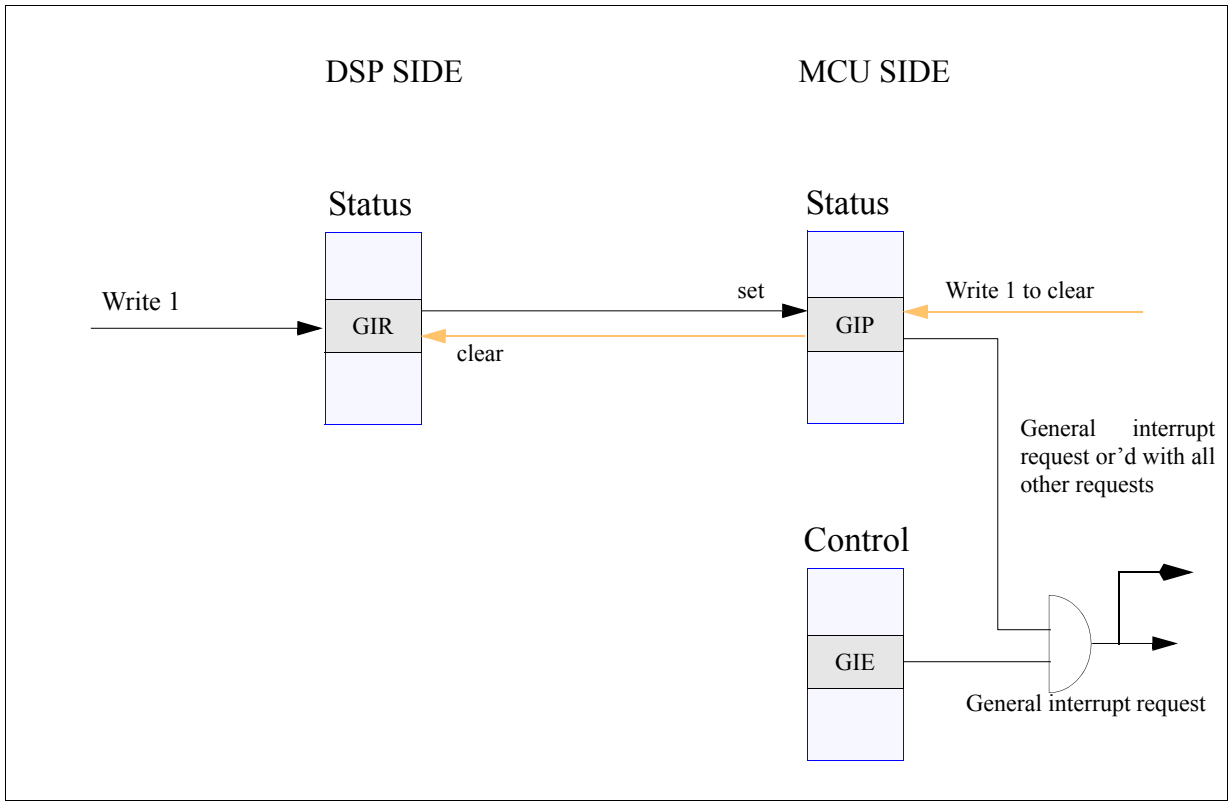


Figure 37-17. Messaging Model Using a General Purpose Interrupt.

The messaging hardware listed above can be used by software to implement messaging protocols for a wide array of message types. Full support is given for both interrupt and polling management. Possible protocol examples:

- Messages up to 16-bits long could be written directly to one of the Transmit Registers.
- Both Transmit registers could be used to pass a 2-word message. Only one of the registers would have it's corresponding interrupt enable bit set at the receiver side. A message first half will be written to the register who's interrupt is masked, and the second half to the other register, triggering an interrupt at the receiver side.
- Transmit registers could be used to pass frame information on long messages written to the shared memory. Such frame information would normally include an initial address and number of words, and perhaps a message type code.
- Events and requests that do not include data words could be signaled from the DSP to the MCU using the two general interrupts, for example: acknowledging that a long message was read from the shared memory. The MCU may interrupt the DSP using the Command interrupt.

- Formatted data with fixed length could be written in predetermined locations in the shared memory. A processor can use a general purpose interrupt (DSP) or command interrupt (MCU) to signal the other processor that the data is ready.
- The 3 flags could be used by a processor to announce to the other processor the program state that it is currently in, or other “bill-board” messages.

37.2.7 MCU Freeing the DSP from a Dead-lock Condition

During normal operation, the MCU may conclude that the DSP does not function properly. In such a case, the MCU has 3 measures it can take to identify and correct the problem. The MSR reflects the DSP power mode. If the DSP is stuck in Stop mode, it could be woken out of it by setting the DWS (DSP Wake from Stop) bit.

If the DSP is in normal mode, The MCU can issue an NMI using the command interrupt. The NMI routine could be made to execute a diagnostic routine predefined for such an event. In the case that none of the measures described above helped, the MCU may issue a hardware reset strobe to the DSP.

37.3 Low-Power Mode Behavior, Clocks and Synchronization

The DSP and MCU clocks operate at different frequencies and from different sources. The DSP will generally run faster, but can run as slow as the MCU. The MDI design does not assume any frequency relationship between the MCU and the DSP. The user must be aware, however, that the frequency relation affects the MDI throughput performance. The data buffers and control logic at each side will operate each with it’s corresponding clock (see Figure 37-1). Both processors may enter and exit low-power modes independently. Low-power mode behavior of the MDI is designed to allow each processor to continue working with its MDI side as much as possible, while shutting down unused logic in order to save power. Both sides of the MDI retain their state during their respective Stops - status and control registers do not return to default values when exiting any low power modes.

37.3.1 MCU Low-Power Modes

The MCU has four power modes: Run, Doze, Wait and Stop.

The MCU can be woken out of a Doze and Wait low power modes by any enabled MCU-side MDI interrupt, as reflected in the MSR (MRF0-1, MTE0-1, MGIP0-1 set) and enabled in the MCR. This way the DSP can actively control when to wake the MCU. This logic enables the DSP to operate independently while the MCU is in any power mode except stop. The user should, however, plan the MCU power mode change protocol with care, regarding the following points:

- What interrupts are enabled at the MCU side.
- What events could be triggered at the DSP side.
- Compatibility with the DSP protocol of entering stop.

For example: It should be decided whether the DSP must wait until interrupts it had triggered at the MCU side are serviced before going into stop. If the DSP does not wait, the MCU protocol must make allowance for the case the DSP is in stop, while the DSR is in a state that will wake the MCU (unless the interrupt is disabled, or the DSP woken) and also MCU event will not be updated on the DSP side, causing MEP to be stuck high.

Before entering Stop, the MCU programmer should check that the MEP bit in the MSR is cleared. This check is needed to ensure that all pending updates from the MCU, including the power mode change when Stop is executed, will be updated in the DSR. If the user wishes to verify that there is no shared memory

write pending before entering Stop, the MSMP bit in the MSR should be checked. If a shared memory write from the MCU side is pending when the MCU enters stop, the access may execute after the MCU has woken up.

While the MCU is in stop mode the clock to the DSP is gated off, so the DSP cannot execute and therefore cannot generate any of the conditions to wake up the MCU from Stop. Therefore, the DSP *must* be placed in Stop mode before the MCU is placed in Stop mode.

Refer to Section 37.3, “Low-Power Mode Behavior, Clocks and Synchronization,” in the Chapter 3, “Clocks, Low Power Control and Reset (CLKRST),” for more information.

37.3.2 DSP Low-Power Modes

The DSP has 3 power modes: Run, Wait and Stop. The MDI, like any other DSP peripheral, cannot distinguish between the Run and Wait modes - in both modes the DSP-side of the MDI will be active. Only in Stop mode, the DSP-side of the MDI will shut down.

DSP entrance and exit from stop are sensitive operations and should be done with care, observing the guidelines given below. The main restriction is making sure that the MCU-side logic of the MDI gets a chance to update the power mode transitions of the DSP (entering or exiting stop), before another transition takes place. This requirement is essential for proper clock control of the shared memory. A sufficient condition for such an update is keeping a minimum delay (measured in MCU clocks) between consecutive DSP entrances to Stop mode. Because the frequency relation between the MCU and the DSP may change, this delay is not fixed in terms of DSP clocks. However, waiting for MDI register events to terminate can supply the needed delay. Therefore, before the DSP enters stop, it must send at least one MDI register event, and wait until the DEP bit in the DSR is cleared. This will ensure the needed delay that is also a function of MCU clocks.This event could be a dummy event, for example:

```
            movep           x:<<DCR,x0
            movep           x0,x:<<DCR;       ;dummy event - write back flags
            nop                               ;nops for pipeline delay
            nop
            nop
_wait
      jset              #DEP,x:<<DSR,_wait
            stop0
```

There is no need that the stop instruction should be immediately after the DEP check, but for proper updating of the MSR while the DSP is in stop the DEP check should be the last MDI access before stop. The dummy event could be skipped if it is ensured that other MDI registers events were given in the interval between the last DSP wake from stop and before the next attempt to enter stop (for example, a transmit register write). However, under no circumstance may the DEP check before entering stop be skipped.

The MCU can wake the DSP from Stop in software by one of the following ways:[1]

1. Setting the DWS bit in the MCR.
2. Triggering a Layer 1 Timer DSP interrupt.
3. Issuing a DSP hardware reset by setting the DHR bit in the MCR.

1. In addition, external sources for waking the DSP from stop are: External DSP interrupt, and system reset. (External DSP debug request and JTAG DSP debug command also wake DSP from stop, but this functionality is done by these signals through DCLKG also. So the functionality, waking DSP out of stop of these signals (idreq_b and njdreq_b) through irqmda_b interrupt is removed from MDI).

Clearing IRQC (by writing “1” to DWSC in the DSR) or clearing IRQD (by writing “1” to DTIC in the DSR) in the interrupt service routine is needed in order to de-assert the interrupt (including the parallel IRQA). The DSP could not enter stop again if IRQA is still asserted. The clearing of DWSC/DTIC just as the DSP exited stop may be considered as an MDI register event for the delay needed before the next entry to stop.

The use should plan the DSP power mode change protocol with care. For example: It should be decided whether the MCU must wait until interrupts it had triggered at DSP side are serviced before going into low power mode. If the MCU does not wait, the DSP protocol must make allowance for the case the MCU is in low power mode and the DSP event will not be updated on the MCU side causing the DEP to be stuck high.

37.3.3 The Shared Memory in DSP Stop Mode

The dual access memory array operates using the DSP clock when possible (i.e. when DSP is not in Stop). The reason is that there is no way to stall DSP accesses, so the memory must respond in accordance to the DSP clock. MCU access to the shared memory will be internally synchronized to the DSP clock. Memory access signals from the MCU must be allowed 2 DSP cycles in order to be synchronized to the DSP clock, and 2 MCU cycles to synchronizes the DSP acknowledgment to the MCU clock. If the DSP runs in a relatively low frequency, extra wait states will be added to the MCU access. These wait states are due to synchronization needs and are not related to wait states due to contention on the memory (see Section 37.5, “Simultaneous Access of the DSP and MCU to the Memory.”)

While the DSP is in Stop mode and the MCU is in Run mode, the MCU is the clock source of the shared memory. The memory controller monitors the Stop mode indicator at the DSP side to know if the DSP has exited from Stop mode or has entered Stop mode. If the Stop mode indicator changes during memory access from the MCU (DSP goes in stop), the MDI switches the mode, and the memory access is regenerated in MDI (the control signals such as read/write strobe to memory and the strobe for latching the read data in buffer in MDI are regenerated) and then access is completed. This is done to prevent loss of data during switching. When the DSP has exited from the Stop mode while MCU access going on, the memory access from the MCU is completed first and then mode is switched from stop to shared mode.

NOTE:

If the DSP memories require quadrature clock in Neptune LTE and the MCU wants to read the shared memory while the DSP is in STOP, a clock path in the CCM must be selected that provides it (MDI_SHIFT_CLK) to the shared memory space.

37.4 MDI Reset

Any MDI reset must affect MCU and DSP sides, as each side reflects the states of the other side. Therefore the MDI has one reset source signal, that initiates the reset sequence at both sides. The MDI reset is triggered at any one of the following cases:

1. MCU (system) hardware reset
2. When setting the “MDI reset” (MDIR) bit at the MCR.
3. When setting the “DSP Hardware Reset” (DHR) bit at the MCR.

The DSP side of the MDI will not be reset when issuing the software RESET instruction from the DSP side. From the MDI point of view, this instruction in equivalent to NOP. After issuing any of the above reset events, the MCU programmer could verify that the reset sequence at the DSP side ended, by checking the DRS bit at the MSR. The user should not access the MDI shared memory before the DRS is cleared.

MDI reset affects both the messaging section and the shared memory controller, at both MCU and DSP sides. MDI reset causes all control and status registers to return to their default values (except the DHR bit itself), and all internal states to be cleared.

The DHR bit directly controls the hardware reset input to the DSP. It is asserted and cleared by software only, so the required assertion length should be determined by the programer to last above the data-sheet minimum, based on the frequency ratio between the processors at the time of assertion. If an MDI reset (caused by MDIR or DHR being set) is done while an MCU write to the shared memory is pending in the write buffer, the access may be lost. To ensure the data was written, the user should check that the MSMP bit in the MSR is cleared before triggering the MDI reset. There is no other sure way to verify that a pending write has finished, since contentions between the DSP and the MCU on the shared memory may in principle stall the MCU access for an unknown number of cycles (see Section 37.5, “Simultaneous Access of the DSP and MCU to the Memory.”).

MDIR assertion is a delicate operation because it affects the DSP side registers asynchronously (from the DSP point of view). This may cause unpredictable behavior in case the DSP is concurrently testing an MDI register bit (for example, “jset #DTE0,x:DSR,tx_sbr). The user should verify that the DSP is not engaged in an MDI signalling activity before asserting MDIR. This should include disabling the DSPWAKE interrupt event at the Interrupt Generator Mask (IGM) Register in the Layer 1 Timer, and then verifying that both the DWS and the MTIR bits in the MSR are cleared. Verifying no MDI activity at DSP side could be done for example by issuing an NMI command prior to MDIR assertion. Also, the MDIR bit could only be asserted when the DSP is not in stop mode. An attempt to set MDIR when the DSP is in stop will be ignored. The instruction immediately following assertion of the MDIR bit should not write to MDI registers. Such a write may be overridden by the reset sequence, and the register will remain with the reset value. The user should wait at least one instruction before attempting such a write.

37.5 Simultaneous Access of the DSP and MCU to the Memory

The memory controller is designed to stall the MCU accesses whenever there is contention on the dual access memory. “Contention” is defined as simultaneous access (read or write) of both MCU and DSP to the same block (7K dsp memory is divided into 8 blocks - so one block is equal to 7K/8) of the shared memory. This partition is beneficial from a functional point of view: The DSP works with a faster clock, therefore if the DSP has to wait for the slower MCU to finish an access it will cause a relatively heavier performance penalty.

Simultaneous access to different blocks (7K dsp memory is divided into 8 blocks - so one block is equal to 7K/8) in the shared memory by the two processors will not cause an MCU stall. Also, access to the DSP/MCU control registers could be done in parallel to any access by the other processor without stall.

(Do NOT READ this paragraph in RED bold as it is meant for the Neptune LT. There is change for the Neptune LTS and LTE for supporting the higher frequency ratio and the IP bus interface. The next paragraph is written for the change**) The MCU side has a data buffer that can store a word from a write request. Therefore the MDI can release the MCU write access with no stall even if the memory array is busy with a DSP access. If the buffer is still busy when the MCU issues another access (read or write), then the MDI will stall the MCU.**

MDI will stall the MCU if there is access going on from the MCU to the shared memory and the contention.

In some cases the stall may not even be evident on the MCU side, if it lasted less then one MCU clock. On the other hand, consecutive 1-cycle accesses by the DSP to the MDI memory may stall an MCU access for as long as they last.

The following DSP code is the fastest way for transferring data from the MDI to the internal Y memory, executing an MDI read every cycle:

```
            move            x:(r0)+,a
            move            x:(r0)+,b
            DO #(N/2-1), _BE_NASTY_TO_MCU
            move            x:(r0)+,aa,y:(r4)+;r0 points to MDI memory
            move            x:(r0)+,bb,y:(r4)+;r4 points to other memory
_BE_NASTY_TO_MCU
            move            a,y:(r4)+
            move            b,y:(r4)+
```

This loop will stall the MCU in case the MCU issues a memory read, for as long as the loop lasts. A more considerate mode of work will have the DSP moving data from the shared memory once every 2 cycles, so there will be available time slots for the MCU to conclude accesses in parallel. For example, the following DSP code does the same thing as the previous example:

```
            DO          #N,_IM_OK_MCU_OK
            move            x:(r0)+,x0          ;r0 points to MDI memory
            move            x0,y:(r4)+          ;r4 points to other memory
_IM_OK_MCU_OK
```

The second instruction in the loop allows pending MCU accesses to execute.

37.6 MDI Access and Event Update Timing

The following sections describe MDI access and timing in the DSP. As the frequency relationship between the DSP and MCU core changes, the MDI will continue to function, but the shared memory access time will change.

37.6.1 Access to the Shared Memory from the DSP Side

When the DSP is active, it has priority when accessing the shared memory. Every access to the MDI shared memory or an MDI control register lasts 1 cycle, and is executed as part of the DSP pipeline without stalling it.

37.6.2 Access to the Shared Memory from the MCU Side

The basic timing of an access to a module under the IP bus is one MCU clocks per access. This timing does not include the instruction fetch time, which should be added according to the case. MCU access to MDI control registers will always be 1 cycles long. Shared memory accesses may last longer, according to the following rules:

1. Clock source of the shared memory: If the DSP is in stop mode, the shared memory will operate using the MCU clocks. If the DSP is active, it will generate the memory clocks at full frequency and all MCU accesses should be synchronized to it.
2. Access type: The MCU can do continuous read/write to the shared memory. The access will not be stalled (except for the case of contention). The access will continue until the wait signal is asserted. The wait signal is de-asserted when the synchronized acknowledge of read or write access is received. (*Please DO NOT READ the descriptions in RED bold. It was present in Neptune LT. This has been changed for Neptune LTS and LTEfor supporting higher frequency ratio and IP bus interface. **An MCU write is done to a buffer at the MCU side. If the buffer is empty, the write access will last the 3-cycle minimum.The**

MDI will complete the write to the shared memory later, in a minimum of another 2 MCU cycles. Only then will the buffer be free. In case of a read, or a write when the buffer is not yet free from a previous write, the access will stall.)

3. Relative frequency of the MCU and the DSP clocks: An MCU access generates a request to the DSP side that must be synchronized to the DSP clock (2 DSP clocks in the worst case), and an acknowledge from the DSP to the MCU side, that must be synchronized to the MCU clock (2 MCU clocks in the worst case). The synchronization stall therefore depends on the frequency of both processors. The slower the DSP frequency is, relative to the MCU frequency, the longer the access time (measured in MCU clocks). In a typical system configuration, the DSP's frequency is higher or equal to the MCU's frequency. In this assumption, the maximum MCU stall is if the frequencies of the MCU and the DSP are equal. If the DSP frequency is lower than the MCU frequency, the access time (measured in MCU clocks) may in principle be very long, depending on how slow the DSP is.
4. DSP parallel accesses: Any DSP access in parallel to an MCU access to the same 7K/8 memory block may further stall a pending MCU access. If the DSP does not run consecutive 1-cycle accesses, and the MCU frequency is not faster than the DSP's frequency, than an MCU contention stall will be no more than 1 MCU cycle.

MCU side access timing is summarized in Table 37-19.

Table 37-19. MCU Access Timing to the MDI in INDY (Neptune LT) Do not read for Neptune LTE

Access type	DSP clocks	MCU Cycles (minimum[1])	MCU Cycles (maximum[2])	Comments
Shared memory read	not active	4(5)	4(5)	Assuming the write buffer was empty.
	active	5	9	
Shared memory write	Any case	3	3	Assuming the write buffer was empty.
Buffer busy after shared memory write	not active	+ 0	+ 0	Added MCU stall for consecutive access.
	active	+ 2(1)	+ 4	
MCU-DSP shared memory contention	active	+ 0	+ 1	Added MCU stall. Assuming the DSP does not execute multiple 1-cycle instructions.
Control register access	any case	3	3	

1. Minimum case is if the DSP clock frequency is >> than the MCU frequency.
2. Assuming the DSP is not slower than the MCU, the maximum case is when the DSP clock frequency equals the MCU clock frequency

Table 37-20. MCU Access Timing to the MDI (Number of wait states in terms of MCU clock)

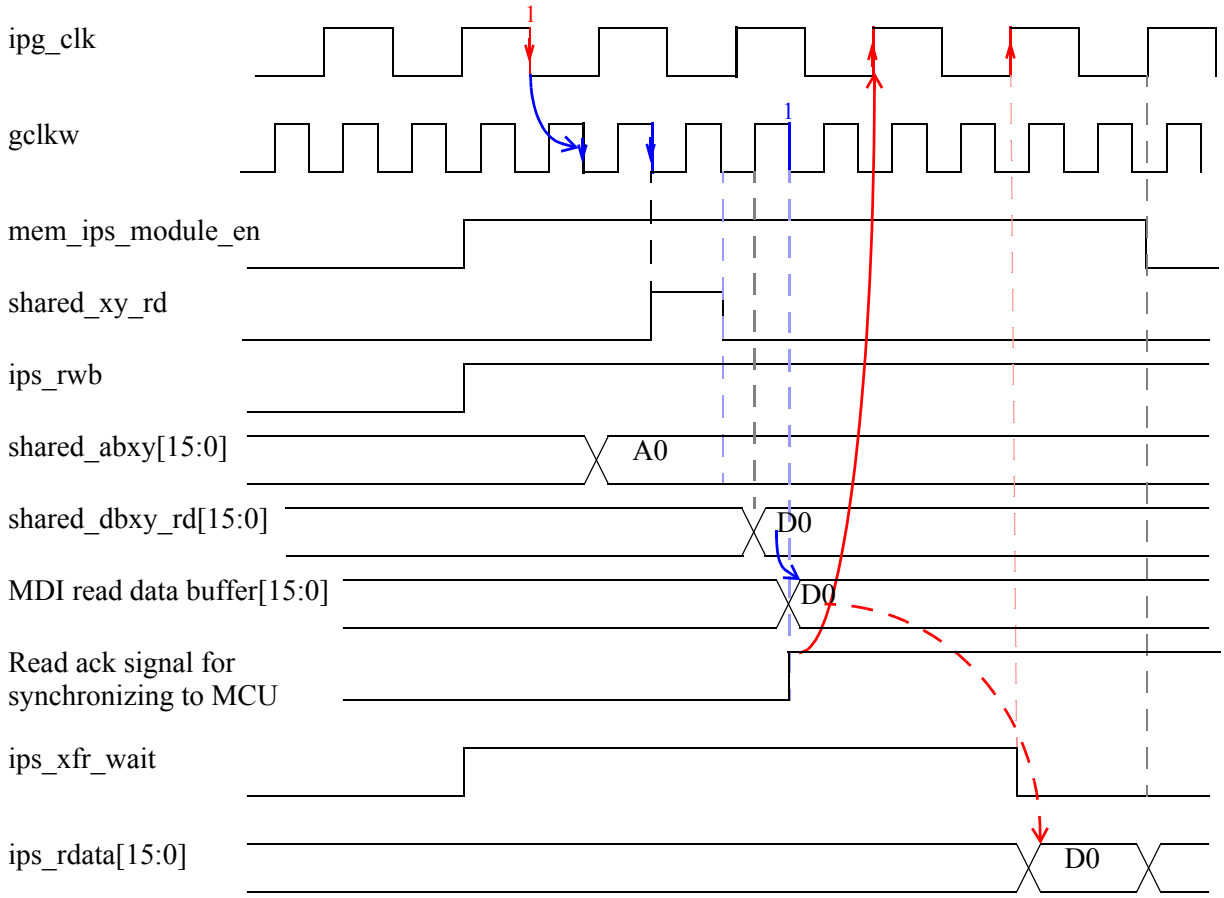
Access type	DSP clocks	MCU Cycles (minimum[1])	MCU Cycles (maximum[2])	Comments
Shared memory read	not active	2	2	The module select will remain for one cycle more than no. of wait states. So the complete access will be number of wait states plus one.
	active	3	6	
Shared memory write	not active	2	2	The module select will remain for one cycle more than no. of wait states. So the complete access will be number of wait states plus one
	active	2	5	
MCU-DSP shared memory contention	active	+ 0	+ 1	Added MCU stall. Assuming the DSP does not execute multiple 1-cycle instructions.
Control register access	any case	1	1	

1. Minimum case is if the DSP clock frequency is >> than the MCU frequency.
2. Assuming the DSP is not slower than the MCU, the maximum case is when the DSP clock frequency equals the MCU clock frequency

37.6.2.1 MDI Timing diagram for shared memory Read and Write

The MDI read and write to the shared memory timing diagrams are shown below. The Figure 37-18 on page 37-50 shows timing diagram for the Read MCU read access to shared memory and Figure 37-19 on page 37-51 shows timing diagram for the MCU write to the shared memory.

Figure 37-18. MDI read to shared memory timing diagram

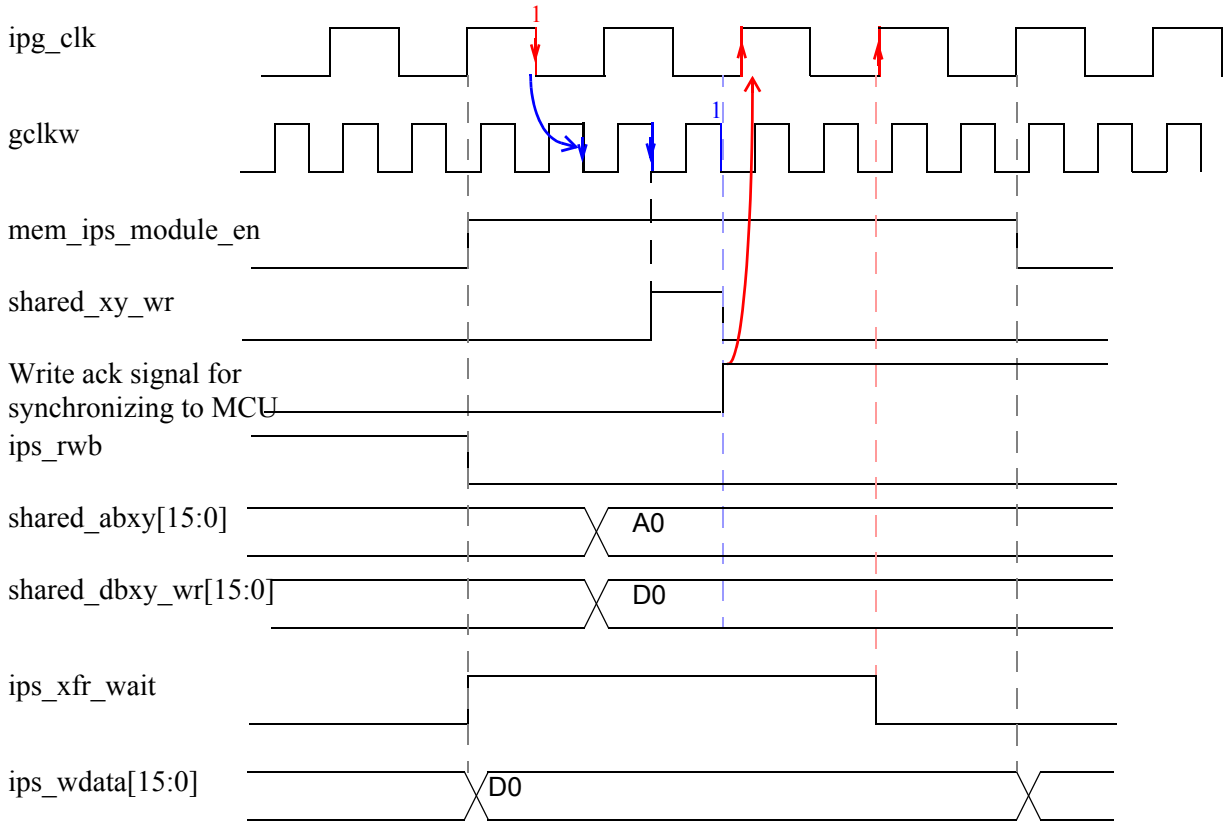


- The diagram shows the read timings of the MDI to the shared memory. The frequency is assumed to be DSP:MCU 2:1 for this diagram only. There can be any relation between the frequencies. Please refer Section 37.7.1.1, “MCU and DSP Frequency Ratio Dependency.” for frequency relation.

- The signal generated on the edge 1 of the MCU clock is synchronized on the DSP clock (bold blue edges on the blue arrow) to generate the read strobe.

- The signal generated on the edge 1 of the DSP clock is synchronized on the MCU clock (bold red edges on the red arrow) to give the acknowledge back to the MCU side that the read strobe is generated and the read data is stored in the read buffer.

Figure 37-19. MDI write to shared memory timing diagram



- The diagram shows the write timings of the MDI to the shared memory. The frequency is assumed to be DSP:MCU 2:1 for this diagram only. There can be any relation between the frequencies Please refer Section 37.7.1.1, “MCU and DSP Frequency Ratio Dependency.” for frequency relation.

- The signal generated on the edge 1 of the MCU clock is synchronized on the DSP clock (bold blue edges on the blue arrow) to generate the write strobe.

- The signal generated on the edge 1 of the DSP clock is synchronized on the MCU clock (bold red edges on the red arrow) to give the acknowledge back to the MCU side that the write strobe is generated.

37.6.3 Event Update Timing

Each processor’s MDI messaging side (DSP or MCU) has a hardware mechanism to send “event update requests” to the other processor’s side. An “event” is considered any information change that should be reflected at the Status Register of the receiving processor. The event update latency is the delay between the event being ready at one processor and the resulting update at the Status Register of the other processor.

The minimum event latency is 1 clock of the sending side + 2 1/2 clocks of the receiving side. The longest event latency is 6 clocks of the sending side + 6 1/2 clocks of the receiving side. The minimum case is if there is no event pending when the new event occurs. The maximum case is if the event occurred just after a previous event was sent to the other side.

37.7 Software Restrictions Summary

This section lists the software restrictions needed when accessing the MDI.

37.7.1 General Restrictions

37.7.1.1 MCU and DSP Frequency Ratio Dependency

For Neptune LTE, the software should use the MDI design at any clock ratio DSP:MCU from 0.25 (1:4) TO 10 (10:1) (inclusive). The figure 0.25 represents MCU runs four times the DSP rate), and 10.0 represents the design maximum ratio (DSP faster).

37.7.2 Write-After-Write to a Transmit Register

A write to a transmit register signals the receiver side that data is ready for retrieval. Writing again to the transmit register without verifying that the data was retrieved is prohibited since the transmitter side has no way of knowing the exact instance the receiver will attempt to retrieve the data. The transmitter side should wait for a “Transmitter Empty” interrupt, or poll the Transmitter Empty bit in the Status Register before attempting to write again to the transmit register. Failing to follow this restriction may result in wrong data being read at the receiver side.

37.7.2.1 Read-After-Read from a Receive Register

In the same way, the receiver processor should not read a receive register before receiving a “Receiver Full” interrupt or polling the Receiver Full bit in the Status Register.

37.7.2.2 Status Check Before Entering Stop

Before entering Stop mode, the relevant processor should verify that the Event Pending bit in the Status Register is clear (MEP in the MSR, or DEP in the DSR). If the Event Pending bit is still set, the relevant processor should wait and poll it until it is cleared, before executing the Stop instruction. For the DSP stop, please refer to Section 37.3.2, “DSP Low-Power Modes.” .

37.7.3 Restrictions at the DSP Side

37.7.3.1 Setting General Interrupt Requests (DGIR0,1)

Before setting a General Interrupt Request bit (DGIR0,1), the user must check that the DGIR bit is cleared - meaning that the general interrupt is not pending. Generally setting the DGIR bit while it is set will be ignored, but in some border cases it may issue a second interrupt. The restriction is to prevent this indeterministic behavior.

37.7.3.2 Proper use of IRQC, IRQD

IRQC and IRQD should be defined as level triggered in IPRC. They should be in the enabled state before the DSP enters Stop mode. IRQC should be cleared in the interrupt service routine by writing “1” to DWSC bit in the DSR; IRQD should be cleared by writing “1” to the DTIC bit in the DSR.

37.7.3.3 DSP Pipeline Effects on Registers

Due to the DSP pipeline structure, a read of a status register will not reflect the effect of register writes (e.g. write to DTR0) done up to three preceding instructions. Please refer to the 56600 family manual, appendix B, section 5 (Peripheral pipeline restrictions) for a description of the possible problems and work arounds. When testing the DEP bit in the DSR, one additional clock delay is needed above the 56600 manual description due to the MDI internal pipeline.

37.7.3.4 Continuous 1-Cycle Accesses to the Shared Memory

Consecutive 1-cycle accesses to the shared memory may cause the MCU to stall if it will try to read data from the shared memory in parallel. See a detailed example in Section 37.5, “Simultaneous Access of the DSP and MCU to the Memory,” above. Failing to follow this restriction may stall the MCU, but the code on both processors will be executed correctly

37.7.3.5 TRNM bit restriction while coming out of stop

If the DSP X-memory blocks other than the Shared memory block get the same DSP clock as the Shared Memory DSP clock, then, when DSP comes out of stop mode, the user should check that the TRNM bit in DSR of MDI is cleared before accessing any location in DSP X-memory from DSP. This is necessary because the MDI signals to switch the clock of the DSP X-memory from MCU clock to DSP clock when the DSP comes out of stop mode. If this bit is set, the DSP should not perform X-memory access.

37.7.4 Restrictions at the MCU Side

37.7.4.1 Byte-Wide Writes to the MDI Shared Memory

The MDI latches all 16 bits when receiving data written to it. In byte-wide writes, the MCU drives only the written byte, therefore bad data may be written to the unspecified byte of the shared memory location.

37.7.4.2 Writes to MCVR While the MC bit is Set

The MCVR (command vector register) could not be written to if the MC (MCU Command) bit is set, meaning that a command interrupt is pending. The user should check that the MC bit is cleared before writing data to the MCVR. A write to the MCVR while the MC bit is set will be ignored by hardware.

37.7.4.3 Setting the DWS bit While it is Set

The DWS bit should be tested as zero before setting it. Attempting to set the DWS bit while it is set will be ignored.

37.7.4.4 L1 Timer DSP Interrupt Override

If a Layer 1 Timer interrupt is given while the MTIR bit in the MSR is still set due to a previous interrupt, the second interrupt request will be lost.

37.7.4.5 MDI Reset (MDIR, DHR) Restrictions:

1. The user should verify that the DSP side is not engaged in MDI activity before asserting the MDIR bit. This could be done for example by issuing an NMI command prior to the MDIR operation.

2. Before issuing MDIR or DHR, the user should disable the DSP timer interrupt at the TIER in the Layer 1 Timer, and then verify that the DWS and DTIR bits in the MSR are cleared.
3. The MDIR bit could be set only if the DSP is not in Stop mode. Attempting to set the MDIR bit while the DSP is in Stop mode will be ignored by the hardware.
4. The user should not write to an MDI register in the instruction immediately following the assertion of MDIR, because the written data can be overridden by the reset value.
5. MDI software reset (using the MDIR or DHR bits at the MCR) may cancel an MCU write to the shared memory that is pending in the write buffer. The user should verify that no write is pending by verifying that the MSMP bit in the MSR is cleared.
6. After any one of the MDI reset conditions (MCU hardware reset, DSP hardware reset, MDI reset) the user should not access the shared memory before verifying that the reset sequence at the DSP side ended, by checking that the DRS bit at the MSR is cleared.

Chapter 38
ARM7 Platform (ARMPLAT)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01	Mike Fitzsimmons	Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	03/27/02	Lloyd Khuc	Removed the APIG modules Modified the AHBMUX module. Added the MDPI module Removed the security ROM from the AWPT module
0.2	04/04/02	Lloyd Khuc	Update MDPI I/O signals name
0.3	04/25/02	Lloyd Khuc	Added the Memory Size & Protetion section
0.4	05/30/02	Lloyd Khuc	Added Boot External Protetion table
0.5	05/07/03		Updated for LTE specification release.

Reference Documents

ARM7TDMI-S Technical Reference Manual
AMBA Specification Rev 2.0
IEEE 1149.1 JTAG Specification
IP Bus 2.0 Specification
Neptune Chapter 39 - Core Bus Arbitration Specification (CARB)
Neptune Chapter 43 - ARM Interrupt Controller Specification (AITC)
Neptune Chapter 44 - ARM Watchpoint Specification (AWPT)
Neptune Chapter 45 - Memory Controller Specification (MEM_CTL)
Neptune Chapter 47 - External Interface Module Specification (AEIM)
Neptune MDPI Specification
Neptune Chapter 63 - System JTAG Controller Specification (SJC)

Comments, corrections, and enhancements should be directed to:

Lloyd Khuc- TX32-PL31 Lloyd.Khuc@motorola.com or ra13510@email

38.1 Introduction

The ARM7 Platform consists of the ARM7TDMI-S processor, bus-arbitration logic, interfaces to external peripherals, RAM and ROM memory controllers, MCU to Dual Port RAM Peripheral Interface (MDPI), interrupt control, as well as watchpoint and breakpoints support. Figure 38-1 on page 38-4 shows a block diagram of the platform.

38.2 Performance Characteristics

The ARM7 Platform is designed to run at 52 MHz. Logic synthesis will be done using the 0.13 micron hip7 library characterized at 1.35 V and 105 degrees C. Processor and alternate bus master accesses to on chip RAM and ROM will be zero wait state. Word accesses to 32-bit AIPI devices will be three cycles for writes and two cycles for reads. Word accesses to 16-bit AIPI devices will take four clocks for write operations and two clocks for reads.

38.3 ARM7 Platform Submodules

The submodules of the platform are listed below along with short functional descriptions.

Note that accesses to undefined register space could have unexpected and potentially unwanted affects.

38.3.1 ARM7TDMI-S (A7S)

The ARM7TDMI-S is a member of the ARM family of general purpose 32-bit microprocessors. The architecture is based on RISC principles and supports two instruction sets:

- the 32-bit ARM instruction set
- the 16-bit Thumb instruction set

A JTAG port is provided along with associated signals to support debug related functions. The ARM7TDMI-S processor is fully synthesizable. Refer to the ARM7TDMI-S Technical Reference Manual for more information.

The ARM7 co-processor interface will not be utilized. All co-processor inputs will be tied off internally to the platform.

38.3.2 Core Bus Arbitration (CARB)

The platform's bus arbitration module permits off platform alternate bus masters to request and gain control of the internal R-AHB (reduced AHB) bus. In this manner, external alternate bus masters may access internal RAM/ROM as well as external memory and peripherals. The CARB module also performs several functions related to debug mode. See Section 38.7.5 and refer to Chapter 40, "ARM7 Platform Core Arbiter (CARB)," for more detail.

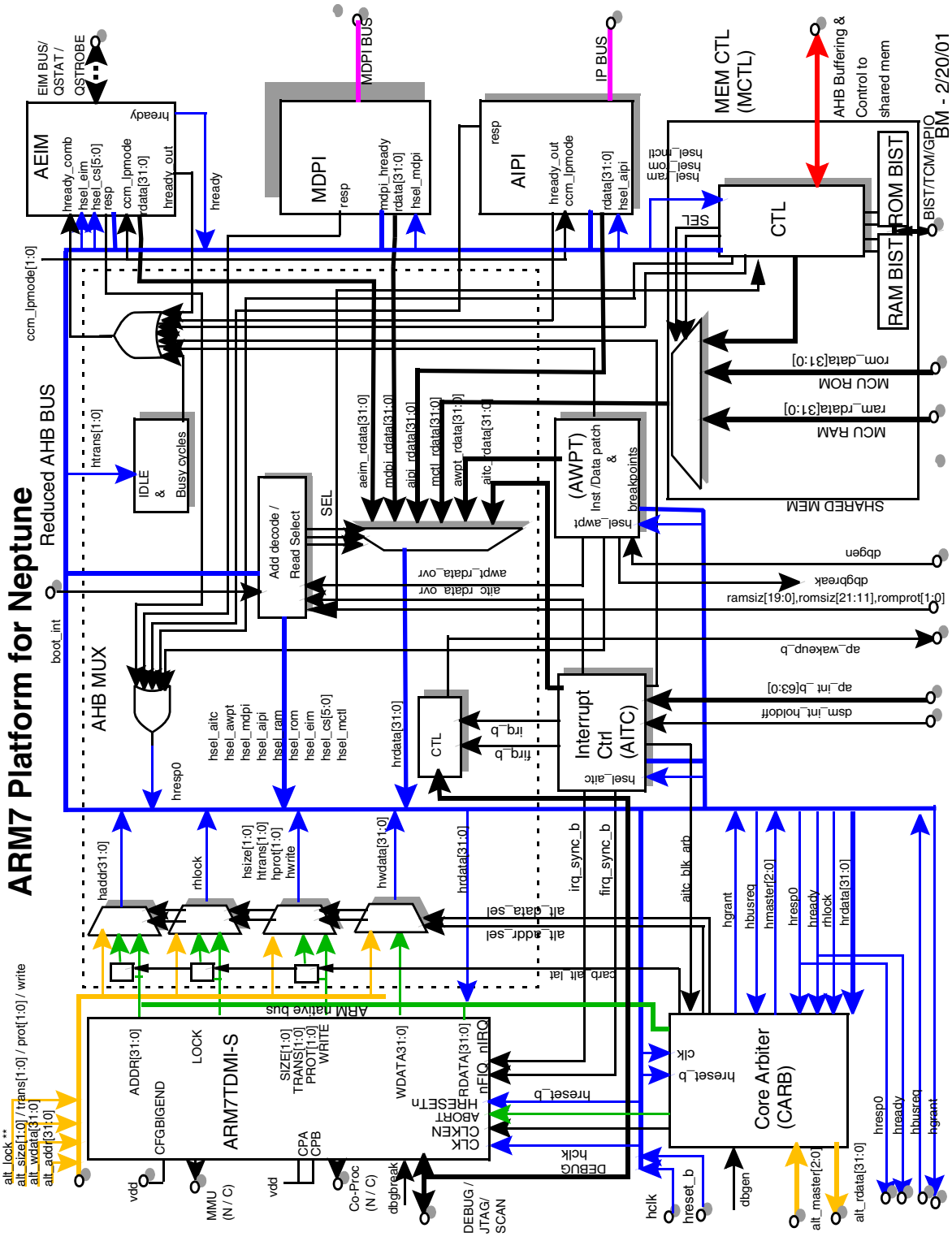


Figure 38-1. ARM7 Platform Block Diagram

38.3.3 R-AHB Mux (AHBMUX)

The AHBMUX module is outlined (dotted line box) in the block diagram of the platform shown in Figure 38-1 It contains logic to support centralized R-AHB slave address decoding (HSELx), alternate master bus ownership as well as the R-AHB read data muxing. Also contained in this module is logic to perform ARM7 reset and IDLE cycle HREADY generation. The AHBMUX is discussed in more detail in Section 38.9.

38.3.4 ARM External Interface Module (AEIM)

The AEIM module is attached to the R-AHB bus and interfaces to external memory. The AEIM is also responsible for the R-AHB bus watchdog function as well as support for SHOW cycles. Refer to the Neptune AEIM specification for more detail.

38.3.5 ARM IPBus Interface (AIPI)

The AIPI module interfaces the R-AHB to the external IPBus. The IPBus interface and its peripherals will conform to the IPBus Rev 2.0 specification. All IPBus peripherals will exist external to the arm7_platform. Refer to the Chapter 41, “R-AHB IP Interface (AIPI),” for more detail.

38.3.6 MCU to Dual Port RAM Peripheral Interface (MDPI)

The Memory Dual Port Peripheral Interface is a dual port ram memory sharing access between the ARM MCU Neptune_lts platform and the DSP.

38.3.7 ARM Interrupt Controller (AITC)

The ARM7 platform’s interrupt controller is called the AITC and is connected to the R-AHB as a slave device. It will generate the normal and fast interrupts to the ARM7 processor. Refer to the Neptune AITC specification for more detail.

38.3.8 ARM Watchpoint Module (AWPT)

The watchpoint module (AWPT) is connected to the R-AHB as a slave device. The AWPT snoops the R-AHB address bus and performs selective instruction and data patching by providing data and data select over-ride signals to the AHBMUX module. The address compare circuitry in this module can also be programmed to support breakpoints by driving the ARM7 DBGBREAK input. For more information refer to the Neptune AWPT specification.

38.3.9 Memory Controller (MEM_CTL)

The MEM_CTL module interfaces the R-AHB bus to zero wait-state ROM and RAM. The MEM_CTRL also provides R-AHB bus buffering and control for access to off platform shared memory. Refer to the Neptune MCU Memory Controller specification for more detail.

38.4 Platform Bus Support

The arm7_platform supports an internal bus as well as several external buses for peripheral and memory accesses. Figure 1-1 shows the platform's bus architecture. Each bus is briefly described below.

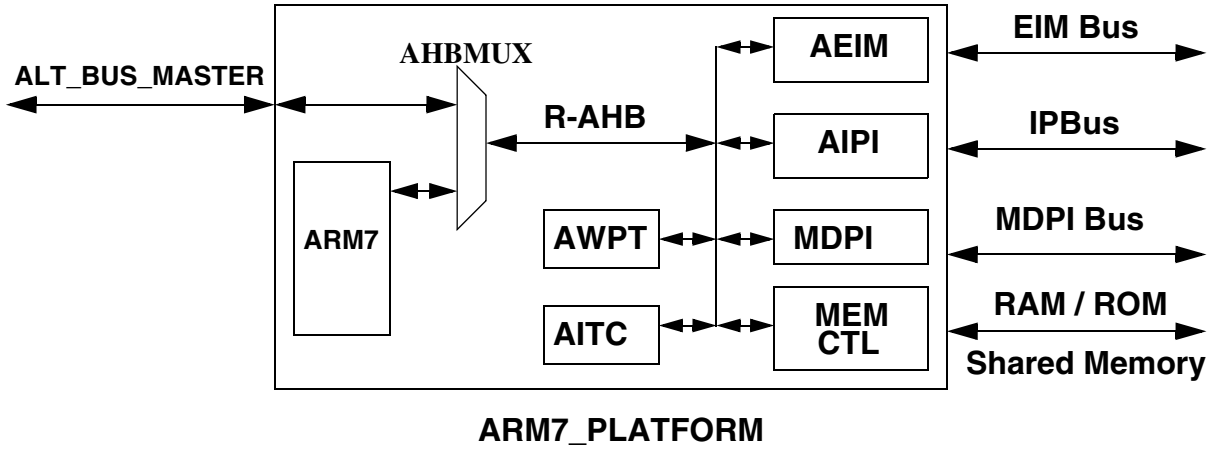


Figure 38-2. ARM7_PLATFORM Bus Structure

38.4.1 Internal Bus (R-AHB)

The R-AHB bus is the arm7_platform's internal bus, and stands for "reduced" AHB (ARM High Performance Bus). It is "reduced" in that the AHB retry, split, and burst operations are not supported.

The ARM7TDMI-S native bus, as well as the alternate bus master interface to the arm7_platform, are converted to the R-AHB in the AHBMUX module. The AHBMUX will drive and monitor all R-AHB I/O and interface the R-AHB to the ARM7 processor and to the alternate bus masters.

The AEIM, AIPI, MDPI, MEM_CTL, AITC, and AWPT modules are all R-AHB slaves within the platform. The ARM7 processor is the only R-AHB bus master inside the platform. Alternate bus masters are external to the platform and gain access to the R-AHB through the bus arbitration interface and CARB module. See Section 38.4.5.10.

Refer to the AMBA Specification, Rev 2.0, Chapter 3 - AMBA AHB for more information on the AHB bus. The AMBA AHB document is the reference source for the R-AHB bus as not all information contained in that document will be duplicated in this specification. Differences between the AHB and R-AHB will be documented in this specification. See Section 38.4.5.10.

Refer to Section 38.9 for a more detailed discussion of the AHBMUX module.

38.4.2 R-AHB Slave Signal List

The R-AHB bus slave's signal list is shown in Table 38-1. All signals function as documented in the AMBA Specification, Rev 2.0, Chapter 3 - AMBA AHB, except were noted. **The signal "Type" is referenced from the R-AHB slave's perspective.**

Table 38-1 is a list of the ARMPLAT module pins.

Table 38-1. RAHB Signal List

Pin List	Direction	Description
hclk	Input	AHB Reference Clock
hreset_b	Input	AHB Reset
hselx	Input	Slave Address decoder (lives in AHBMUX)
haddr[31:0]	Input	AHB Address Bus
htrans[1:0]	Input	AHB Transfer Type
hprot[1:0]	Input	AHB Processor Protection Control Indicators
hwrite	Input	AHB Write/Read Indicator
hsize[1:0]	Input	AHB Transfer Size Indicator
hwdata{31:0]	Input	AHB Write Data
hready_in	Input	AHB HREADY (Output of EIM) See Section 38.4.3.2, “AHB / R-AHB Bus Arbitration Differences.”
hready_out	Output	Slaves HREADY Output (To AHBMUX and AEIM). Non AHB signal. See Section 38.4.3.2, “AHB / R-AHB Bus Arbitration Differences.”
rdata[31:0]	Output	Read Data Out (input to AHBMUX which in turn drives HRDATA)
resp0	Output	Slave transfer error acknowledge. Used by AEIM and AHBMUX to generate HRESP0 which in turn will drive the cpu_abort input. Non AHB signal.
hmaster[2:0]	Output	R-AHB Bus Master Ownership 000 ARM7 001 GEM 010 DMAC 011 TCM 100 CCM 101 PORT 3 110 Reserved 111 Reserved, 111<000

For R-AHB slave timing, refer to “Electrical Specifications” on page 48.

38.4.3 R-AHB / AHB Bus Differences

38.4.3.1 Unsupported AHB Transactions

The “reduced” AHB bus does not support the AHB bus split, retry, and burst operations. For this reason, only one response signal is supported on the R-AHB, **resp0**, rather than the AHB specified **resp[1:0]**.

38.4.3.2 AHB / R-AHB Bus Arbitration Differences

The bus arbitration handshake for the ARM7_PLATFORM is not compliant with the AHB specification. In particular, alternate bus master addresses are driven onto the R-AHB bus off the same rising edge of **hclk** that **hgrant** is issued. See Chapter 40, “ARM7 Platform Core Arbiter (CARB),” for more detail on this and other differences between the two methods of bus arbitration.

38.4.3.3 HLOCK

The **hlock** signal is not driven on the R-AHB bus. The LOCK signal from the ARM7 processor is used by the CARB module to prevent alternate bus master requests from being granted during LOCK’ed processor transactions.

38.4.3.4 R-AHB Delayed Cycle Termination

SHOW cycle functionality in the AEIM can delay R-AHB transaction termination. Because of this, the **hready** signal used in the platform’s R-AHB implementation is a slightly modified version of the AHB specification. Each R-AHB slave device will drive **hready_out** on selected transaction termination. The AHBMUX module will OR all the slave **hready_out** signals together and send the resulting signal, **r_ahb_ready**, to the AEIM. This is necessary because the AEIM, during SHOW cycles, may delay the **hready** it drives onto the R-AHB bus. The **aeim_hready** output is connected to all R-AHB slave’s **hready_in** inputs and is the R-AHB’s HREADY signal. This is depicted in the arm7_platform block diagram, Figure 1. Each R-AHB slave is responsible for monitoring **hready_in** and staying in sync with the R-AHB bus state.

NOTE:

Each slave must drive HREADY_OUT, RDATA, and RESP0 until the EIM drives HREADY onto the R-AHB bus. Figure 38-3 on page 38-9 shows a non-delayed cycle termination. Figure 38-4 on page 38-9 depicts the SHOW cycle delay case.

A typical slave transaction is shown in Figure 1-2. The delayed SHOW cycle case is shown in Figure 1-3.

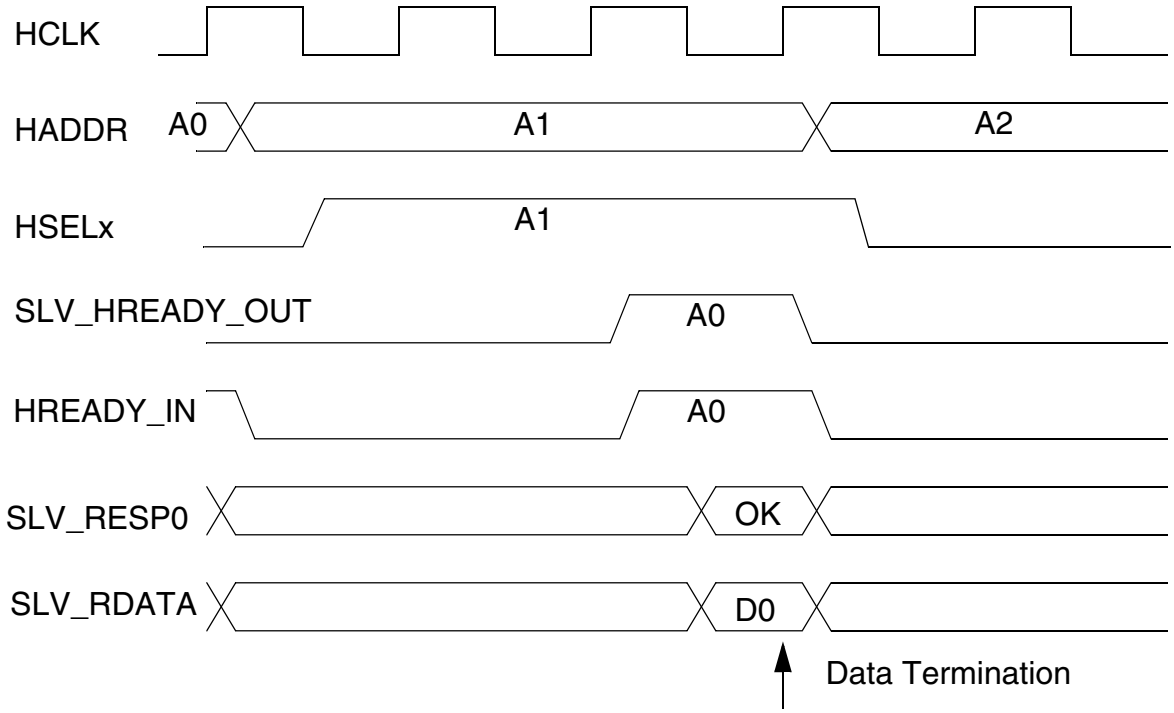


Figure 38-3. HREADY Typical Case

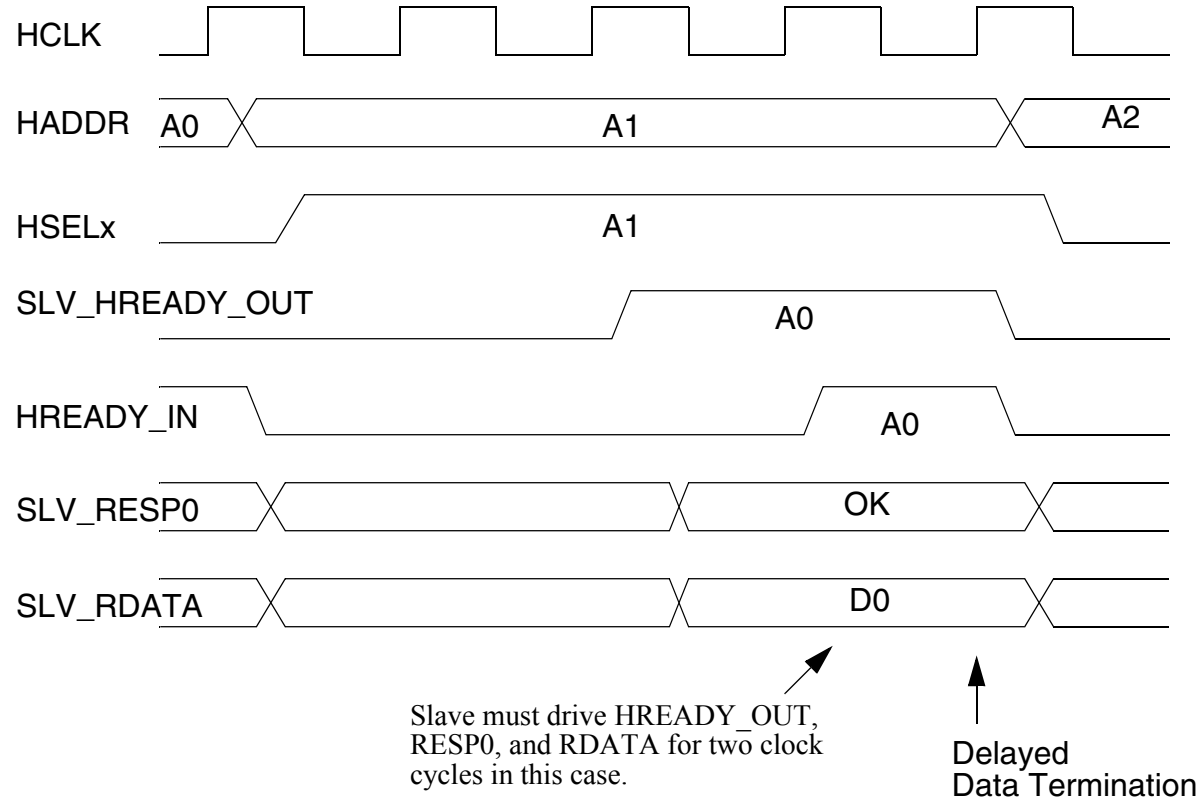


Figure 38-4. HREADY - Show Cycle Delay Case

38.4.3.5 Lower Order Address Bits

The ARM7TDMI-S processor specification requires slaves to ignore the bottom address bits. Therefore, the significant address bits of the R-AHB are listed in Table 38-2

Table 38-2. Significant Address Bits

HSIZE[1:0]	R-AHB Access	Significant Address Bits
00	Byte	HADDR[31:0]
01	Halfword	HADDR[31:1]
10	Word	HADDR[31:2]

NOTE:

The address bits below the significant bits should be ignored by all R-AHB slave devices. That is, there are NO unaligned transfers on the R-AHB.

38.4.3.6 IDLE - SEQ Cycles

When R-AHB bus ownership changes between the ARM7 processor and an alternate bus master, or between two alternate bus masters, the transaction types (**htrans[1:0]**) might incorrectly indicate a sequential cycle when in fact the address is NOT sequential. To insure R-AHB bus non-sequential and sequential transaction compliancy, the CARB module will output a signal to the AHBMUX module, **carb_force_nonseq**, which will negate **htrans[0]** forcing a non-sequential transfer. This signal will assert during the address phase of the first non-IDLE cycle following a change of bus ownership.

38.4.3.7 Two- Cycle AWPT Aborts

The AWPT module does not follow the AHB bus two-cycle abort protocol. This is primarily because the module can abort transactions (assert **hresp0**) for which it was not the intended target. For aborts of this nature, the AWPT does not control the assertion of **hready_out** - this is the target slave's responsibility. For this reason the AEIM module will monitor the **awpt_resp** signal. When **awpt_resp** is asserted coincident with **r_ahb_hready** assertion, the AEIM will delay the assertion of **hready** by a single clock (non SHOW cycle case). The AWPT module will keep **awpt_resp** asserted until it samples **hready** asserted. The transaction on the R-AHB bus will thereby conform with the AHB two-cycle abort protocol.

38.4.3.8 BUSY Cycles

Even though there is no external coprocessor in the arm7_platform, the ARM7TDMI-S is capable of generating an **htrans[1:0]=01**, or "coprocessor register transfer cycle", because internal coprocessor number 14 is the debug controller. This transaction type corresponds to BUSY on the R-AHB bus. All BUSY cycles will be terminated with an **hready** generated by the AHBMUX module.

38.4.4 External Buses

The ARM7 platform has several buses emanating to/from the top-level. These include:

- Alternate Bus Master Interface
- EIM Bus
- IP Bus
- MDPI Bus
- Shared Memory Bus

38.4.5 Memory Size & Protection

For each memory, the last address of that memory is determined, based on the assumption that the memory starts at $0000_0000. For example, a 256K byte memory would end at $0003_FFFF. Then the associated address sizes bits are tied high or low to match this address. The BIST for that memory is matched to that exact size. The select is matched to either the exact size or the closest larger size available.

There is also a need for selectively protecting a portion of ROM. Two bits are used for this purpose. The ROM size discussions will initially treat the ROM as unprotected to make it easier to understand. The protection scheme and it's effects will be added after.

An overall system block diagram is shown in **Figure37-5**.

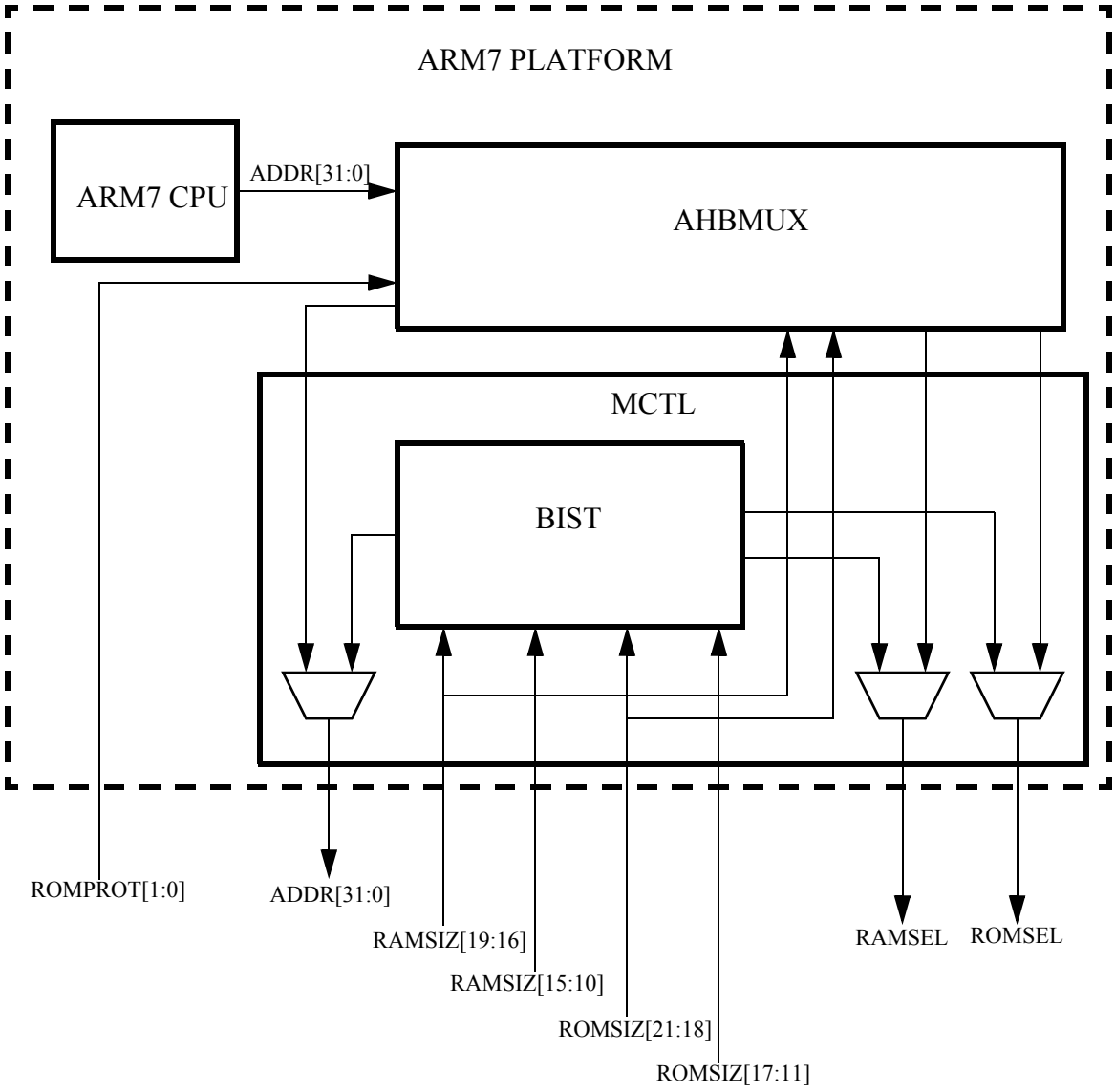


Figure 38-5. Overall Block Diagram

38.4.5.1 Memory Size & Protection Summary

On the hardened core platform, there will be a set of size select bits for the ROM and another set for the RAM. There will also be a pair of bits to select the protection on the ROM. This protection is in addition to that offered by the MSU.

38.4.5.2 RAM

The RAM will be selectable in 1K increments from 1K up to 1M via 10 size bits.

The BIST engine will use all the select bits to identify the exact size of the RAM.

The select logic in the AHBMUX will use the upper four bits to determine one of five select space sizes. The sizes are 64KB, 128KB, 256KB, 512KB and 1MB. The physical memory size will either match one of these sizes or it will match the next closest larger.

If the RAM is less than 32KB it will be aliased in the 64KB space. The number of times it is aliased depends on the physical size of the memory. If the MSU is used to protect a portion of a memory which is aliased, the user (supervisor) should set it to protect that same portion in each of the aliased locations as well. The MSU could also be used to protect the whole memory space, including all the aliased locations.

The RAM will reside in the memory map ending at $03FF_FFFF.

The same RAM will also reside in the memory map ending at $FFFF_FFFF. This RAM will only be accessible in supervisor mode.

If the RAM is not a power of 2 and does not fill the entire select space, then accesses to unimplented RAM within the select space will be ignored but will not cause an abort.

Accesses to either 32MB RAM space which are outside the select space will cause an abort.

38.4.5.3 ROM

The ROM will be selectable in 2K increments from 2K up to 4M via 11 size bits.

The BIST engine will use all the select bits to identify the exact size of the ROM.

The select logic in the AHBMUX will use the upper four bits to determine one of five select space sizes. The sizes are 256KB, 512KB, 1MB, 2MB and 4MB. The physical memory size will either match one of these sizes or it will match the closest larger.

If the ROM is less than 128KB it will be aliased in the 256KB space. The number of times it is aliased depends on the physical size of the memory. If the MSU is used to protect a portion of a memory which is aliased, it must protect that same portion in each of the aliased locations as well. The MSU could also be used to protect the whole memory space, including all the aliased locations.

If the ROM is not a power of 2 and does not fill the entire select space, then accesses to unimplented ROM within the select space will be ignored but will not cause an abort.

Accesses to the 32MB ROM space which are outside the select space will cause an abort.

38.4.5.4 ROM Protection

There are four protection choices for protection of ROM space via two protect bits.

One choice is no protection. The ROM starts at $0000_0000 and fills the address space in a linear fashion.

The other three choices create a 4M protected space starting from $0000_0000. One of these choices is for all the memory to be protected (supervisor access only). For this case the ROM starts at $0000_0000 and fills the address space in a linear fashion.

The next choice is to protect 16KB located at $0000_0000. The remaining ROM is not protected. It is located in the memory map starting at the 4MB boundary plus 16KB ($0040_4000).

The final choice is to protect 64KB located at $0000_0000. The remaining ROM is not protected. It is located in the memory map starting at the 4MB boundary plus 64KB ($0041_0000).

Non-supervisor accesses to protected space will cause an abort.

38.4.5.5 Memory Select Sizes

The AHBMUX generates the selects for the MCU ROM and MCU RAM. The size bits at the edge of the platform are each tied high or low when the MCU is integrated. This allows the integrator to determine the amount of address space these memories occupy for a particular chip. The select details are shown in **Table 37-3** for the MCU RAM and **Table 37-4** of the MCU ROM.

Table 38-3. RAM Select

RAMSIZ19	RAMSIZ18	RAMSIZ17	RAMSIZ16	SIZE (Bytes)	START ADDRESS	END ADDRESS
0	0	0	0	65,536	$03FF_0000	$03FF_FFFF
0	0	0	1	131,072	$03FE_0000	$03FF_FFFF
0	0	1	X	262,144	$03FC_0000	$03FF_FFFF
0	1	X	X	524,288	$03F8_0000	$03FF_FFFF
1	X	X	X	1,048,576	$03F0_0000	$03FF_FFFF

Table 38-4. ROM Select

ROMSIZ21	ROMSIZ20	ROMSIZ19	ROMSIZ18	SIZE (Bytes)	START ADDRESS	END ADDRESS
0	0	0	0	262,144	$0000_0000	$0003_FFFF
0	0	0	1	524,288	$0000_0000	$0007_FFFF
0	0	1	X	1,048,576	$0000_0000	$000F_FFFF
0	1	X	X	2,097,152	$0000_0000	$001F_FFFF
1	X	X	X	4,194,304	$0000_0000	$003F_FFFF

NOTE: *For the BIST engine to work properly, the size select bits must be set correctly even though the select logic allows them to be "don't cares".*

The block diagram in **Figure 37-6**, shows the generation of memory selects in the AHBMUX. The upper addresses are compared to determine if the address is in the RAM range or in the ROM range. This generates signals which gate the remaining operations.

If the address is in the ROM range, then address bits 18 to 21 (depending on the size selection) may be used to further qualify the access. If all address criteria are met, then a select of the ROM occurs.

If the address is in the RAM range, then address bits 16 to 19 (depending on the size selection) may be used to further qualify the access. If all address criteria are met, then a select of the RAM occurs.

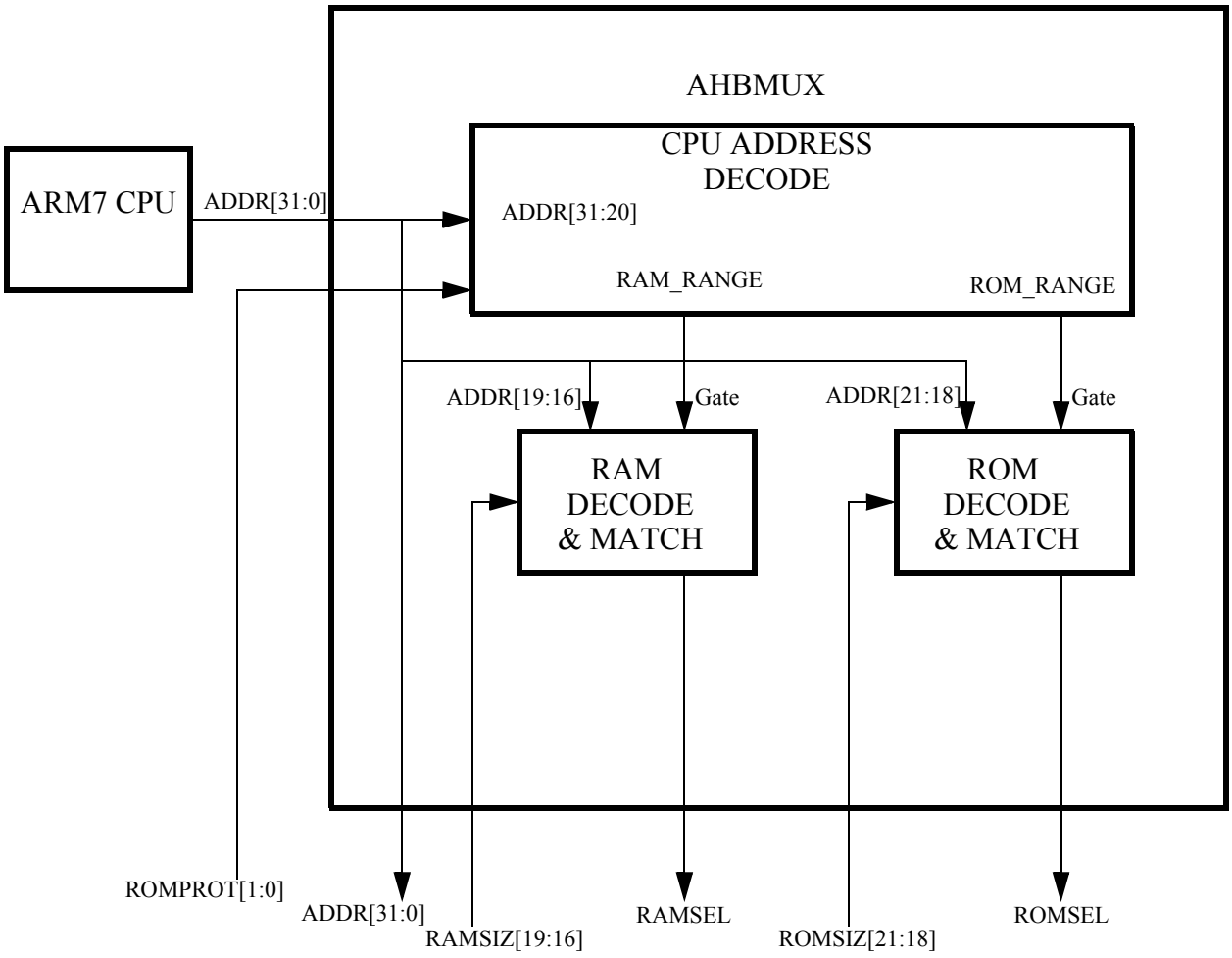


Figure 38-6. Block Diagram of Select Operation in AHBMUX

38.4.5.6 Effects of Memory Map Space

With a limited number of memory size selects, there will be times when the memory space ends up larger than the memory. In most cases the memory is a non power of two size and nearest larger power of two size select space. This leaves a portion of the address space in which the memory can be selected, but there is no physical array present. For the current memories, any read of this space will return unknown data and any write to this space will have no effect. If different memory is considered in the future, it needs to be checked for this requirement.

In some cases a memory smaller than half the smallest select is used. In this case the memory will be aliased in the space, that is, it will have more than one location where it is accessible. The user must be aware that this is the same memory appearing in multiple places rather than different memories. If the memory is a non power of two, there will also be “holes” in which the memory is accessed, but does not have any physical array present.

38.4.5.7 BIST Sizes

The BIST engines for the MCU ROM and MCU RAM use the size inputs to determine the last address of the memory. The last address of the memory is determined assuming the memory starts at $0000_0000. The address size bits are each tied high or low matching the last address. Note that these BIST engines work on 32 bit words only, so BIST address bit 0 is the same as physical address bit 2. A representative set of RAM BIST sizes is shown in **Table 37-5.** A representative set of ROM BIST sizes is shown in **Tbale 37-6**. Note that neither of these tables lists all of the more than 1000 combinations.

Table 38-5. RAM BIST Sizes

RAM SIZ 19	RAM SIZ 18	RAM SIZ 17	RAM SIZ 16	RAM SIZ 15	RAM SIZ 14	RAM SIZ 13	RAM SIZ 12	RAM SIZ 11	RAM SIZ 10	SIZE (Bytes)	BIST START ADDRESS	BIST END ADDRESS
0	0	0	0	0	0	0	0	0	0	1024	$0000_0000	$0000_03FF
0	0	0	0	0	0	0	0	0	1	2048	$0000_0000	$0000_07FF
0	0	0	0	0	0	0	0	1	0	3072	$0000_0000	$0000_0BFF
0	0	0	0	0	0	0	0	1	1	4096	$0000_0000	$0000_0FFF
0	0	0	0	0	0	0	1	0	0	5120	$0000_0000	$0000_13FF
:												
0	0	0	0	0	0	0	1	1	0	7168	$0000_0000	$0000_1BFF
0	0	0	0	0	0	0	1	1	1	8192	$0000_0000	$0000_1FFF
0	0	0	0	0	0	1	0	0	0	9216	$0000_0000	$0000_23FF
:												
0	0	0	0	0	0	1	1	1	0	15,360	$0000_0000	$0000_3BFF
0	0	0	0	0	0	1	1	1	1	16,384	$0000_0000	$0000_3FFF
0	0	0	0	0	1	0	0	0	0	17,408	$0000_0000	$0000_43FF
:												
0	0	0	0	0	1	1	1	1	0	31,744	$0000_0000	$0000_7BFF
0	0	0	0	0	1	1	1	1	1	32,768	$0000_0000	$0000_7FFF
0	0	0	0	1	0	0	0	0	0	33,792	$0000_0000	$0000_83FF
:												
0	0	0	0	1	1	1	1	1	0	64,512	$0000_0000	$0000_FBFF
0	0	0	0	1	1	1	1	1	1	65,536	$0000_0000	$0000_FFFF
0	0	0	1	0	0	0	0	0	0	66,560	$0000_0000	$0001_03FF
:												
0	0	0	1	1	1	1	1	1	0	130,048	$0000_0000	$0001_FBFF
0	0	0	1	1	1	1	1	1	1	131,072	$0000_0000	$0001_FFFF
0	0	1	0	0	0	0	0	0	0	132,096	$0000_0000	$0002_03FF

Table 38-5. RAM BIST Sizes

RAM SIZ 19	RAM SIZ 18	RAM SIZ 17	RAM SIZ 16	RAM SIZ 15	RAM SIZ 14	RAM SIZ 13	RAM SIZ 12	RAM SIZ 11	RAM SIZ 10	SIZE (Bytes)	BIST START ADDRESS	BIST END ADDRESS
:												
0	0	1	1	1	1	1	1	1	0	261,120	$0000_0000	$0003_FBFF
0	0	1	1	1	1	1	1	1	1	262,144	$0000_0000	$0003_FFFF
0	1	0	0	0	0	0	0	0	0	263,168	$0000_0000	$0004_03FF
:												
0	1	1	1	1	1	1	1	1	0	523,264	$0000_0000	$0007_FBFF
0	1	1	1	1	1	1	1	1	1	524,288	$0000_0000	$0007_FFFF
1	0	0	0	0	0	0	0	0	0	525,312	$0000_0000	$0008_03FF
:												
1	1	1	1	1	1	1	1	1	0	1,047,552	$0000_0000	$000F_FBFF
1	1	1	1	1	1	1	1	1	1	1,048,576	$0000_0000	$000F_FFFF

Table 38-6. ROM BIST Sizes

ROM SIZ 21	ROM SIZ 20	ROM SIZ 19	ROM SIZ 18	ROM SIZ 17	ROM SIZ 16	ROM SIZ 15	ROM SIZ 14	ROM SIZ 13	ROM SIZ 12	ROM SIZ 11	SIZE (Bytes)	BIST START ADDRESS	BIST END ADDRESS
0	0	0	0	0	0	0	0	0	0	0	2048	$0000_0000	$0000_07FF
0	0	0	0	0	0	0	0	0	0	1	4096	$0000_0000	$0000_0FFF
0	0	0	0	0	0	0	0	0	1	0	6144	$0000_0000	$0000_17FF
0	0	0	0	0	0	0	0	0	1	1	8192	$0000_0000	$0000_1FFF
0	0	0	0	0	0	0	0	1	0	0	10,240	$0000_0000	$0000_27FF
:													
0	0	0	0	0	0	0	0	1	1	0	14,336	$0000_0000	$0000_37FF
0	0	0	0	0	0	0	0	1	1	1	16,384	$0000_0000	$0000_3FFF
0	0	0	0	0	0	0	1	0	0	0	18,432	$0000_0000	$0000_47FF
:													
0	0	0	0	0	0	0	1	1	1	0	30,720	$0000_0000	$0000_77FF
0	0	0	0	0	0	0	1	1	1	1	32,768	$0000_0000	$0000_7FFF
0	0	0	0	0	0	1	0	0	0	0	34,816	$0000_0000	$0000_87FF
:													
0	0	0	0	0	0	1	1	1	1	0	63,488	$0000_0000	$0000_F7FF
0	0	0	0	0	0	1	1	1	1	1	65,536	$0000_0000	$0000_FFFF
0	0	0	0	0	1	0	0	0	0	0	67,584	$0000_0000	$0001_07FF
:													
0	0	0	0	0	1	1	1	1	1	0	129,024	$0000_0000	$0001_F7FF
0	0	0	0	0	1	1	1	1	1	1	131,072	$0000_0000	$0001_FFFF
0	0	0	0	1	0	0	0	0	0	0	133,120	$0000_0000	$0002_07FF
:													
0	0	0	0	1	1	1	1	1	1	0	260,096	$0000_0000	$0003_F7FF
0	0	0	0	1	1	1	1	1	1	1	262,144	$0000_0000	$0003_FFFF

Table 38-6. ROM BIST Sizes

ROM SIZ 21	ROM SIZ 20	ROM SIZ 19	ROM SIZ 18	ROM SIZ 17	ROM SIZ 16	ROM SIZ 15	ROM SIZ 14	ROM SIZ 13	ROM SIZ 12	ROM SIZ 11	SIZE (Bytes)	BIST START ADDRESS	BIST END ADDRESS
0	0	0	1	0	0	0	0	0	0	0	264,192	$0000_0000	$0004_07FF
:													
0	0	0	1	1	1	1	1	1	1	0	522,240	$0000_0000	$0007_F7FF
0	0	0	1	1	1	1	1	1	1	1	524,288	$0000_0000	$0007_FFFF
0	0	1	0	0	0	0	0	0	0	0	526,336	$0000_0000	$0008_07FF
:													
0	0	1	1	1	1	1	1	1	1	0	1,046,528	$0000_0000	$000F_F7FF
0	0	1	1	1	1	1	1	1	1	1	1,048,576	$0000_0000	$000F_FFFF
0	1	0	0	0	0	0	0	0	0	0	1,050,624	$0000_0000	$0010_07FF
:													
0	1	1	1	1	1	1	1	1	1	0	2,095,104	$0000_0000	$001F_F7FF
0	1	1	1	1	1	1	1	1	1	1	2,097,152	$0000_0000	$001F_FFFF
1	0	0	0	0	0	0	0	0	0	0	2,099,200	$0000_0000	$0020_07FF
:													
1	1	1	1	1	1	1	1	1	1	0	4,192,256	$0000_0000	$003F_F7FF
1	1	1	1	1	1	1	1	1	1	1	4,194,304	$0000_0000	$003F_FFFF

A block diagram showing the configuration for one aspect of the RAM BIST engine is shown in **Figure 37-7**. In this block diagram the BIST will generate all the RAM addresses starting with the BIST address of all zeros and incrementing by 32-bit words until the last address is reached. Note that the BIST always accesses 32-bit words and that it always starts the memory at an address of all zeros. By contrast, physical memory is addressed on a byte basis.

The end address register gets the high order addresses from the RAM size selects, physical address bits 19:10, which are the same as BIST address bits 17:8. The low order bits of the BIST end address are all ones. These are BIST address bits 7:0.

With this configuration the incrementor can generate all the RAM word addresses. The generated address is rescaled as it leaves the BIST. Since the RAM is high aligned in memory, the address is inverted as well (the actual RTL is different, but this aids in the general understanding). After this, physical address bits 1:0 are tacked on as zeros so that the start of a 32-bit word is always selected.

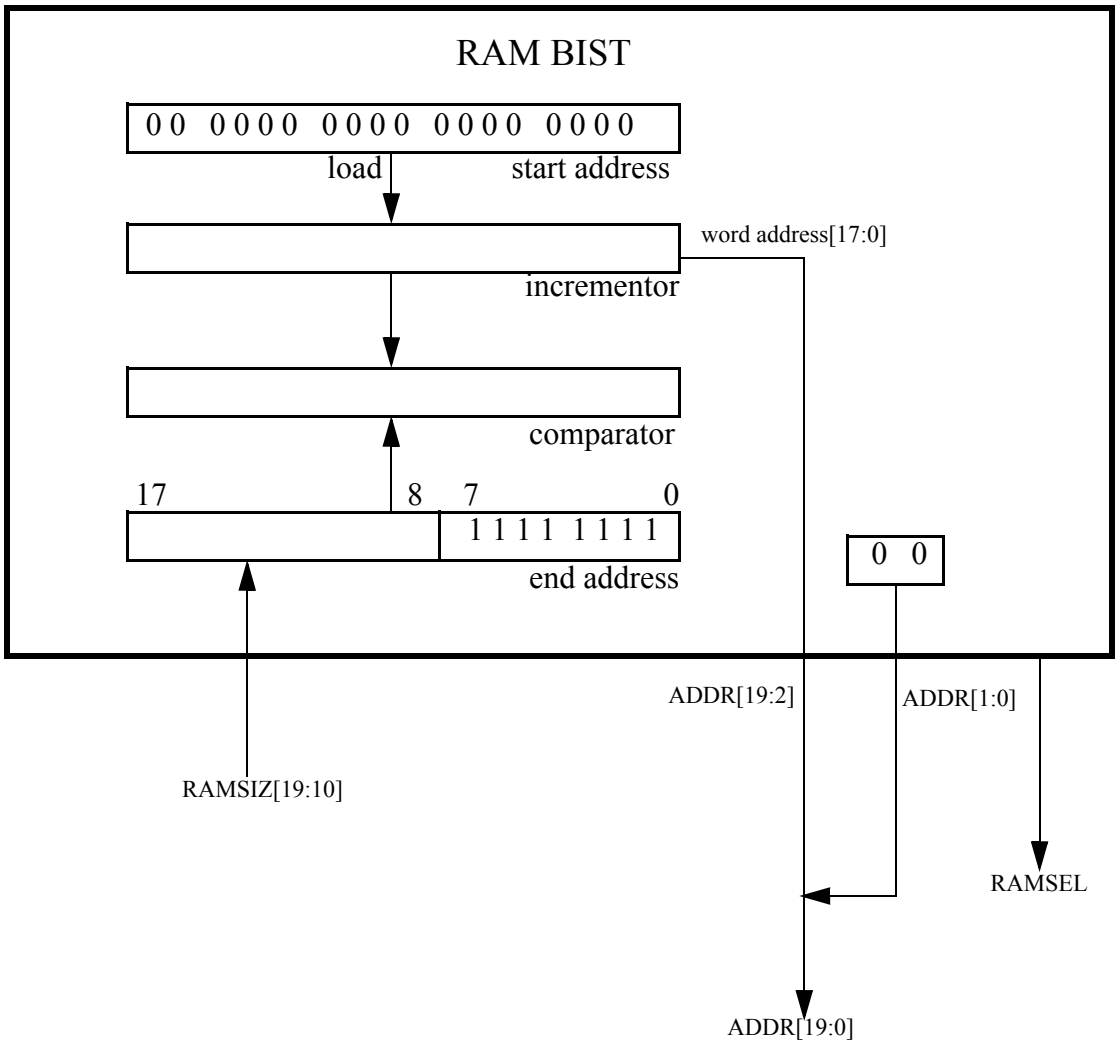


Figure 38-7. RAM BIST Block Diagram (One Configuration)

In a similar fashion, a block diagram showing the configuration for one aspect of the ROM BIST engine is shown in **Figure 37-8**. In this block diagram the BIST will generate all the ROM addresses starting with the BIST address of all zeros and incrementing by 32-bit words until the last address is reached. Note that the BIST always accesses 32-bit words and that it always starts the memory at an address of all zeros. By contrast, physical memory is addressed on a byte basis.

The end address register gets the high order addresses from the ROM size selects, physical address bits 21:11, which are the same as BIST address bits 19:9. The low order bits of the BIST end address are all ones. This is BIST address bits 8:0.

With this configuration the incrementor can generate all the ROM word addresses. The generated address is rescaled as it leaves the BIST. After this, physical address bits 1:0 are tacked on as zeros so that the start of a 32-bit word is always selected.

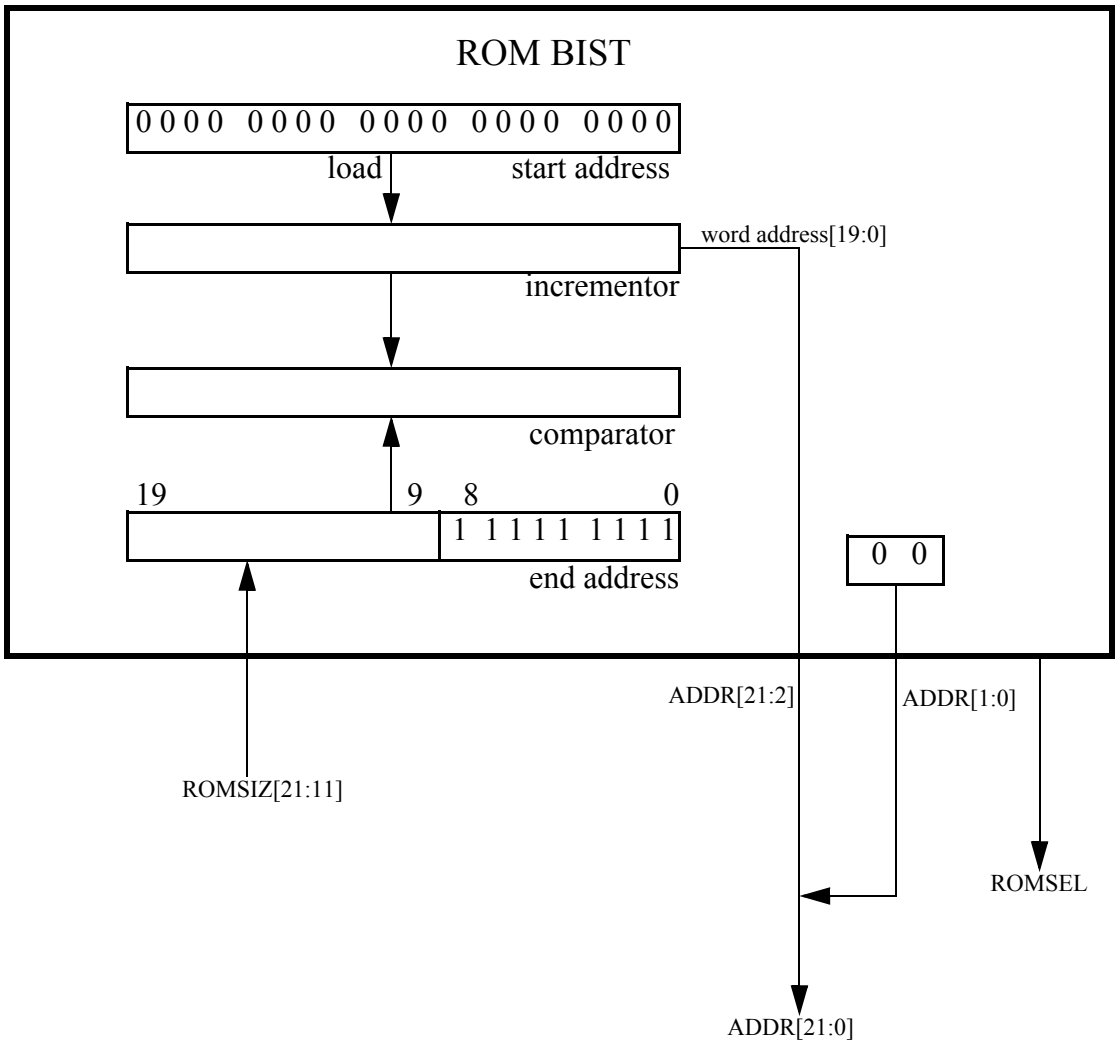


Figure 38-8. ROM BIST Block Diagram (One Configuration)

38.4.5.8 Examples

512K (524,288) Byte RAM Example:

We determine the last address is $0007_FFFF, assuming the RAM had a start address of $0000_0000. Matching these address bits to the RAM size selects, we tie RAMSIZ[19] to 0 and RAMSIZ[18:10] to 1. This gives us a BIST size that matches the memory size as seen in Table 38-5.

This also gives us a mapping select which is an exact match as seen in Table 38-3. The RAM will be in the memory map at $03F8_0000 to $03FF_FFFF.

458,752 Byte RAM Example:

We determine the last address is $0006_FFFF, assuming the RAM had a start address of $0000_0000. Matching these address bits to the RAM size selects, we tie RAMSIZ[19] to 0, RAMSIZ[18:17] to 1, RAMSIZ[16] to 0 and RAMSIZ[15:10] to 1. This gives us a BIST size that matches the memory size, (this is one of the choices not shown in Table 38-5).

This also gives us a mapping select which is 524,288 bytes (slightly larger) as seen in Table 38-3. The RAM will be in the memory map at $03F8_0000 to $03FF_FFFF. In this case the first portion of space will decode the RAM, but there will not be any array present from $03F8_0000 to $03F8_FFFF.

32K (32,768) Byte RAM Example:

We determine the last address is $0000_7FFF, assuming the RAM had a start address of $0000_0000. Matching these address bits to the RAM size selects, we tie RAMSIZ[19:15] to 0, and RAMSIZ[14:10] to 1. This gives us a BIST size that matches the memory size, as seen in Table 38-5.

This also gives us a mapping select which is 65,536 bytes (larger) as seen in Table 38-3. The RAM will be in the memory map at $03FF_0000 to $03FF_FFFF. In this case the RAM will be aliased, appearing in the map twice: at $03FF_0000 to $03FF_7FFF and at $03FF_8000 to $03FF_FFFF. The second location should be used by the software writers, since this is consistent with this high aligned memory.

2M (2,097,152) Byte ROM Example:

We determine the last address is $001F_FFFF, assuming the ROM had a start address of $0000_0000. Matching these address bits to the ROM size selects, we tie ROMSIZ[21] to 0 and ROMSIZ[20:11] to 1. This gives us a BIST size that matches the memory size as seen in Table 38-6.

This also gives us a mapping select which is an exact match as seen in Table 38-4. The ROM will be in the memory map at $0000_0000 to $001F_FFFF.

1,835,008 Byte ROM Example:

We determine the last address is $001B_FFFF, assuming the ROM had a start address of $0000_0000. Matching these address bits to the ROM size selects, we tie ROMSIZ[21] to 0, ROMSIZ[20:19] to 1, ROMSIZ[18] to 0 and ROMSIZ[17:11] to 1. This gives us a BIST size that matches the memory size, (this is one of the choices not shown in Table 38-6).

This also gives us a mapping select which is 2,097,152 bytes (slightly larger) as seen in Table 38-4. The ROM will be in the memory map at $0000_0000 to $001F_FFFF. In this case the last portion of space will decode the ROM, but there will not be any array present from $001C_0000 to $001F_FFFF.

131,072 Byte ROM Example:

We determine the last address is $0001_FFFF, assuming the ROM had a start address of $0000_0000. Matching these address bits to the ROM size selects, we tie ROMSIZ[21:17] to 0, and ROMSIZ[16:11] to 1. This gives us a BIST size that matches the memory size, as seen in Table 38-6.

This also gives us a mapping select which is 262,144 bytes (larger) as seen in Table 38-4. The ROM will be in the memory map at $0000_0000 to $0003_FFFF. In this case the ROM will be aliased, appearing in the map twice: at $0000_0000 to $0001_FFFF and at $0002_0000 to $0003_FFFF. The first location should be used by the software writers, since this is consistent with this low aligned memory.

38.4.5.9 Another Level - Adding ROM Protection

In order to provide ROM protection without using up some channels of the MSU, we provide two select bits which provide four protection choices. The protection is accomplished by blocking a portion of the memory map and then shifting the unprotected memory to a different portion of the map as shown in **Table 37-7**.

Table 38-7. ROM Protection Select in Boot Internal (Normal)

ROMPROT1	ROMPROT0	Protected Size (Bytes)	Protected Start Address	Protected End Address	Unprotected Start Address	Unprotected End Address
0	0	none	-	-	$0000_0000	$003F_FFFF
0	1	16,384	$0000_0000	$0000_3FFF	$0040_4000	$007F_FFFF
1	0	65,536	$0000_0000	$0000_FFFF	$0041_0000	$007F_FFFF
1	1	all (4M)	$0000_0000	$003F_FFFF	-	-

Table 38-8. ROM Protection Select in Boot External

ROMPROT1	ROMPROT1	Protected Size (Bytes)	Protected Start Address	Protected End Address	Unprotected Start Address	Unprotected End Address
0	0	none	-	-	$1000_0000	$103F_FFFF
0	1	16,384	$1000_0000	$1000_3FFF	$1040_4000	$107F_FFFF
1	0	65,536	$1000_0000	$1000_FFFF	$1041_0000	$107F_FFFF
1	1	all (4M)	$1000_0000	$103F_FFFF	-	-

The ROM size will still determine the size of the overall ROM. The selected ROM protection will then determine the placement and protection of the ROM. First the ROM will fill the protected space. If there is more ROM left after the protected space has been filled, then the ROM continues with the first location of the unprotected space until the end of the ROM.

Note that in both the no protection and all protection cases, the ROM is in the first 4M of the memory map. Also, if protection is selected, it always includes the vector space.

Figure 37-9 shows an example of the protection choices with a 2M ROM (boot internal mode).

Figure 37-10 shows an example of the protection choices with a 4M ROM (boot internal mode).
Figure 37-11 shows an example of the protection choices with a 2M ROM (boot external mode).
Figure 37-12 shows an example of the protection choices with a 4M ROM (boot external mode).

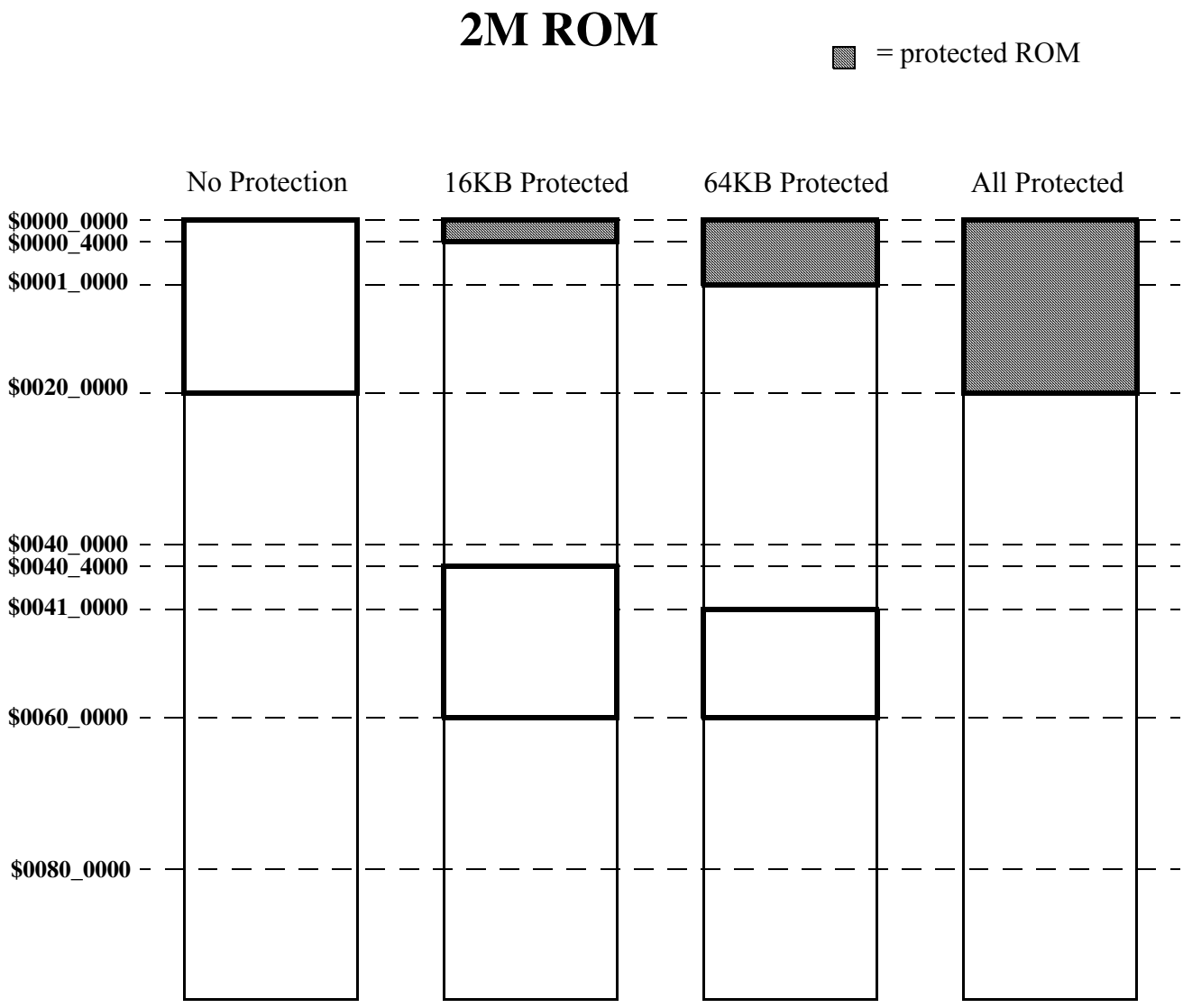


Figure 38-9. 2M ROM Protection Example in Internal Boot Mode (Normal)

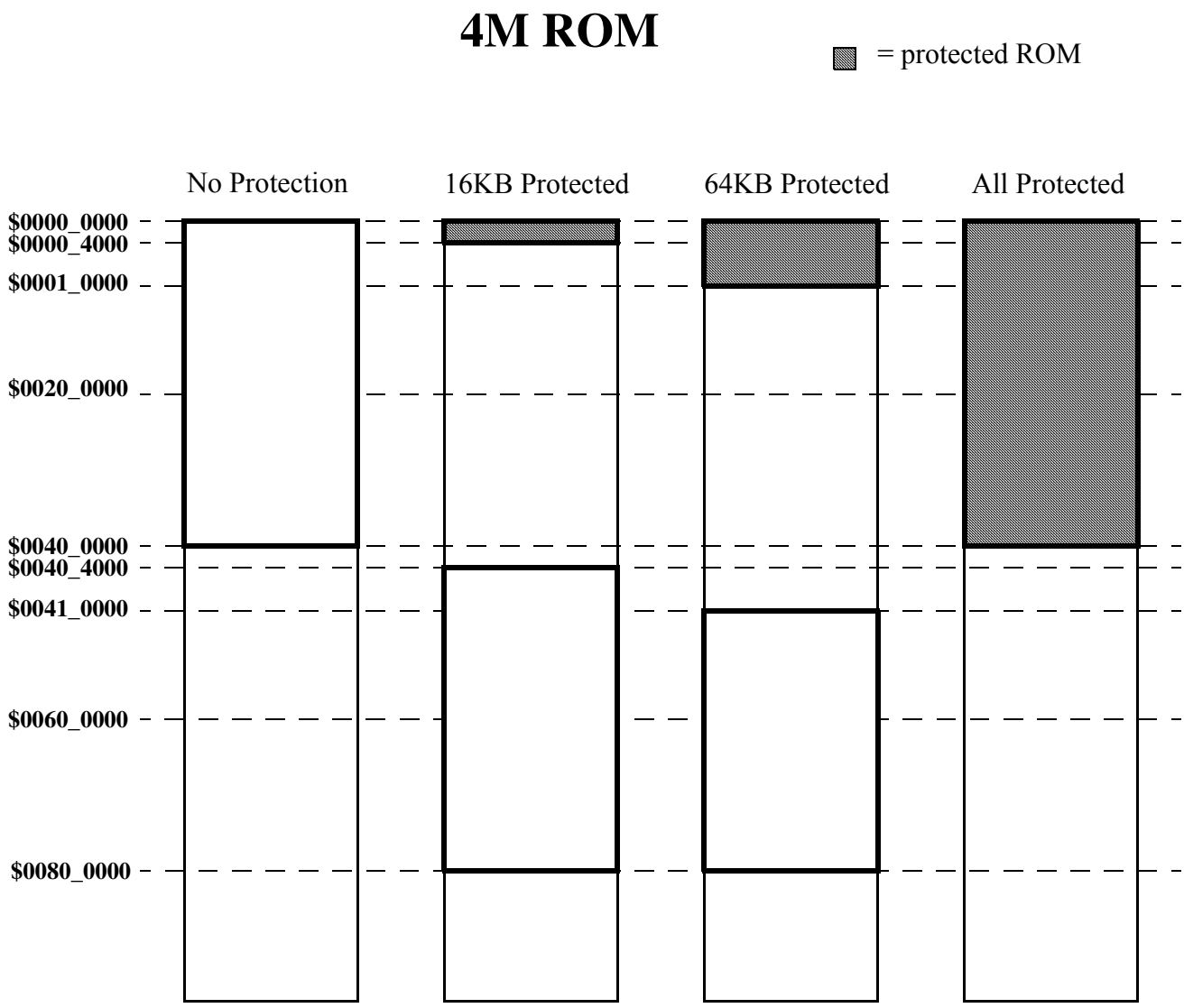


Figure 38-10. 4M ROM Protection Example in Internal Boot Mode (Normal)

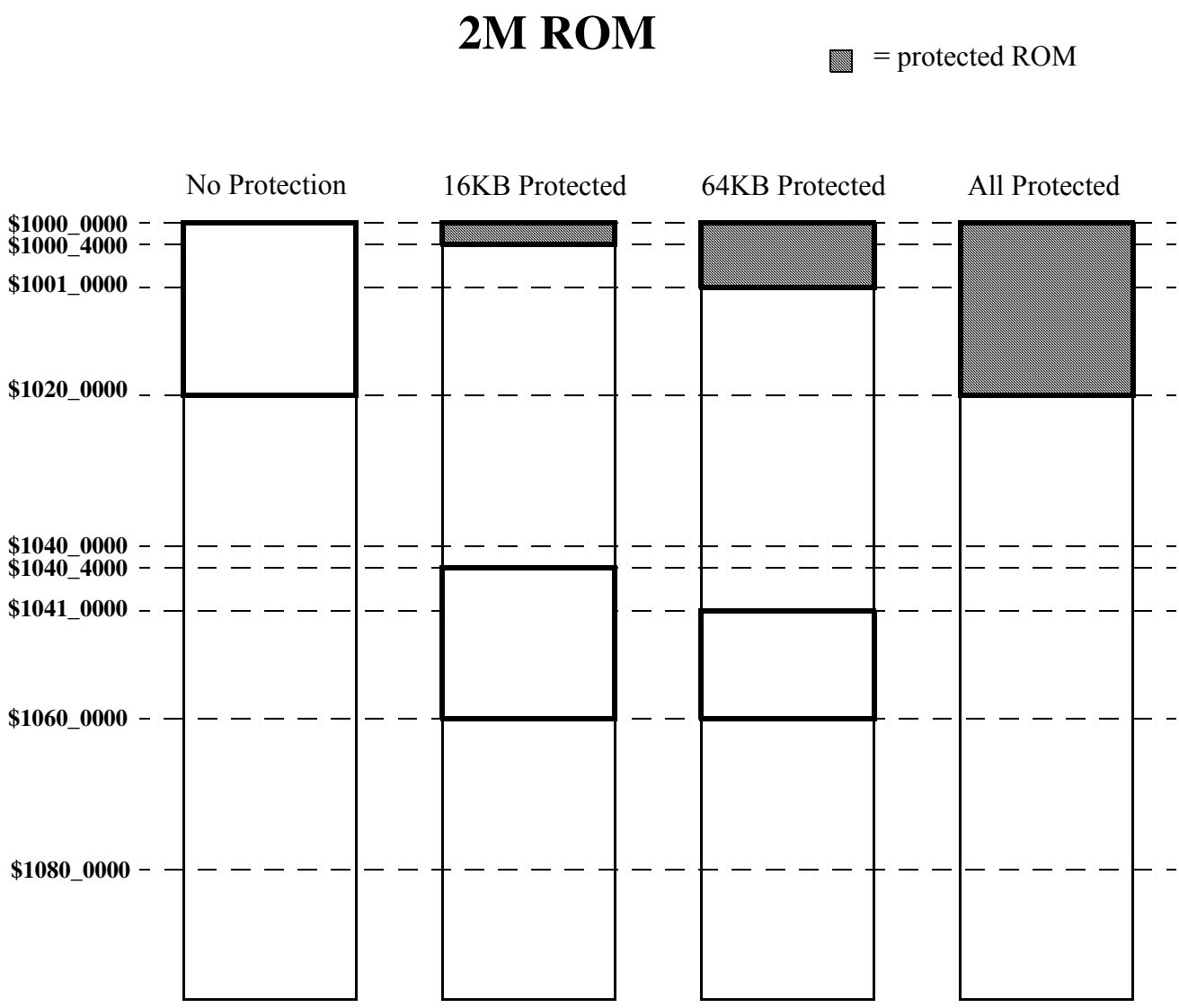


Figure 38-11. 2M ROM Protection Example in External Boot Mode

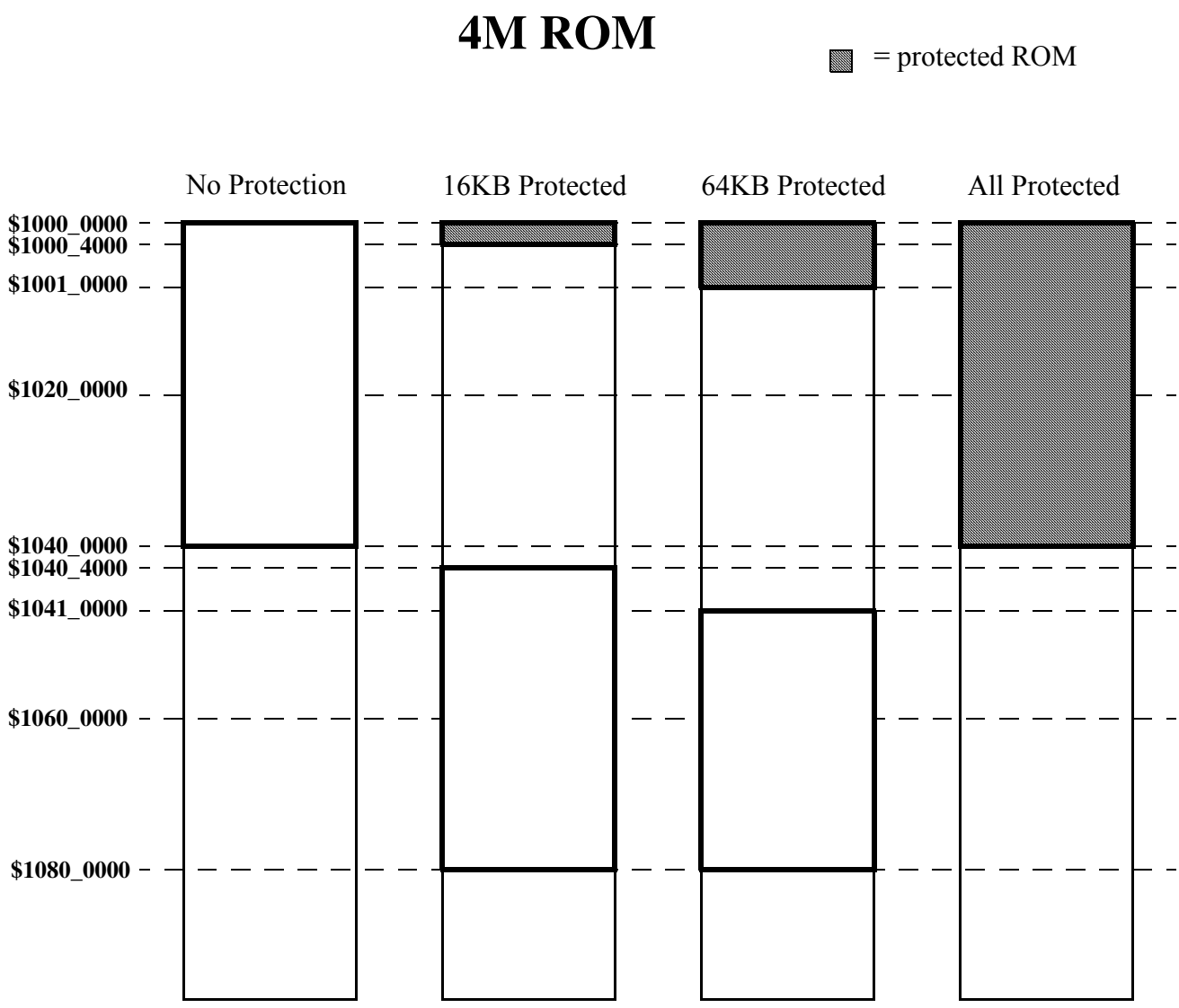


Figure 38-12. 4M ROM Protection Example in External Boot Mode

38.4.5.10 Alternate Bus Master Interface

The platform supports external bus masters by providing an alternate bus master interface to the platform's core arbitrator (CARB). Bus ownership may be requested/granted thereby allowing external bus masters access to the platform's internal R-AHB bus and it's resources.

38.4.5.10.1 R-AHB Bus Arbitration Signal List

The R-AHB alternate bus master signals list is shown in Table 38-1 . All signals function as explained in the AMBA Specification, Rev 2.0, Chapter 3 - AMBA AHB except where noted. **The signal "Type" is referenced from the platform perspective.** Similar AHB signal names are noted in parenthesis.

Table 38-9. R-AHB Bus Arbitration SIgnal List

Signal	Type	Description
HCLK	Input	R-AHB reference clock
HRESET_B	Input	R-AHB reset
HBUSREQ	Input	R-AHB Bus request
HGRANT	Output	R-AHB Bus grant
ALT_TRANS[1:0]	Input	Alternate Bus Master Transfer Type (HTRANS)
ALT_PROT[1:0]	Input	Alternate Bus Master Protection Control (HPROT)
ALT_ADDR[31:0]	Input	Alternate Bus Master Address (HADDR) (See Section 38.4.3.2 on page 38-8)
ALT_WDATA[31:0]	Input	Alternate Bus Master Write Data (HWDATA)
ALT_WRITE	Input	Alternate Bus Master Write/Read Indicator (HWRITE)
ALT_SIZE[1:0]	Input	Alternate Bus Master Transfer Size Indicator (HSIZE)
HRESP0	Output	Slave Response, 0 - OK, 1 - Error
HREADY	Output	Slave Data Ready Indicator (HREADY)
ALT_RDATA[31:0]	Output	Slave Read Data (HRDATA)
ALT_MASTER[2:0]	Input	Which Alternate Master Indicator

For alternate bus master interface timing, refer to Section 38.13.3 on page 38-52.

38.4.5.11 EIM Bus

The EIM Bus interfaces the R-AHB to external memory devices. The EIM also contains the R-AHB Bus watchdog timer. See the Chapter 48, “ARM External Interface Module (AEIM),” for more detail.

38.4.5.12 IP Bus

The AIPI module’s IP Bus interface enables R-AHB bus masters to access external IP Bus peripherals. See the IP Bus Rev 2.0 Specification for more detail.

38.4.5.13 MDPI Buses

The MDPI module Bus interface enables R-AHB bus masters to access dual port memory peripheral interface bus. See the MDPI Specification for more detail

38.4.5.14 Shared Memory Bus

The MEM_CTL module interfaces the R-AHB Bus to external shared memory via the Shared Memory Interface. See the Neptune MEM_CTL Specification for more detail.

38.5 Endian Configuration of ARM7TDMI

The endian configuration of the ARM is controlled by the CFGBIGEND input signal. This configures the byte and halfword extraction logic in the processor to extract the addressed data appropriately when accessing sub-word quantities. Tying this input HIGH causes the ARM to operate in big-endian mode. Tying it LOW causes the ARM to operate in little-endian mode. ARM registers are little-endian.

Note: Neptune's arm7_platform will operate in **big-endian** mode only. Little Endian information is presented for completeness only.

Once the system's endianness has been configured, any application code built to run on that system must be built to match. To do this, use the compiler/assembler options -li (for a little endian application - the default) or -bi (for a big-endian application) as appropriate. Library extensions must also be of the same endianness. Note that the ARM Software Development toolkit does not support dynamic endian configuration.

38.5.1 Big Endian Operation

A big-endian configured ARM (CFGBIGEND tied HIGH) should be connected to memory as follows:

- Byte 0 of the memory connected to D[31:24]
- Byte 1 of the memory connected to D[23:16]
- Byte 2 of the memory connected to D[15:8]
- Byte 3 of the memory connected to D[7:0]

The byte write enables are decoded as follows:

Table 38-10. Big Endian Byte Write Enable Decoding

WRITE	SIZE[1:0]	A[1:0]	WE[31:24]	WE[23:16]	WE[15:8]	WE[7:0]
0	?	?	0	0	0	0
1	00	00	1	0	0	0
1	00	01	0	1	0	0
1	00	10	0	0	1	0
1	00	11	0	0	0	1
1	01	0?	1	1	0	0
1	01	1?	0	0	1	1
1	10	?	1	1	1	1
1	11	?	Reserved			

NOTE:

During halfword and byte accesses, the address bits marked '?' should be ignored.

For more information, see ARM Application Note 61, Big and Little Endian Byte Addressing. This document can be downloaded from the ARM website (www.arm.com).

38.5.2 Little Endian Operation

A little-endian configured ARM (CFGBIGEND tied LOW) should be connected to memory as follows:

- Byte 0 of the memory connected to D[7:0]
- Byte 1 of the memory connected to D[15:8]
- Byte 2 of the memory connected to D[23:16]
- Byte 3 of the memory connected to D[31:24]

The byte write enables are decoded as follows:

Table 38-11. Little Endian Byte Write Enable Decoding

WRITE	SIZE[1:0]	A[1:0]	WE[31:24]	WE[23:16]	WE[15:8]	WE[7:0]
0	?	?	0	0	0	0
1	00	00	0	0	0	1
1	00	01	0	0	1	0
1	00	10	0	1	0	0
1	00	11	1	0	0	0
1	01	0?	0	0	1	1
1	01	1?	1	1	0	0
1	10	?	1	1	1	1
1	11	?	Reserved			

NOTE:

During halfword and byte accesses, the address bits marked '?' should be ignored.

38.6 JTAG

The ARM7TDMI-S processor contains an internal JTAG compatible tap-controller. The JTAG interface and functionality is documented in the ARM7TDMI-S Technical Reference Manual and will not be duplicated here. Refer to that document for a more complete discussion. In addition, the Neptune device will have a system-level JTAG controller (SJC) to coordinate top-level JTAG functionality as well as communication between the ARM7 and DSP JTAG controllers. Refer to the Chapter 71, “System JTAG Controller (SJC),” for more information.

However, several JTAG and debug related issues deserve special consideration. These topics are discussed below.

38.6.1 JTAG Clocking and Synchronization

The ARM7TDMI-S processor has a single clock (**clk**) input. There is no JTAG clock (TCK) input to ARM7TDMI-S. The **clk** must be running during JTAG operation (external TCK). An external synchronization circuit is necessary for the JTAG interface (see ARM7TDMI-S Technical Reference Manual, page 5-3) since the ARM core itself has a single clock domain, CLK. The **dbgtcken** input is synchronized to **clk** and controls a mux on all JTAG based registers. For instance, the tap-controller inside the ARM7 core will change state on a rising edge of **clk** when **dbgtcken** is high.

The synchronization circuit will reside in Neptune’s top-level System JTAG Controller(SJC). All flip-flops in the synchronization circuit will be reset by (**por** || ~**trst_b**).

The “Return TCK” signal (RTCK) shown in the synchronization circuit will be pinned out and available on the Neptune JTAG port but it’s use may or may not be supported by all development systems. For those development systems that do not support RTCK use, it is therefore a requirement that:

- **tck** freq <= 1/4 **clk** freq.

This clock frequency relationship insures proper synchronization circuit functionality.

38.6.2 The ARM7 DBGTRST_B Input

The reset input, **dbgtrst_b**, will be driven as follows:

dbgtrst_b = ~(**por** || ~**trst_b**);

where **por** is the Neptune power-on-reset signal and **trst_b** is Neptune’s JTAG reset. The JTAG related registers within the ARM7 must be accessible during system reset assertion (**reset_b** LOW). That is, the two clock domain’s resets are kept separate. This allows programming of internal ARM7 breakpoints during system reset and subsequent debugging through the boot process. **trst_b** may be asynchronously asserted but must be synchronously deasserted (**clk**).

38.6.3 The ARM7 DBGTDO Output

dbgtdo changes on the rising edge of **clk** when **dbgen** is asserted. This is differs from the JTAG specification which states that TDO should change on the falling edge of TCK (an input clock which the ARM7TDMI-S does not have). Logic in the SJC module, as well as the interconnection of the ARM7 JTAG controller with the ONYX JTAG controller will insure that this is not a problem. Refer to the Chapter 71, “System JTAG Controller (SJC),” for more information.

38.7 Debug Mode Considerations

38.7.1 The ARM7 Debug Enable (DBGEN)

The **dbgen** input must be asserted HIGH before the ARM7 processor will acknowledge debug and breakpoint requests. **dbgen** will be synchronized to **hclk**.

38.7.2 The ARM7 Debug Request (DBGRQ)

The **dbgrq** input on the ARM7 processor is internally synchronized and requests the processor to enter the debug state. This input port on the ARM7 will be driven by the AHBMUX module and should not be confused with the **dbgrq** port on the top-level of the arm7_platform. **dbgrq** is enabled in the ARM7 processor only when the **dbgen** input is asserted.

38.7.3 MCU Debug Acknowledge

There are two arm7_platform top-level signals relating to ARM7 processor debug acknowledge.

38.7.3.1 DBGACK

dbgack is the ARM7 debug acknowledge. This signal will assert HIGH when the processor has entered the debug state and **dbgen** is HIGH. When a system speed access from debug state occurs, the ARM7 processor temporarily drops out of debug state, and so **dbgack** can go LOW.

38.7.3.2 MCUACK_B

Neptune's top-level MCU_DE_B pin is an open-drain bi-directional I/O cell. As in input, it functions as a debug request to the ARM7 processor (synchronized in the SJC). As an output, it is driven low for 3 CLK's as a debug acknowledge. The **mcuack_b** output of the arm7_platform is a low-active signal which should be connected to the output enable of the MCU_DE_B output driver, whose data_in should be tied LOW.

38.7.4 Breakpoints (DBGBREAK)

The ARM7 processor supports two breakpoints in the internal ICE logic. Two additional breakpoints are provided in the platform by the watchpoint module (AWPT). The AWPT module will drive the **dbgbreak** input of the ARM7 processor. When HIGH, this signal causes the current access to be breakpointed. If the access was an instruction fetch, the processor will enter the debug state if the instruction reaches the execute stage of the pipeline. When the access is for data, the ARM7 will enter debug mode after the current instruction completes execution.

NOTE:

The registers which support ARM7 breakpoints within the AWPT module are NOT accessible via JTAG. Programming these registers requires bus mastership of the AHB by the ARM7 processor or an alternate bus-master (perhaps the TCM). If the development tools need to coordinate programming of these registers, they should first force the ARM7 into debug mode, and access the AWPT registers via single-step routines.

38.7.5 Debug Mode Behavior

When the ARM7TDMI-S enters debug mode, **dbgack** is asserted HIGH and **trans[1:0]**=2'b00 indicating an internal operation or IDLE cycle. The ARM7 specification requires that all internal cycles be terminated with a zero wait-state OKAY response. The AHBMUX module will issue zero-wait state **hready**'s for both IDLE and BUSY cycles which reach the R-AHB bus. If the alternate bus master owns the R-AHB bus, the CARB module will supply the ARM7 processor with active **clken**'s underneath the alternate bus master cycles for ARM7 IDLE and BUSY transactions.

When a debug request for a load or store has been made, the **trans[1:0]** signals will indicate a memory access and the transaction will proceed normally. The external development tool will poll the debug status register and determine if the transaction has completed and if the ARM7 processor has returned to debug mode or continued to execute normally.

The debug acknowledge signal **dbgack** is buffered and driven out to both the IPI bus. Peripherals on these buses monitor this signal and take appropriate debug mode action.

38.7.5.1 Alternate Bus Master Considerations

Alternate bus master accesses are allowed during debug mode. In the case where an alternate bus master has ownership of the R-AHB bus, the CARB is required to assert a zero wait-state **clken** to the ARM7 processor when IDLE (Internal) transactions are requested (mov r0,r0 for instance), and for BUSY cycles as well. The CARB module is also responsible for driving **abort** negated for these IDLE and BUSY transactions. When the ARM7 processor re-establishes R-AHB ownership, the AHBMUX module will resume the role of IDLE and BUSY cycle **hready** generation and the R-AHB bus **resp0** signal will be routed by the CARB to the ARM7 **abort** input.

When a debug request for a load or store has been made, the **trans[1:0]** signals will indicate a memory access and the transaction will proceed normally. However, if an alternate bus master has the bus, the debug generated transaction will not complete until the ARM7 processor regains ownership of the R-AHB bus. However, this is not a problem as the external development tool will poll the debug status reqister and determine if the transaction has completed. If the status register **trans[1]** bit is high, it means the transaction has not completed. If the status register **trans[1]** bit is low (indicating an IDLE cycle) it means the processor has returned to debug mode.

38.8 Low Power Modes

38.8.1 CCM_LPMODE[1:0]

The clock control module drives the arm7_platform's CCM_LPMODE inputs. These inputs are routed to the CARB, EIM, and AIPI modules. The CARB will use these inputs to gate the CLK input to the ARM7 processor. The peripheral interfaces will buffer these signals and provide them to the external devices.

Table 38-12. CCM_LPMODE Encoding

CCM_LPMODE[1:0]	Low Power Mode
00	STOP
01	WAIT
10	DOZE
11	normal

38.8.2 AP_WAKEUP_B

The **ap_wakeup_b** output signal is connected to the CCM module. This signal is asserted when an interrupt is pending or if a debug request has been made. The interrupts monitored are non-synchronized (**firq_b** and **irq_b**). The debug request monitored is the external input **dbgrq** as opposed to internally generated debug request. The CCM will respond by enabling the HCLK to the ARM7 platform. This will allow subsequent interrupt handling and debug functionality. **ap_wakeup_b** is an asynchronous output signal.

38.9 AHBMUX Module

The AHBMUX module basically connects masters and slaves of the R-AHB bus together. Individual circuits within the module are discussed below.

38.9.1 Alternate Bus Master Interface to R-AHB

The R-AHB bus can be mastered by either the ARM7 processor or by an external bus master via the alternate bus master interface. See Section 38.3.2 on the CARB module and Section 38.4.5.10 for a description of the alternate bus master interface. Figure 38-13 below shows a simple functional diagram of the R-AHB bus master muxing which exists in the AHBMUX module.

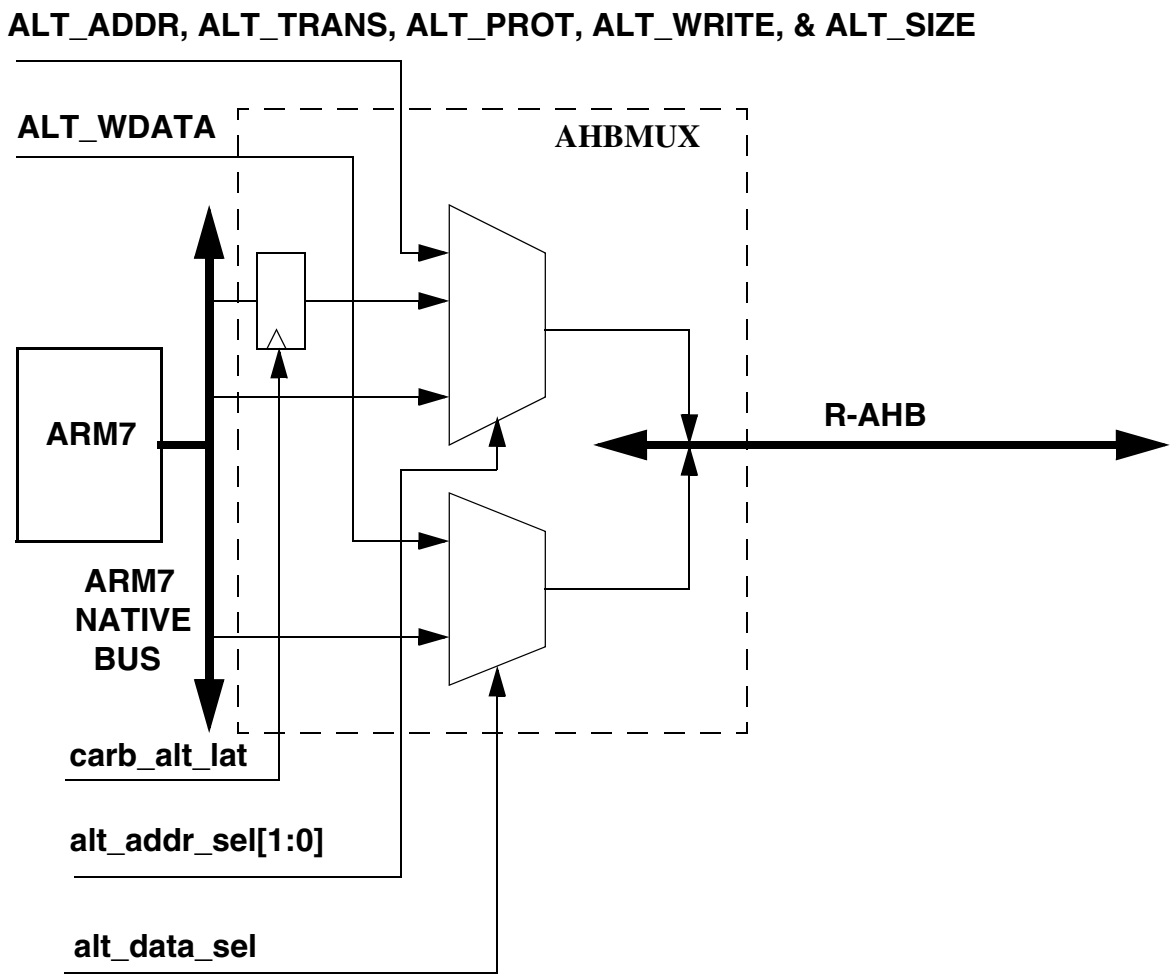


Figure 38-13. R-AHB Bus Master Mux

Control signals from the CARB module select which mux inputs will drive the R-AHB address, attributes, and write data. The **carb_alt_lat** signal enables a set of flip-flops to capture the ARM7 native bus address and address attributes. It is necessary to capture this information so that it is not lost during bus-arbitration upon reacquiring bus ownership. See Chapter 40, “ARM7 Platform Core Arbiter (CARB),” for more detail. Table 38-13 shows the encoding of the **alt_addr_sel[1:0]** inputs.

Table 38-13. alt_addr_sel Encoding

ALT_ADDR_SEL[1:0]	Mux Input Selected
00	ARM7 Native Bus
01	Alternate Bus Master I/F
10	Native Bus Flip-Flops
11	Reserved

The **alt_data_sel** signal selects the ARM7 write data bus when low, and the ALT_WDATA bus when asserted HIGH.

38.9.2 R-AHB Centralized Address Decoding

The AHBMUX module contains centralized decoding of the R-AHB **haddr[31:0]** bus. The decode is implemented as specified in the Chapter 6 of the Neptune Specification, Chip Configuration and Memory Maps (Section 6.4 MCU Memory Map). Note memory aliasing and multi-mapping schemes.

38.9.2.1 Internal / External Boot Select

The **boot_int** input to the arm7_platform module is routed to the AHBMUX module. When this signal is HIGH the processor will boot from internal memory and **hsel_rom** will be assert for the reset vector fetch. When **boot_int** is tied LOW the processor will boot from external memory through the AEIM module and **hsel_cs[0]** will assert for the reset vector fetch.

38.9.2.2 Module Select Outputs (hsel_)

The following **hsel** output signals are connected to the appropriate slave device:

(Refer to Chapter 6, “Chip Configuration and Memory Maps (MEMMAP),” for detailed address mapping information.)

- hsel_rom
- hsel_ram
- hsel_mctl
- hsel_cs[5:0]
- hsel_aeim
- hsel_mdpi
- hsel_awpt
- hsel_aitc

The **hsel*** signals are combinational decodes of the **haddr** bus, and can change whenever the **haddr** bus changes. R-AHB bus slaves should sample the appropriate **hsel** signal on the rising edge on HCLK when **hready_in** is asserted.

38.9.3 Special R-AHB Decoding

Two additional hsel signals are sent to the memory controller module (MCTL) to assist in aborted accesses to unimplemented memory holes (**hsel_hole**) and protected memory space (**hsel_prot**). **hsel_rom** or **hsel_ram** will always be asserted whenever **hsel_hole** or **hsel_prot** is asserted.

38.9.3.1 Holes in the Memory Map (hsel_hole)

The **hsel_hole** output is a decode of all ROM "hole" (Refer to Section 6.3.2, "On-Chip MCU RAM."). The **hsel_hole** decode includes holes in the "implemented" portion of the ROM, as well as the holes in the aliased and multi-mapped regions. **hsel_hole** will not assert unless either **hsel_rom** is also asserted. Accesses to the these "holes" will be aborted by the memory controller.

38.9.3.1.1 ROM Holes

Allocated ROM space is 32 MB. The physical memory is 1792 kB. Physical and unimplemented ROM memory is aliased to fill the 32 MB space, starting at 2MB (32'h0020_0000).

Table 38-14 lists the regions of the R-AHB bus (**haddr[31:0]**) within mcu ROM space which are decoded as holes. The location of the ROM holes is dependent on the state of the **boot_int** input since ROM space is remapped to upper memory when booting externally.

Table 38-14. MCU ROM Decoding for hsel_hole

HADDR[31:0] (boot_int=0)	HADDR[31:0] (boot_int=1)
$101C_0000 - $101F_FFFF	$001C_0000 - $001F_FFFF
$103C_0000 - $103F_FFFF	$003C_0000 - $003F_FFFF
$105C_0000 - $105F_FFFF	$005C_0000 - $005F_FFFF
$107C_0000 - $107F_FFFF	$007C_0000 - $007F_FFFF
$109C_0000 - $109F_FFFF	$009C_0000 - $009F_FFFF
$10BC_0000 - $10BF_FFFF	$00BC_0000 - $00BF_FFFF
$10DC_0000 - $10DF_FFFF	$00DC_0000 - $00DF_FFFF
$10FC_0000 - $10FF_FFFF	$00FC_0000 - $00FF_FFFF
$111C_0000 - $111F_FFFF	$011C_0000 - $011F_FFFF
$113C_0000 - $113F_FFFF	$013C_0000 - $013F_FFFF
$115C_0000 - $115F_FFFF	$015C_0000 - $015F_FFFF
$117C_0000 - $117F_FFFF	$017C_0000 - $017F_FFFF

Table 38-14. MCU ROM Decoding for hsel_hole

HADDR[31:0] (boot_int=0)	HADDR[31:0] (boot_int=1)
$119C_0000 - $119F_FFFF	$019C_0000 - $019F_FFFF
$11BC_0000 - $11BF_FFFF	$01BC_0000 - $01BF_FFFF
$11DC_0000 - $11DF_FFFF	$01DC_0000 - $01DF_FFFF
$11FC_0000 - $11FF_FFFF	$01FC_0000 - $01FF_FFFF

38.9.3.1.2 RAM

The physical memory is 512Kx8 (7_FFFF bytes), which exists from $03F8_0000-$03FF_FFFF. RAM memory is also multi-mapped to upper memory (refer to the Chapter 6, “Chip Configuration and Memory Maps (MEMMAP).”)

38.9.3.2 HSEL_PROT

The **hsel_prot** output is a decode of the protected (supervisor mode access only) upper 4k of mcu RAM space. The **hsel_prot** decode includes the upper 4k of the implemented memory and mapped region of mcu RAM space. Non supervisor-mode accesses to these regions will be aborted by the memory controller. **Table 38-15** shows mapped regions of mcu RAM which are decoded as “protected” space.

Table 38-15. Decoding for hsel_prot

HADDR[31:0] (LOWER)	HADDR[31:0] (UPPER)
$03FF_FXXX	$FFFF_FXXX

38.9.4 R-AHB State Trackers

The AHBMUX module uses the **hsel*** decodes and the **hready** signal to keep track of which slave is the current target of a transaction. In the case of a read cycle, these state trackers control the selects on the R-AHB read data mux. See Section 38.9.5.

38.9.5 Read Data Muxing

The R-AHB read data from all seven slave devices

- aeim_rdata[31:0]
- mdpi_rdata[31:0]
- aipi_rdata[31:0]
- mem_rdata[31:0]
- awpt_rdata[31:0]
- aitc_rdata[31:0]

is sent to the AHBMUX module. Internally, the **hsel*** and bus tracking logic controls selection of which slave's read data is driven on to the R-AHB **hrdata[31:0]** lines.

38.9.5.1 Read Data Over-Rides

Both the AITC and AWPT modules have over-ride signals (**aitc_rdata_ovr** & **awpt_rdata_ovr**) which are inputs to the AHBMUX module. When either is asserted, they will over-ride the internal selection of read data with **aitc_rdata[31:0]** or **awpt_rdata[31:0]**. The AITC over-ride has priority over the AWPT over-ride.

38.9.6 HREADY

The seven slave devices in the arm7_platform all drive an **hready_out** signal to the AHBMUX module. They are OR'd together and sent to the AEIM module as **r_ahb_hready**. The AEIM is then responsible for driving the R-AHB **hready** signal. Note that the AEIM can delay the assertion of **hready** if SHOW cycles are enable. See Section 38.4.3.4 Delayed Cycle Termination.

38.9.7 RESP0 (Abort)

Six slave devices (all slaves except the AITC module) in the arm7_platfrom drive an **resp0** signal to the AHBMUX module. A HIGH value indicates a transaction Error. They are OR'd together in the AHBMUX and then driven onto the **hresp0** output. **hresp0** is subsequently driven to the CARB module. In the case of ARM7 R-AHB bus ownership, the CARB module will simply forward the **hresp0** signal onto the processor **abort** input. However, during alternate bus master ownership, the CARB will always drive the processor **abort** input negated.

38.9.8 HTRANS[0]

In certain cases, the AHBMUX will force a NONSEQ bus transaction by negating **htrans[0]**.

The first case is on accesses across 1k byte address boundaries. The AHBMUX will compare **haddr[9:2]** for all ones, and negate **htrans[0]** on the following address phase.

The second case is when the CARB module drives the **carb_force_nonseq** input to the AHBMUX. This signal will be asserted by the CARB to ensure the first transaction after a change of bus-ownership is a non-sequential transfer.

38.9.9 Reset, IDLE, and BUSY State HREADY Assertion

The AHBMUX will generated an asserted **r_ahb_ready** on the initial bus cycle after **hreset_b** is deasserted. The AHBMUX is also the default responder for all IDLE and BUSY cycles. This consolidates irregular cycle termination inside the AHBMUX.

38.9.10 AHBMUX_DBGRQ

The ARM7 platform's **dbgrq** input is synchronized by **hclk** in the AHBMUX module. The **ahbmux_dbgrq** output is then driven to the ARM7 processor's DBGRQ input. The unsynchronized **dbgrq** input is combined with **irq_b** and **firq_b** to generate the **ap_wakeup_b** output. In low-power mode (**hclk** turned off), a debug request via **dbgrq** will assert **ap_wakeup_b** which will cause **hclk** to turn on. Two clocks later, **ahbmux_dbgrq** will assert and stay asserted until the **dbgack** signal confirms that the ARM7 processor has indeed entered debug mode. The synchronization circuit is cleared by **dbgtrst_b**. The connections are shown in Figure 38-14

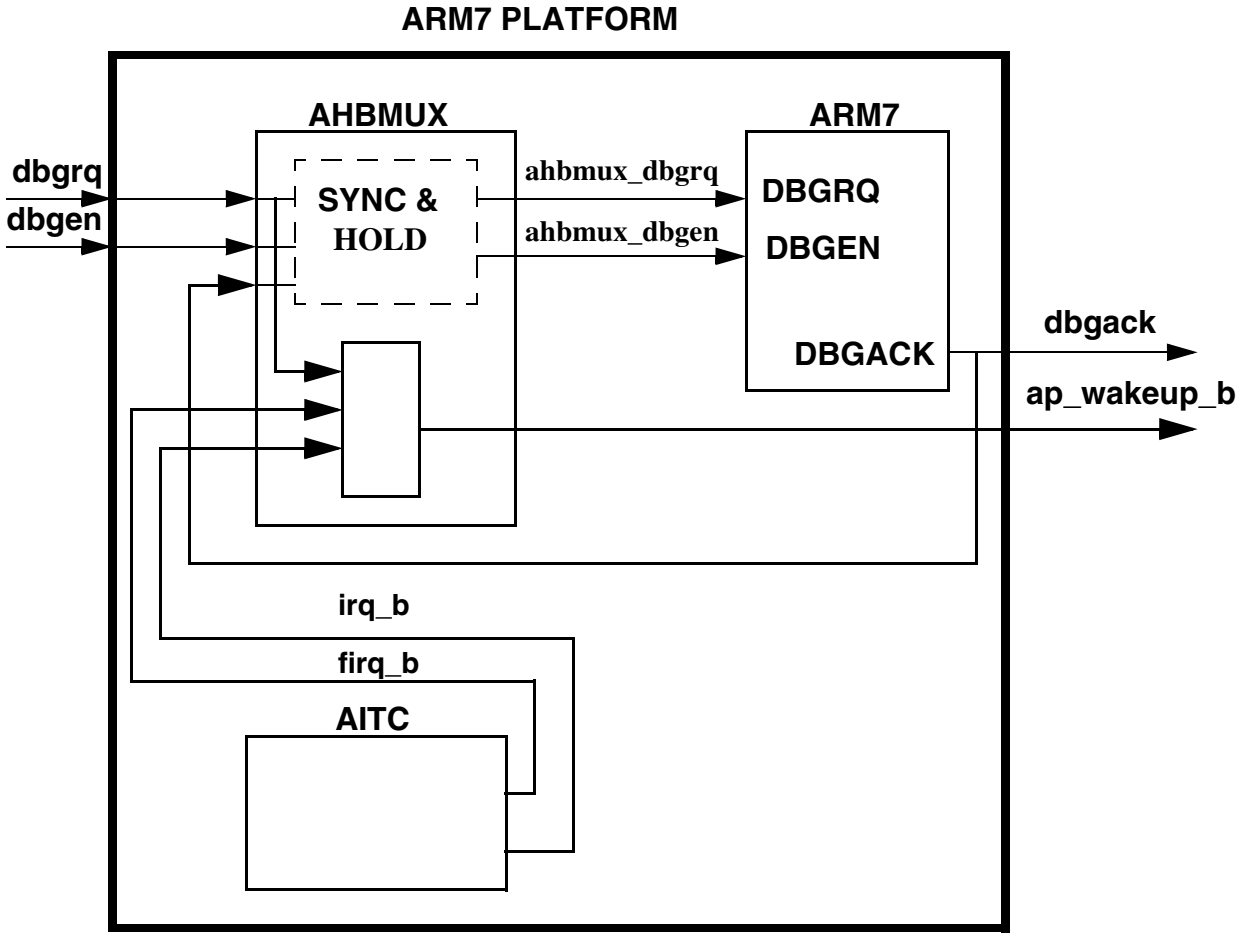


Figure 38-14. DBGRQ and DBGEN Interconnections

38.9.11 AHBMUX_DBGEN

The ARM7 processor's DBGEN input must be asserted in order for it to respond to a debug request. For this reason the ARM7 platform's **dbgen** input is OR'd with **ahbmux_dbgrq** and sent to the ARM7's DBGEN input as **ahbmux_dbgen.** Once **ahbmux_dbgen** asserts, it is held by a flop inside the AHBMUX module. **dbgtrst_b** must be asserted to negate it. The connections are shown in Figure 38-14

38.10 Memory Map

Chapter 6 of the Neptune Specification, Chip Configuration and Memory Maps, contains the MCU Memory Map in Section 6.4. This is the memory map for the arm7_platform's R-AHB bus. Refer to that specification for all arm7_platform address allocation questions.

38.11 Scan Wrapper and DFT

The arm7_platform will be designed with a scannable wrapper through which almost all input and output signals will traverse. The wrapper will enable a very high degree of testability to a very deeply embedded arm7_platform. Path test capable inputs and outputs will be driven by a two-stage stimulus flip-flops and captured by two-stage monitor flip-flops, respectively. This will allow at-speed test capability as well as fault based test coverage. Details of the scan wrapper and other DFT related issues will be covered in the arm7_platform DFT document TBD.

38.12 ARM7 Platform Pin List

The complete list of inputs and outputs for the ARM7 platform are listed in Table 38-16.

Table 38-16. ARM Platform Pin List

Signal	Type	Description
hclk	Input	AHB reference clock
h_reset_b	Input	AHB reset
fuse_reset_b	Input	Memory Reset Pulse
boot_int	Input	Internal or external boot select
DEBUG RELATED SIGNALS		
dbgen	Input	ARM7 Debug Enable
dbgrq	Input	ARM7 Debug Request
dbgack	Output	ARM7 Debug Acknowledge
JTAG INTERFACE		
dbgtcken	Input	Test Clock (JTAG) Enable.
dbgtrst_b	Input	JTAG Test Reset
dbgtms	Input	JTAG Test Mode Select
dbgtdi	Input	JTAG Test Data Input
dbgtdo	Output	JTAG Test Data Output
dbgtdoen_b	Output	JTAG Test Data Output Tri-state Control
ALT BUS MASTER I/F		
hbusreq	Input	AHB Bus request
hgrant	Output	AHB Bus grant
alt_addr[31:0]	Input	Alternate Bus Master Address (HADDR)
alt_wdata[31:0]	Input	Alternate Bus Master Write Data (HWDATA)
alt_write	Input	Alternate Bus Master Write/Read Indicator (HWRITE)

Table 38-16. ARM Platform Pin List

Signal	Type	Description
alt_size[1:0]	Input	Alternate Bus Master Transfer Size Indicator (HSIZE)
alt_trans[1:0]	Input	Alternate Bus Master Transfer Type (HTRANS)
alt_prot[1:0]	Input	Alternate Bus Master Protection Control (HPROT)
alt_master[2:0]	Input	Alternate Bus Master Owner Indicator
hresp0	Output	Slave Response
hready	Output	Slave Data Ready Indicator
alt_rdata[31:0]	Output	Slave Read Data (HRDATA)
MEMORY INTERFACE		
ramsiz[19:10]	Input	address switch select RAM size
romsiz[21:11]	Input	address switch select ROM size
romprot[1:0]	Input	ROM size protection select
ipt_test_membist_invoke	Input	
ipt_test_membist_reset	Input	
ipt_test_membist_mrom_sel	Input	
ipt_test_membist_mram_sel	Input	
ipt_test_membist_sel	Output	
ipt_test_membist_fail	Output	
ipt_test_membist_done	Output	
ipt_test_membist_bitmap_data_out	Output	
mctl_ram_reset_b	Output	Reset to RAM
mctl_rom_reset_b	Output	Reset to ROM
tcm_mem_mode	Input	Test Mode Select

Table 38-16. ARM Platform Pin List

Signal	Type	Description
mem_q_ram[31:0]	Input	MCU RAM Data Out
mem_dis_ram	Input	MCU RAM DisBit
mem_fuse_ram[112:0]	Input	MCU RAM Fuse Lines
mem_q_rom[31:0]	Input	ROM Data OUT
mctl_d_ram[31:0]	Output	MCU RAM Data In
mctl_addr_ram[19:0]	Output	MCU RAM Address
mctl_asel_ram_b	Output	MCU RAM Address Strobe
mctl_dsel_ram_b	Output	MCU RAM Data Out Strobe
mctl_rw_ram_b	Output	MCU RAM Read not Write
mctl_ben_ram[3:0]	Output	MCU RAM Byte Enable
mctl_wval_ram[3:0]	Output	MCU Write TIming
mctl_rval_ram[3:0]	Output	MCU RAM Read Timing
mctl_fusetest_ram	Output	MCU RAM Fuse Test Mode
mctl_fsense_ram	Output	109K Ram Fuse Sense
mctl_testmode_ram	Output	MCU RAM Test Mode
mctl_prechg_in	Output	MCU RAM GDL Precharge Mode
mctl_tscan_en_ram	Output	MCU RAM DisBit Scan Enable
mctl_t_clk_ram	Output	MCU RAM DisBit Scan Clock
mctl_addr_rom[20:0]	Output	MCU ROM Address
mctl_asel_rom_b	Output	MCU ROM Address Strobe
mctl_ben_rom[3:0]	Output	MCU ROM Byte Enable
mctl_rval_rom[3:0]	Output	MCU ROM Read Timing

Table 38-16. ARM Platform Pin List

Signal	Type	Description
mctl_rval2_rom[3:0]	Output	MCU ROM Read Timing 2
AEIM INTERFACE		
aeim_data_out[31:0]	Output	AEIM Output Data Bus
aeim_addr_out[31:0]	Output	AEIM Output Address Bus
aeim_gpo[15:0]	Output	AEIM General Purpose Outputs
aeim_cs_b[4:0]	Output	AEIM Chip Selects
aeim_eb_b[3:0]	Output	AEIM Byte Enables
aeim_wr_oe[3:0]	Output	AEIM Write Output Enable
aeim_qstat[3:0]	Output	AEIM Status Signals (for Debug)
aeim_cs5	Output	AEIM Chip Select 5
aeim_siz[1:0]	Output	AEIM Debug Version of AHB SIZE[1:0]
aeim_oe_b	Output	AEIM Output Enable
aeim_bclk_out	Output	AEIM Burst Clock Output
aeim_lba_b	Output	AEIM Load Burst Address
aeim_rw_b	Output	AEIM Read/Write_B
aeim_strobe	Output	AEIM Show Strobe
aeim_data_in[31:0]	Input	AEIM Input Data Bus
aeim_ecb_b	Input	AEIM End Current Burst
aeim_bclk_in	Input	AEIM bclk feedback from pin
aeim_show_rw_b	Output	AEIM Read/Write Indicator for Show Cycles
aeim_cs0_dsz[2:0]	Input	CSO Data Port Size and Alignment
aeim_dbg_cstate	Output	AEIM Debug Output of Current State

Table 38-16. ARM Platform Pin List

Signal	Type	Description
aeim_dbg_bclk_cstate	Output	AEIM Debug Output of BCLK Current State
aeim_dbg_edead	Output	AEIM Debug Output of EDEAD
aeim_dbg_latch_data	Output	AEIM Debug Output of Latch Data
aeim_dbg_insert_wait	Output	AEIM Debug Output of insert_wait
aeim_dbg_pre_burst	Output	AEIM Debug Output of pre_burst
aeim_dbg_lat_extmem	Output	AEIM Debug Output of lat_extMemAccess
MDPI INTERFACE		
dpr_xdb_rd[15:0]	input	Data port from X-DPR
dpr_ydb_rd15:0]	input	Data port from Y-DPR
dpr_xab[9:0]	output	X-RAM address bus
dpr_yab[9:0]	output	Y-RAM address bus
dpr_xdb_w[15:0]	output	X-RAM data write bus
dpr_ydb_wr[15:0]	output	Y-RAM data write bus
dpr_arxwr	output	X-RAM write control
dpr_arywr	output	Y-RAM write control
dpr_arxrd	output	X-RAM read control
dpr_aryrd	output	Y-RAM read control
dpr_abort	output	X/Y-RAM abort
IP BUS INTERFACE		
ips_rdata[31:0]	Input	IP Bus Read Data
ips_xfr_wait	Input	IP Bus Transfer Wait State Indictor
ips_xfr_err	Input	IP Bus Illegal Access Indicator

Table 38-16. ARM Platform Pin List

Signal	Type	Description
ips_wdata[31:0]	Output	IPBus Write Data
ips_module_en[31:0]	Output	IP Bus Module Select
ips_addr[11:2]	Output	IP Bus Address
ips_byte_31_24	Output	IP Bus Byte Select
ips_byte_23_16	Output	IP Bus Byte Select
ips_byte_15_8	Output	IP Bus Byte Select
ips_byte_7_0	Output	IP Bus Byte Select
ips_rwb	Output	IP Bus Read/Write_b Indicator
ips_supervisor_access	Output	IP Bus Supervisor Mode Access Indicator
ipg_doze	Output	IP Bus Doze Mode Indicator
ipg_stop	Output	IP Bus Stop Mode Indicator
ipg_wait	Output	IP Bus Wait Mode Indicator
ipg_debug	Output	IP Bus Debug Mode Indicator
MISCELLANEOUS		
ap_int_b[63:0]	Input	External Interrupts
ap_wakeup_b	Output	Wakeup - Turn the CLK On
mcu_ack_b	Output	Output Enable for MCU_DE_B Open-Drain I/O Cell
dsm_int_holdoff	Input	Deep Sleep Module Interrupt Disable
ccm_lpmode[1:0]	Input	Lower Power Mode
TEST INTERFACE		
test_reset	Input	Neptune Specific Test Reset
tcm_scan_mode	Input	Neptune Specific TCM Scan Mode

Table 38-16. ARM Platform Pin List

Signal	Type	Description
tcm_peripheral_scan	Input	Neptune Specific Peripheral Scan Mode
scan_mod_sdi_mcu_bypass[26:0]	Input	Neptune Specific (wrap & core scan)
tcm_scan_divergence	Input	Neptune Specific Scan Divergence
scan_mode_div_in	Input	Neptune Specific Scan Divergence Scan-In
ipt_test_mode[3:0]	Input	Test Mode Control Bits
ipt_test_scan_enable	Input	Scan Test Mode Enable
ipt_test_wrapper_se	Input	Wrapper Scan Test Mode Enable
ipt_test_clk_se	Input	Scan Shift Enable for Clock Gating
ipt_test_wrapper_clk_in	Input	Test Wrapper Clock
ipt_test_safemode_en	Input	Platform Safemode Enable
ipt_test_wrapper_scan_in[7:0]	Input	Scan Wrapper Serial Inputs
ipt_test_wrapper_scan_out[7:0]	Output	Scan Wrapper Serial Outputs
ipt_test_scan_in[44:0]	Input	Scan Test Serial Inputs
ipt_test_scan_out[44:0]	Output	Scan Test Serial Outputs
ipt_test_scan_size[1:0]	Input	Scan Path Partitioning Control Bits
ipt_test_wrapper_scan_size[1:0]	Input	Scan Wrapper Path Partitioning Control

38.13 Electrical Specifications

38.13.1 External Timing References

The timing specifications for the AEIM, MDPI, AIPI and MCTL modules will be covered in the Neptune AEIM, MDPI, AIPI, and MCTL module specifications, respectively.

38.13.2 R-AHB Timing

The timing parameters for an access to an R-AHB bus slave device operating in the arm7_platform are shown on the following pages. The constraints used to generate the preliminary numbers are shown in **Table 38-17**. Figure 38-15 shows the HCLK insertion delay and the R-AHB slave reset timing parameters.Figure 38-16 on the following page shows the main R-AHB timing parameters. Finally, **Table 38-18** list the AC parameters for the arm7_platform R-AHB bus.

Table 38-17. R-AHB Timing Constraints

HCLK_ROOT Period	18 ns (55.55 MHz)
HCLK_ROOT to HCLK Insertion Delay (T_{insert})	1.8 ns
HCLK Uncertainty	350 ps
HADDR, HRDATA, HWDATA Loading	1.5 pF
All Other Output Loading	750 fF
Input Delay (Excluding HRDATA)	11.9 ns (0.7*HCLK)

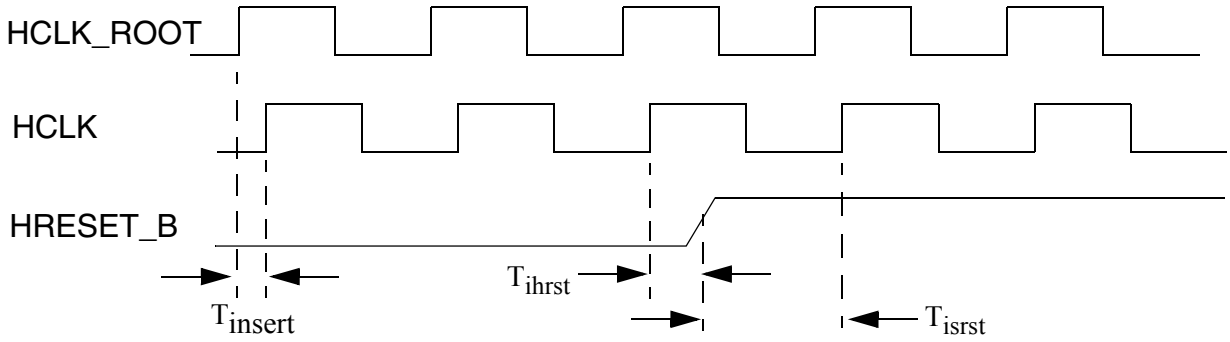


Figure 38-15. R-AHB Slave Reset Timing

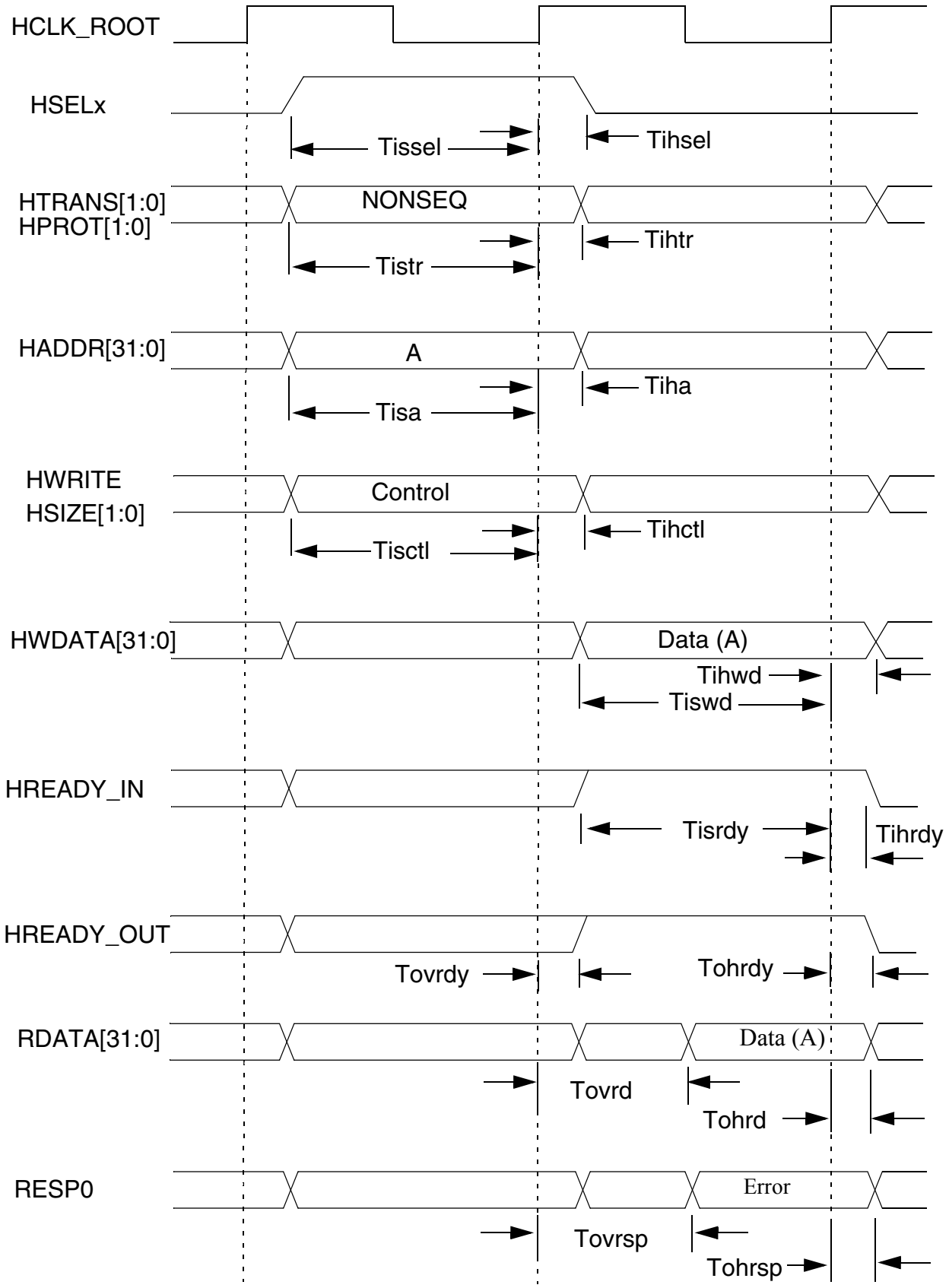


Figure 38-16. R-AHB Bus Timing Parameters

NOTE:

The Timing parameters in Table 38-18 are measured with respect to HCLK_ROOT.

Table 38-18. R-AHB AC Timing Parameters

Description	Parameter	Timing (ns)
HCLK_ROOT minimum clock period	T_{clk}	18
HRESET_B deasserted setup time before HCLK_ROOT	T_{isrst}	11.2
HRESET_B deasserted hold time after HCLK_ROOT	T_{ihrst}	1.7
HSELx setup time before HCLK_ROOT HSEL_ROM setup time before HCLK_ROOT HSEL_CS setup time before HCLK_ROOT	T_{issel}	3.7 4.0 4.7 4.5
HSELx hold time after HCLK_ROOT	T_{ihsel}	>0
HSEL_HOLE and HSEL_PROT setup time before HCLK_ROOT	T_{ishole}	2.5
HSEL_HOLE and HSEL_PROT hold time after HCLK_ROOT	T_{ihhole}	>0
HTRANS / HPROT Transfer Type setup time before HCLK_ROOT	T_{istr}	5.2
HTRANS / HPROT Transfer Type hold time after HCLK_ROOT	T_{ihtr}	>0
HADDR[31:0] setup time before HCLK_ROOT	T_{isa}	5.2
HADDR[31:0] hold time after HCLK_ROOT	T_{iha}	>0
HWRITE and HSIZE[1:0] control signal setup time before HCLK_ROOT	T_{isctl}	5.2
HWRITE and HSIZE[1:0] control signal hold time after HCLK_ROOT	T_{ihctl}	>0
HWDATA Write Data setup time before HCLK_ROOT	T_{iswd}	5.2
HWDATA Write Data hold time after HCLK_ROOT	T_{ihwd}	>0
HREADY_IN Ready In setup time before HCLK_ROOT	T_{isrdyi}	4.0
HREADY_IN Ready In hold time after HCLK_ROOT	T_{ihrdyi}	>0

NOTE:

The Timing parameters in Table 38-18 are measured with respect to HCLK_ROOT.

Table 38-18. R-AHB AC Timing Parameters

Description	Parameter	Timing (ns)
HREADY_OUT Ready Out valid time after HCLK_ROOT	T_{ovrdy}	5.2
HREADY_OUT Ready Out hold time after HCLK_ROOT	T_{ohrdy}	>0
RDATA Read Data valid time after HCLK_ROOT (input to AHBMUX) (See Note at end of Table)	T_{ovrd}	memory = 16.3 aeim = 15.3 slaves = 14.3
RDATA Read Data hold time after HCLK_ROOT (input to AHBMUX)	T_{ohrd}	>0
RESP0 valid time after HCLK_ROOT (input to AHBMUX)	T_{ovrsp}	5.2
RESP0 hold time after HCLK_ROOT (input to AHBMUX)	T_{ohrsp}	>0

NOTE:

Timing on memory RDATA is set more stringent than all other R-AHB slave devices. This enables constraints for the read data mux in the AHBMUX module to prioritize the memory to processor read data path.

38.13.3 Alternate Bus Master Interface Timing

The timing parameters for an access via the alternate bus master interface are presented below. The constraints used to generate the preliminary numbers are shown in Table 38-19. Figure 38-17 on the following page shows timing parameters for all of the signals residing on the Alternate Bus Master Interface. Finally, Table 38-20 lists the AC parameters for the Alternate Bus Master interface to the R-AHB bus.

Table 38-19. Alternate Bus Master Interface Constraints

HCLK_ROOT Period	18 ns (55.55 MHz)
HCLK_ROOT to HCLK Insertion Delay (T_{insert})	1.8 ns
HCLK Uncertainty	350 ps
Input Loading (Presented at the Platform Boundary)	325 fF
Output Loading	1 pF

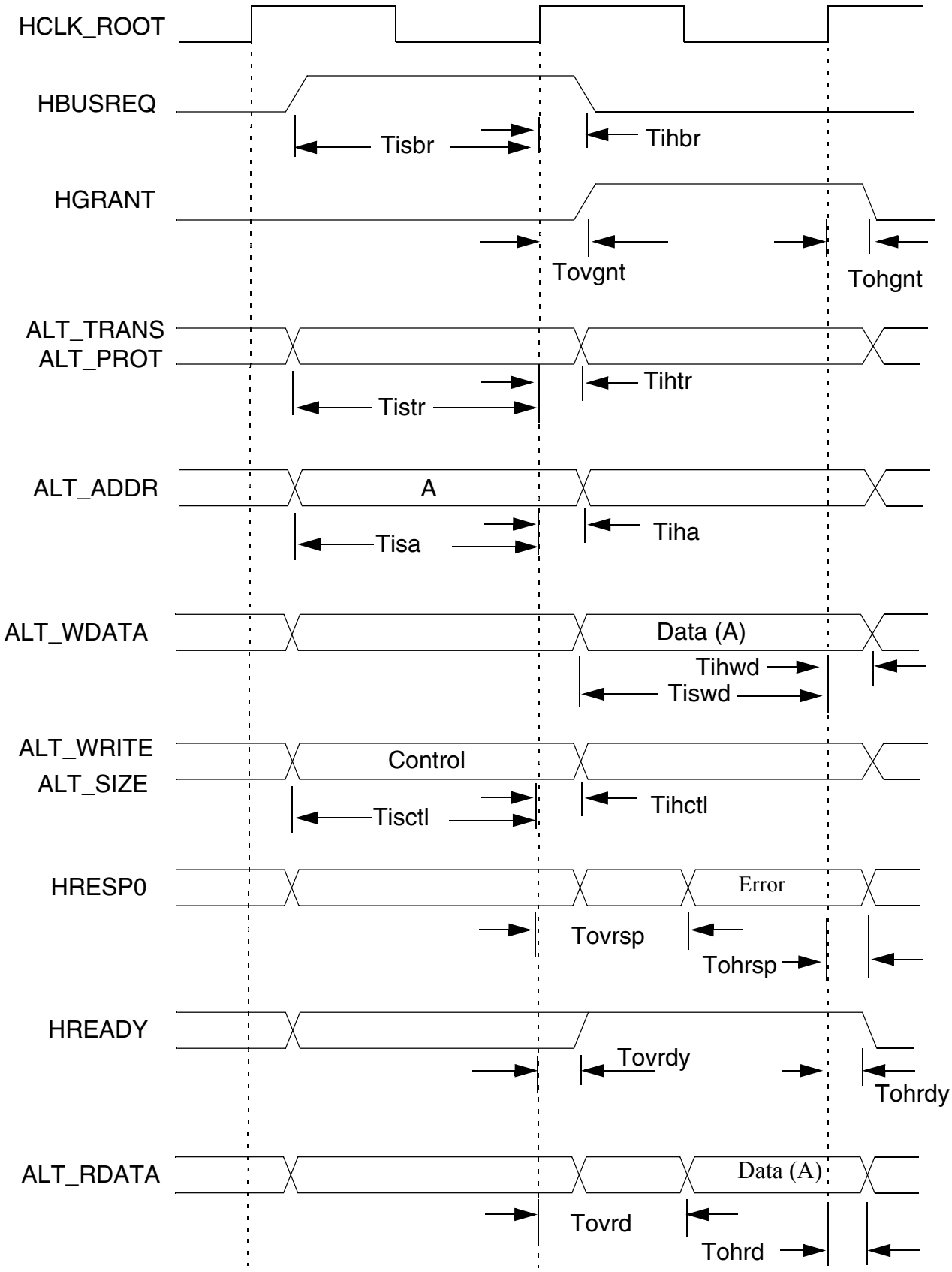


Figure 38-17. Alternate Bus Master Timing Parameters

NOTE:

The timing parameters in Table 38-20 are measured with respect to HCLK_ROOT.

The timing of the internal R-AHB bus is different from the external alternate bus master interface due to different loading and the presence of the scan_wrapper within the arm7_platform. R-AHB bus timing parameters in Table 38-20 take into account scan wrapper delays of 1.3 ns on outputs. ALT_RDATA does not traverse through the wrapper, but is buffered in the CARB. Some margin is added such that ARM7 timing be the critical path as opposed to an alternate bus master.

Table 38-20. Alternate Bus Master AC Timing Parameters

Description	Parameter	Timing (ns)
HCLK_ROOT minimum clock period	T_{clk}	18
HBUSREQ setup time before HCLK_ROOT	T_{isbr}	11.8
HBUSREQ hold time after HCLK_ROOT	T_{ihbr}	>0
HGRANT valid time after HCLK_ROOT	T_{ovgnt}	9.1
HGRANT hold time after HCLK_ROOT	T_{ohgnt}	>0
ALT_TRANS / ALT_PROT setup time before HCLK_ROOT	T_{istr}	11.75
ALT_TRANS / ALT_PROT hold time after HCLK_ROOT	T_{ihtr}	>0
ALT_ADDR[31:0] setup time before HCLK_ROOT	T_{isa}	11.75
ALT_ADDR[31:0] hold time after HCLK_ROOT	T_{iha}	>0
ALT_WDATA Write Data setup time before HCLK_ROOT	T_{iswd}	11.75
ALT_WDATA Write Data hold time after HCLK_ROOT	T_{ihwd}	>0
ALT_WRITE / ALT_SIZE control signal setup time before HCLK_ROOT	T_{isctl}	11.75
ALT_WRITE / ALT_SIZE control signal hold time after HCLK_ROOT	T_{ihctl}	>0
HRESP0 valid time after HCLK_ROOT	T_{isrsp}	13.1
HRESP0 hold time after HCLK_ROOT	T_{ihrsp}	>0

NOTE:

The timing parameters in Table 38-20 are measured with respect to HCLK_ROOT.

The timing of the internal R-AHB bus is different from the external alternate bus master interface due to different loading and the presence of the scan_wrapper within the arm7_platform. R-AHB bus timing parameters in Table 38-20 take into account scan wrapper delays of 1.3 ns on outputs. ALT_RDATA does not traverse through the wrapper, but is buffered in the CARB. Some margin is added such that ARM7 timing be the critical path as opposed to an alternate bus master.

Table 38-20. Alternate Bus Master AC Timing Parameters

Description	Parameter	Timing (ns)
HREADY valid time after HCLK_ROOT	T_{ovrdy}	14.8
HREADY hold time after HCLK_ROOT	T_{ohrdy}	>0
ALT_RDATA Read Data valid time after HCLK_ROOT	T_{ovrd}	17.3
ALT_RDATA Read Data hold time after HCLK _ROOT	T_{ohrd}	>0
HMASTER setup time before HCLK_ROOT	T_{ovmst}	10.2
HMASTER hold time after HCLK_ROOT	T_{ohmst}	>0

38.13.4 ARM7TDMI-S Internal Timing

This section will present timing requirements for miscellaneous ARM7TDMI-S signals which are not been covered by the R-AHB slave timing specification. **Table 38-21** shows timing requirements of all ARM7 processor I/O which originate or terminate **internal** to the ARM7 platform.

NOTE:

The timing constraints in Table 38-21 are measured with respect to HCLK_ROOT.

Table 38-21. ARM7TDMI-S Internal Timing Constraints

Description	Parameter	Timing (ns)
CLKEN setup time before HCLK_ROOT		1.0
CLKEN hold time after HCLK_ROOT		>0
ABORT setup time before HCLK_ROOT		4.2
ABORT hold time after HCLK_ROOT		>0
ADDR[31:0] valid time after HCLK_ROOT		8.2
ADDR[31:0] hold time after HCLK_ROOT		>0
TRANS[1:0] valid time after HCLK_ROOT		7.8
TRANS[1:0] hold time after HCLK_ROOT		>0
PROT[1:0] valid time after HCLK_ROOT		9.8
PROT[1:0] hold time after HCLK_ROOT		>0
WRITE valid time after HCLK_ROOT		9.8
WRITE hold time after HCLK_ROOT		>0
SIZE[1:0] valid time after HCLK_ROOT		9.8
SIZE[1:0] hold time after HCLK_ROOT		>0
LOCK valid time after HCLK_ROOT		9.8
LOCK hold time after HCLK_ROOT		>0
WDATA[31:0] valid time after HCLK_ROOT		9.2
WDATA[31:0] hold time after HCLK_ROOT		>0

Table 38-21. ARM7TDMI-S Internal Timing Constraints

Description	Parameter	Timing (ns)
RDATA[31:0] setup time before HCLK_ROOT		0.25
RDATA[31:0] hold time after HCLK_ROOT		>0
IRQ_B setup time before HCLK_ROOT		2.5
IRQ_B hold time after HCLK_ROOT		>0
FIRQ_B setup time before HCLK_ROOT		2.5
FIRQ_B hold time after HCLK_ROOT		>0
DBGBREAK setup time before HCLK_ROOT		2.5
DBGBREAK hold time after HCLK_ROOT		>0
DBGACK valid time after HCLK_ROOT		7.5
DBGACK hold time after HCLK_ROOT		>0

38.13.5 ARM7TDMI-S External Timing

This section will present timing requirements for miscellaneous ARM7TDMI-S signals which are not been covered by the R-AHB slave timing specification. Table 38-22 shows timing requirements of all ARM7 processor I/O which originate or terminate **external** to the ARM7 platform.

NOTE:

The timing constraints in Table 38-22 are measured with respect to HCLK_ROOT.

Table 38-22. ARM7TDMI-S External Timing Constraints

Description	Parameter	Timing
DBGEN setup time before HCLK_ROOT		3
DBGEN hold time after HCLK_ROOT		>0
DBGRQ setup time before HCLK_ROOT		2.5
DBGRQ hold time after HCLK_ROOT		>0
DBGTCKEN setup time before HCLK_ROOT		5
DBGTCKEN hold time after HCLK_ROOT		>0
DBGTDI setup time before HCLK_ROOT		4
DBGTDI hold time after HCLK_ROOT		>0
DBGTDO valid time after HCLK_ROOT		7.8
DBGTDO hold time after HCLK_ROOT		>0
DBGTDOEN_B valid time after HCLK_ROOT		7.8
DBGTDOEN_B hold time after HCLK_ROOT		>0
DBGTMS setup time before HCLK_ROOT		4
DBGTMS hold time after HCLK_ROOT		>0
DBGTRST_B setup time before HCLK_ROOT		6
DBGTRST_B hold time after HCLK_ROOT		>0
SCANENABLE setup time before HCLK_ROOT		2

Table 38-22. ARM7TDMI-S External Timing Constraints

Description	Parameter	Timing
SCANENABLE hold time after HCLK_ROOT		>0
SCANIN setup time before HCLK_ROOT		3
SCANIN hold time after HCLK_ROOT		
SCANOUT valid time after HCLK_ROOT		8
SCANOUT hold time after HCLK_ROOT		

Chapter 39
ARM7TDMI-S (ARM7)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	05/07/03		Updated for LTE specification release.

Reference Document

Document	Location
ARM7TDMI-S Rev 4	http://www.arm.com/Documentation/Manuals/

Chapter 40
ARM7 Platform Core Arbiter (CARB)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	4/16/2002	John Oakley	Updated ALT_MASTER width [2:0]
0.2	05/07/03		Updated for LTE specification release.

Reference Documents

ARM7TDMI-S Technical Reference Manual
AMBA Specification Rev 2.0

40.1 Introduction

The ARM7 Platform Core Arbiter allows an alternate bus master to take control of the platform system bus, called the Reduced AHB, to perform reads and writes to peripherals or memory. The Core Arbiter arbitrates between a single alternate master port and the ARM7. A typical system will have a secondary arbiter external to the ARM7 Platform, called an External Arbiter, that will arbitrate in a round robin fashion between multiple alternate masters for the single alternate master port to the ARM7 platform. Various alternate masters may include the DMAC, GEM or TCM. Refer to chapter AMARB for details on the External Arbiter. Figure 40-1 shows how the Core Arbiter (CARB) is connected in the system.

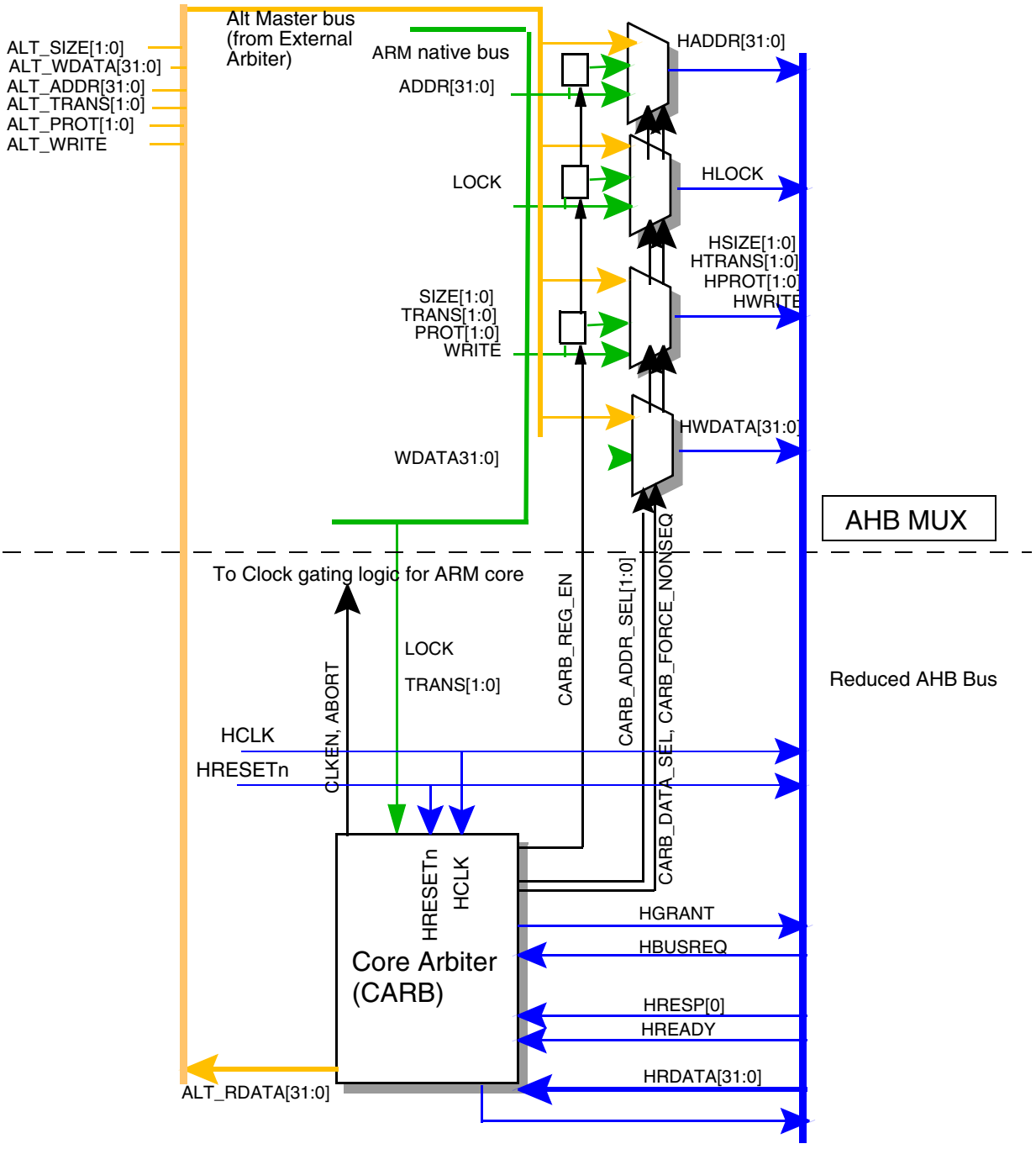


Figure 40-1. ARM7 Platform Core Arbiter Block Diagram

Note that the Core Arbiter provides the control logic for the alternate master handoff. The muxing of the Alternate master bus with the ARM7 native bus is done in the AHB Mux Block.

Features of the ARM7 Platform Arbiter include:

- Simple control interface. No software programmability needed.
- Single cycle handoff to the external arbiter
- Controls the ARM Core clock enable during handoff and entrance/exit into low power modes.

40.2 Core Arbiter Pin Description

.

Table 40-1. ARM7 Core Arbiter Pin List

Pin Name	Type	Description
hclk	Input	AHB Reference Clock
hresetn	Input	AHB Reset
clken	Output	ARM7TDMI-S Clock Enable
hbusreq	Input	AHB Alternate Master Request
hgrant	Output	AHB Alternate Master Grant
hresp[0]	Input	AHB Response - Used to determine if transfer completed successfully
hready	Input	AHB Transfer Done
hrdata[31:0]	Input	AHB Read Data Bus
aitc_blk_arb	Input	Indicates that the interrupt has been detected and to hold off the bus grant till the interrupt (IRQ/FIQ) is serviced
alt_rdata[31:0]	Output	ALT Master Bus Read Data. Same as HRDATA[31:0] but gated off when HRDATA is not being used by an external master.
carb_addr_sel[1:0]	Output	Indicates to AHB MUX which path to gate to AHB BUS. SEL[1:0] for address timed signals
carb_data_sel	Output	Indicates to AHB MUX which path to gate to AHB BUS for data timed signals

Table 40-1. ARM7 Core Arbiter Pin List

Pin Name	Type	Description
carb_reg_en	Output	Used by AHB Mux to register address based data during arbitration handover.
trans[1:0]	Input	ARM7TDMI-S address timed attribute indicating the nature of transfer (idle, busy, non-seq, seq)
lock	Input	ARM7TDMI-S address timed attribute indicating the relationship between two consecutive accesses.
carb_abort	Output	Indicates to the ARM7TDMI-S that the request was aborted by the slave.
alt_master[2:0]	Input	3-bit input indicating the alternate master making the request for the R-AHB bus
hmaster[2:0]	Output	3 bit output for the R-AHB bus indicating which alternate master is making the request
carb_force_nonseq	OUTPUT	Used by the AHB Mux to force the access on the AHB to be non-sequential.

40.3 External Arbiter Programming Model

There are no registers or programmability in the ARM7 Core Platform Arbiter.

40.4 Arbiter Operation

The core arbiter is responsible for granting the reduced AHB bus on a external bus request. On a bus request from the external arbiter (Chapter 42, “Alternate Master Arbiter (AMARB),”) the ARM7TDMI-S completes the current cycle and grants the bus by asserting the HGRANT signal. The address/attributes are driven to the AHB Mux (see ARM7 Platform Spec) from the AMARB (alternate master to the CARB). The core arbiter is responsible selecting the alternate master address/attributes on the reduced AHB bus. The address/attributes are driven on the bus till the request is taken by the slave responsible for servicing the request. If there are no more requests from the external arbiter, the HBUSREQ is deasserted. The HGRANT signal from the core arbiter is asserted till the end of the data phase of the external arbiter request.

When the HGRANT is asserted by the core arbiter, the CLKEN to the ARM7TDMI-S is negated unless the request from the CPU is IDLE/BUSY. Since the negation of the CLKEN is not done till the end of the data phase of the last request, the next request is “taken” from the ARM7TDMI-S which registered and provides as an input to the AHB Mux (unless the request is IDLE/BUSY). Once the external arbiter is done and if there is a pending (registered) request from the ARM7TDMI-S, the mux selects the registered entries and the pending request is driven on the reduce AHB.

40.4.1 Alternate Master Bus Handoff

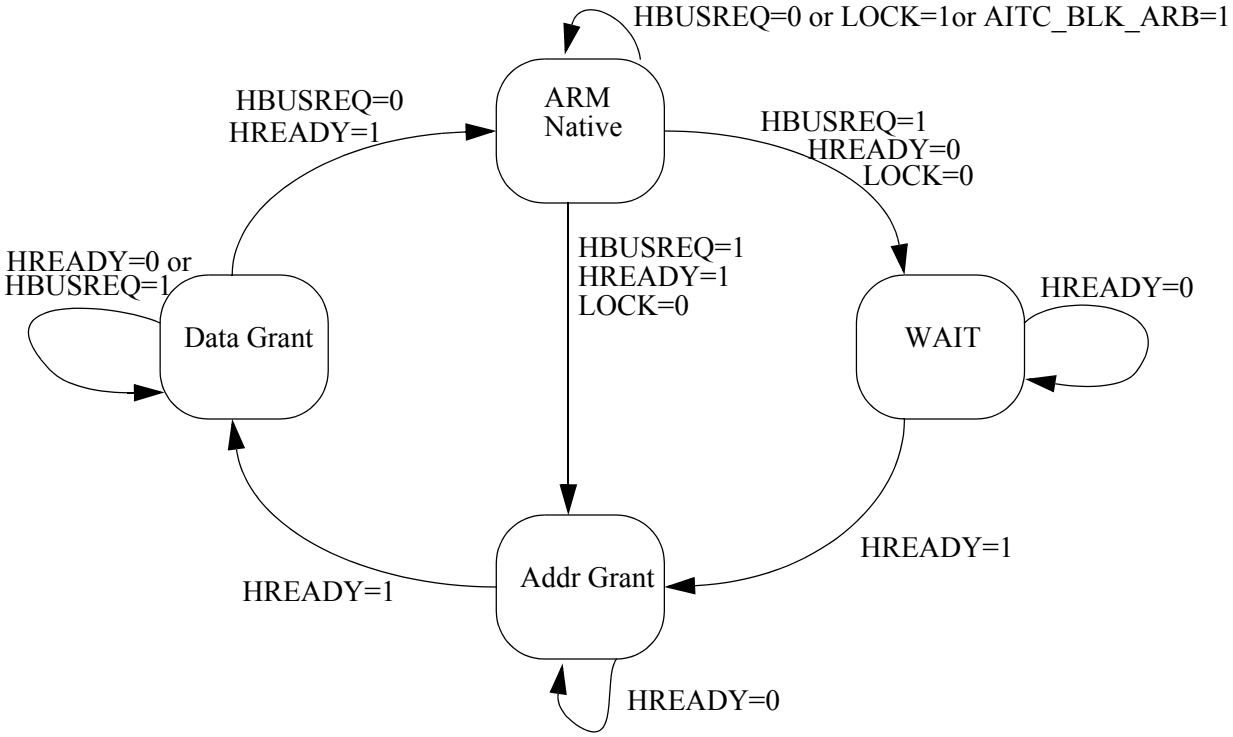


Figure 40-2.

The state machine is implemented as one-hot, i.e each state is represented by a flip-flop. This is done to improve the performance by preventing the additional decode/encode of bits of the state machine. Also the state machine above is responsible for controlling the AHB mux. The select signals used for controlling the mux could be timing critical. One-hot implementation should help improve the timing path.

40.4.1.1 HCLK

HCLK is an input for the ARM7 platform CLK from the CCM Module.

40.4.1.2 HRESET_B

HRESET_B is an input reset to the ARM7 platform

40.4.1.3 HBUSREQ

HBUSREQ is an input to the ARM7 platform indicating that the alternate master requires the AHB bus.

40.4.1.4 HGRANT

HGRANT is an OR of Addr Grant state and Data Grant state. It is asserted till the data phase of the last alternate master request is complete. The HGRANT is asserted when the alternate address/attributes are driven on the RAHB bus. Please note that the address and attributes by the alternate master need to be driven with the HBUSREQ signal.

40.4.1.5 HRESP0

HRESP0 is an input to the CARB module from the AHB Mux. It is used for generating the CLKEN signal and the ABORT signal to the ARM7TDMI-S.

40.4.1.6 HREADY

HREADY is an input signal used to generate the CLKEN signal and determine when a request is “taken” from the ARM7TDMI-S.

40.4.1.7 LOCK

LOCK is an input signal to the CARB from the ARM7TDMI-S and is used to prevent the grant of the AHB bus when this signal is asserted.

40.4.1.8 HRDATA[31:0]

HRDATA[31:0] is a read data bus input to the CARB module from the AHB Mux. HRDATA[31:0] is sent to the alternate master during an alternate master read operation. It is conditioned with the bus arbitration signals in the CARB to prevent it from toggling when there is no alternate bus master activity.

40.4.1.9 TRANS[1:0]

This is a 2 bit input signal from the ARM7TDMI-S indicating the nature of transfer request.

40.4.1.10 AITC_BLK_ARB

AITC_BLK_ARB signal is generated by the AITC module to indicate that an interrupt is pending (IRQ/FIQ) and the AITC has been programmed to prevent a pending interrupt from being pre-empted by an alternate master. The CARB uses this signal in a similar fashion as the LOCK signal and prevents the assertion of the HGRANT till the AITC_BLK_ARB has negated. A stalemate condition can occur when

the FIQ/IRQ (along with a control bit) is asserted by the AITC and the ARM7TDMI-S has the exceptions disabled. The HGRANT will never be asserted as the interrupt will never be handled. We term this as a software error.

40.4.1.11 CARB_REG_EN

CARB_REG_EN signal is asserted when a request from the ARM7TDMI-S is considered to be taken and the alternate master address/attributes are on the RAHB bus. It is used to register the address/attributes from the ARM7TDMI-S when the alternate master is driving the RAHB.

40.4.1.12 CARB_ADDR_SEL[1:0]

CARB_ADDR_SEL signal is used to control the AHB address mux. It chooses between the three addresses timed sources viz. ARM7TDMI-S native bus, Registered ARM7TDMI-S values and Alternate master address/attributes. The AHB address mux switches from the native ARM bus upon external bus request. If there is a pending request after the alternate master bus request is deasserted, the registered arm native bus is selected.

Table 40-2. CARB_ADDR_SEL Decode

CARB_ADDR_SEL[1:0]	Description
00	ARM Native
01	Alternate Master
10	Registered ARM Native Request
11	Reserved

40.4.1.13 CARB_DATA_SEL

CARB_DATA_SEL signal is used to control the AHB data mux. It chosen between the alternate master write data versus the ARM7TDMI-S write data.

Table 40-3. CARB_DATA_SEL Decode

CARB_DATA_SEL	Description
0	ARM Native
1	Alternate Master

40.4.1.14 ALT_RDATA[31:0]

ALT_RDATA is the 32- bit read data bus driven to the alternate master. This is gated off (not toggling) during ARM7TDMI-S accesses. The alternate master monitors this data bus along with HREADY to determine the data on a read access.

40.4.1.15 CLKEN

CLKEN is the clock enable signal generated for the ARM7TDMI-S. This signal is generated by monitoring the HREADY (from the RAHB) and HRESP[0].

40.4.1.16 CARB_ABORT

This signal is generated for the ARM7TDMI-S and connected to the abort pin on the CPU. CARB_ABORT is masked during an alternate master transaction on the R-AHB bus. Only when the ARM7TDMI-S makes a request on the bus and if HRESP0 is asserted, the CARB_ABORT signal is asserted.

40.4.1.17 ALT_MASTER[2:0]

ALT_MASTER[2:0] is a 3bit input indicating the nature of the alternate master making the request. See Table 40-4 on page 40-8 for more details on mapping ALT_MASTER[2:0] to HMASTER[2:0].

The ALT_MASTER[2:0] will be routed through the CARB to the HMASTER[2:0] output when the alternate master controls the R-AHB.

40.4.1.18 HMASTER[2:0]

This is a 3-bit output signal indicating which alternate master is driving the R-AHB bus.

Table 40-4. Mapping ALT_MASTER[2:0] to HMASTER[2:0]

HMASTER[2:0]	Description	ALT_MASTER[2:0]
000	ARM7TDMI-S	n/a
001	Port 0	001
010	Port 1	010
011	Port 2	011
100	CCM	100
101	Port 3	101
110	reserved	110
111	reserved	111 000

If the alternate master controls the R-AHB and the ALT_MASTER[2:0] is equal to “000” indicating the ARM7TDMI-S, the CARB will map HMASTER[2:0] to “111”.

40.4.1.19 CARB_FORCE_NONSEQ

This signal is used by the AHB mux to force the HTRANS[1:0] on the AHB bus to be non-sequential request. This signal is asserted for the following cases:

a) On the first SEQ/NSEQ access after bus handoff (‘ARM7TDMI-S to ALT Master’ or ‘ALT Master to ARM7TDMI-S’).

40.4.2 Low Power Mode Operation

The clock gating to the ARM7TDMI-S is done by the core arbiter. When the AHB bus is granted away the clock enable signal to the ARM7TDMI-S is deasserted for all memory operations by the ARM7TDMI-S. During low power mode, the clock control module (see Chapter 5, “Clock Control Module (CCM),”) asserts bus request to the external arbiter. The external arbiter asserts the HBUSREQ signal to the core arbiter. The CARB asserts the HGRANT signal and negates the ARM7TDMI-S clock enable signal.

40.4.3 Operation during DEBUG Mode

During DEBUG mode, the clocks of the ARM7TDMI-S will be on. If the HGRANT is asserted and a debug event is encountered, the CARB will force the clocks to be on during the duration of the debug event. During internal cycles (DEBUG mode), the CARB will generate the clock enable signal (different from the AHB HREADY signal). If the ARM7TDMI-S requires the memory bus during DEBUG then the CARB will negate clock enable until it gets the memory bus to perform the access.

40.4.4 Differences between AHB Bus Arbitration and CARB

The CARB is not responsible for generating the HMASTER[3:0] (see AHB spec) or HMASTLOCK (see AHB spec) signal. The CARB only supports a single master. ARM7TDMI-S is the default master until the HBUSREQ signal is asserted. After completing the current request (or subsequent locked requests). the bus is granted away for as long as the HBUSREQ is asserted. This requires the address/attributes be driven along with the request for the bus (HBUSREQ assertion) and not after the assertion of HGRANT.

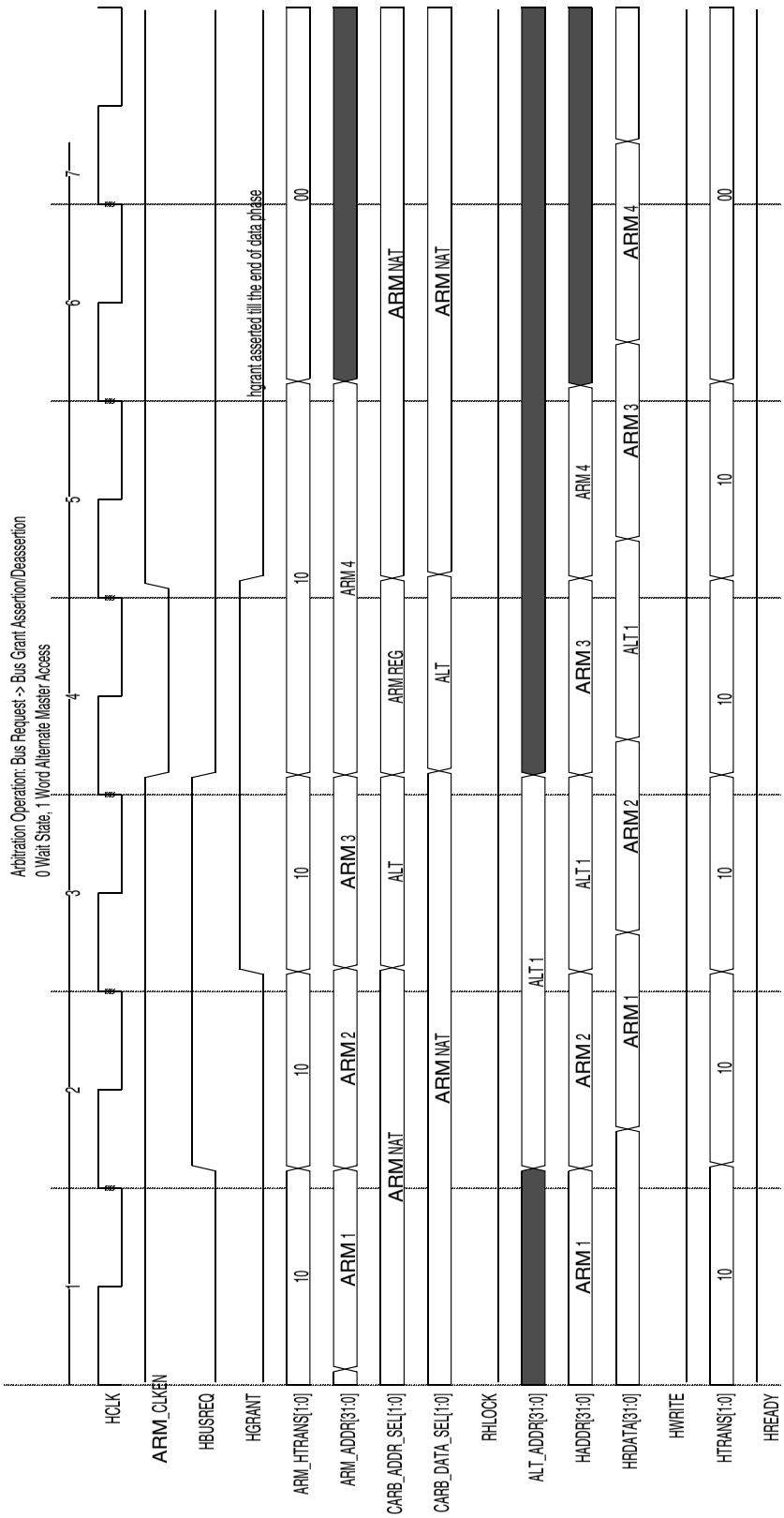


Figure 40-3. 0 wait state bus handoff

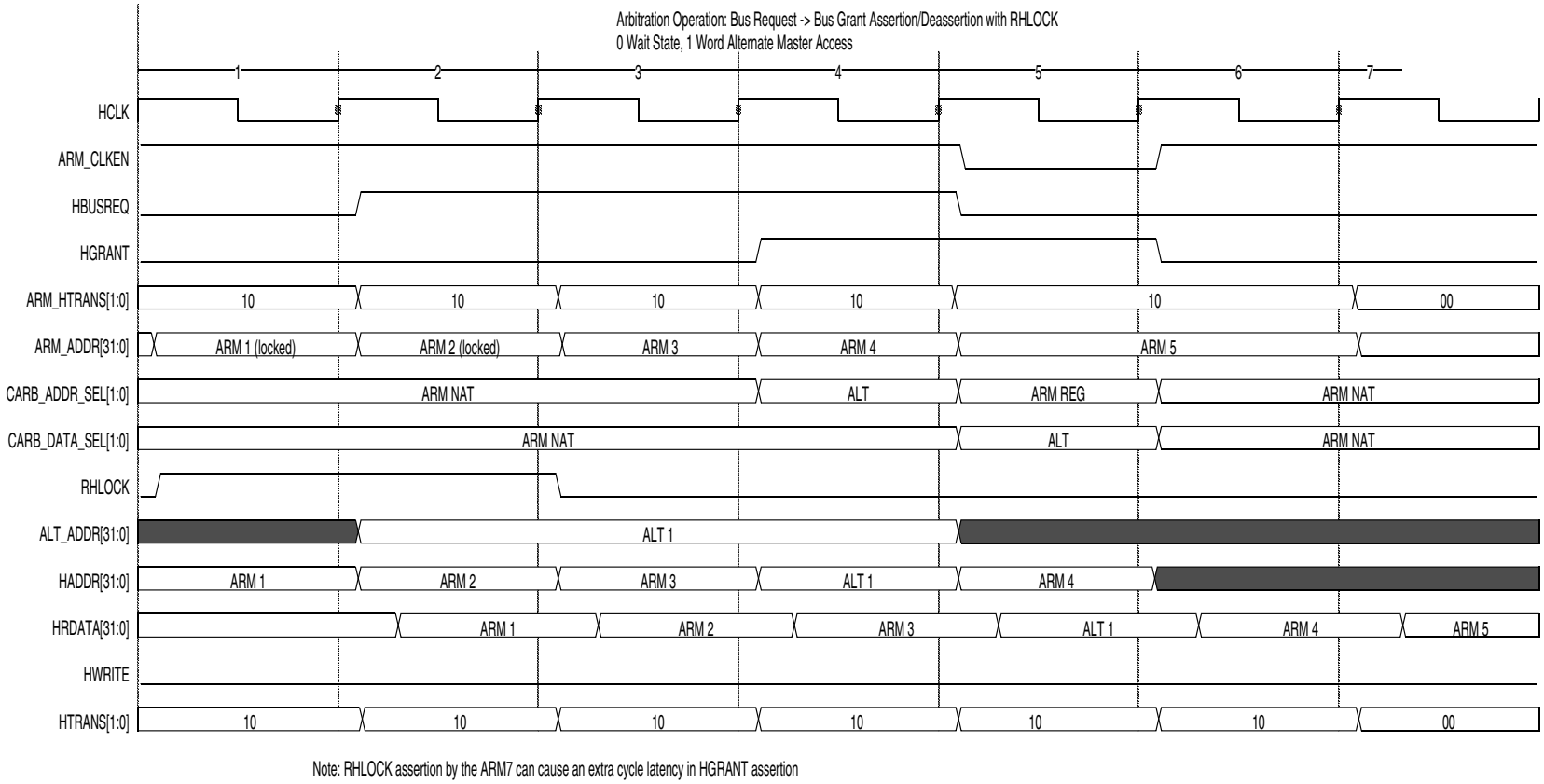


Figure 40-4. 0 wait state, with LOCK asserted bus handoff

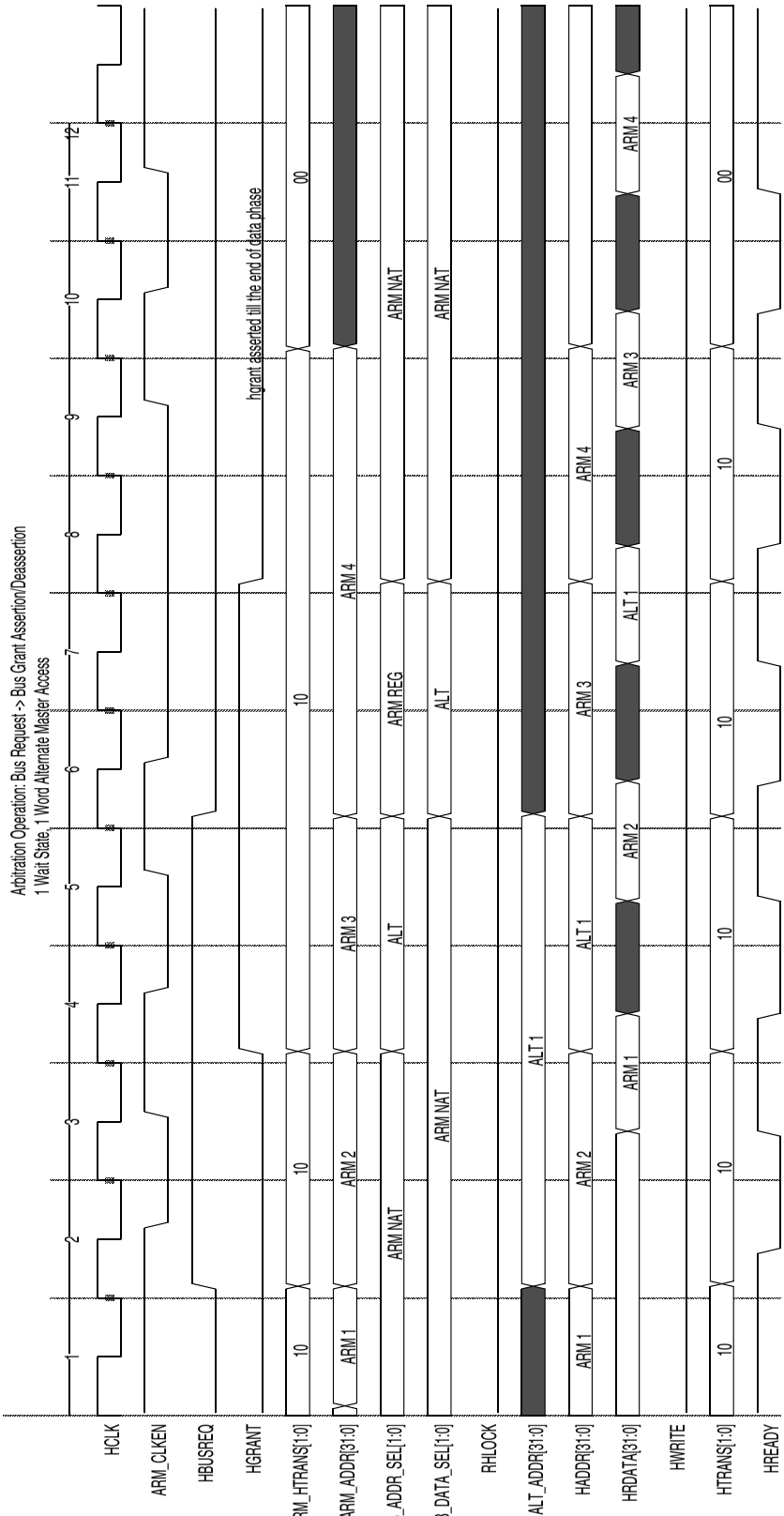


Figure 40-5. 1 wait state bus handoff

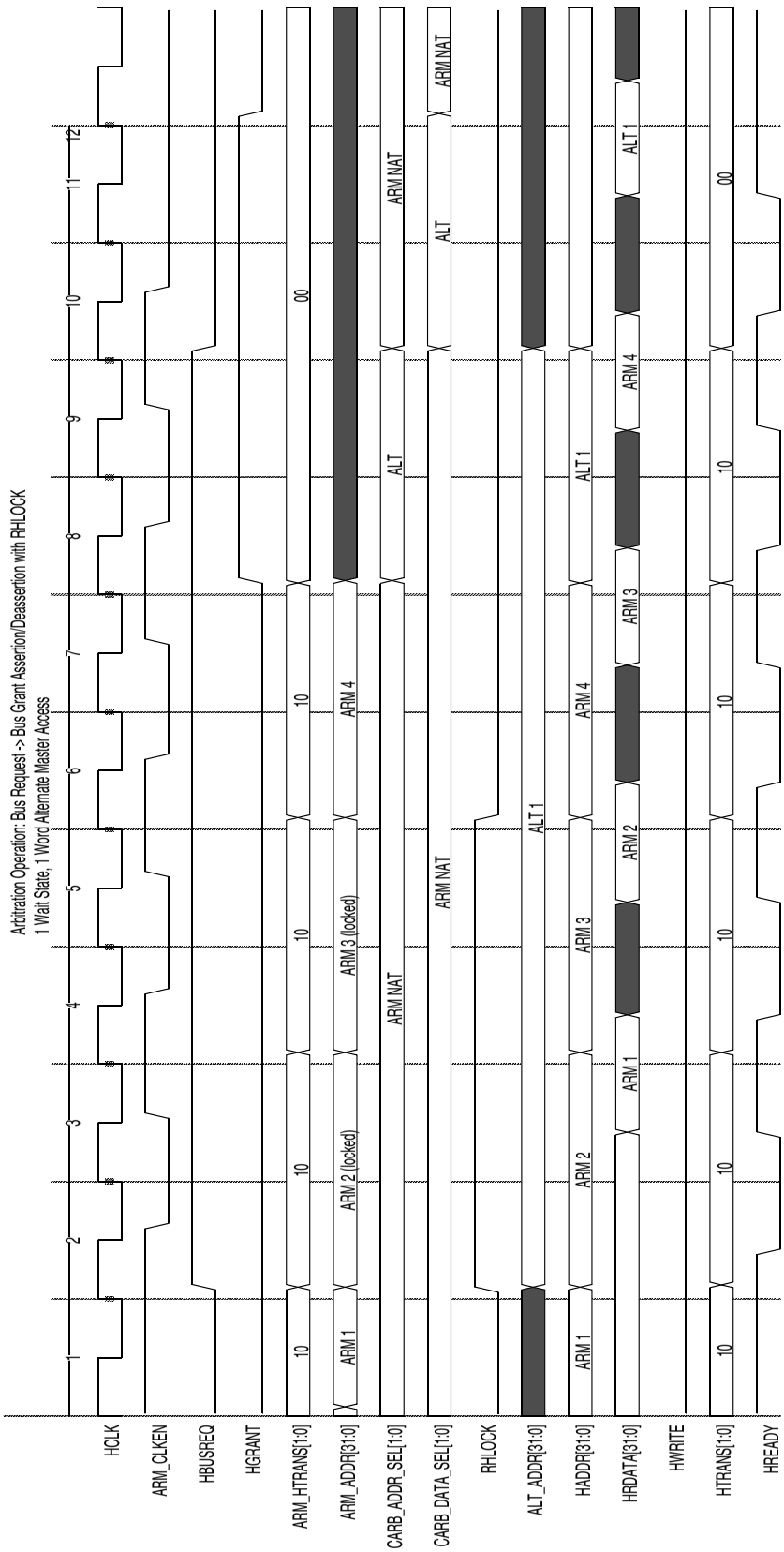


Figure 40-6. 1 wait state, with LOCK asserted bus handoff

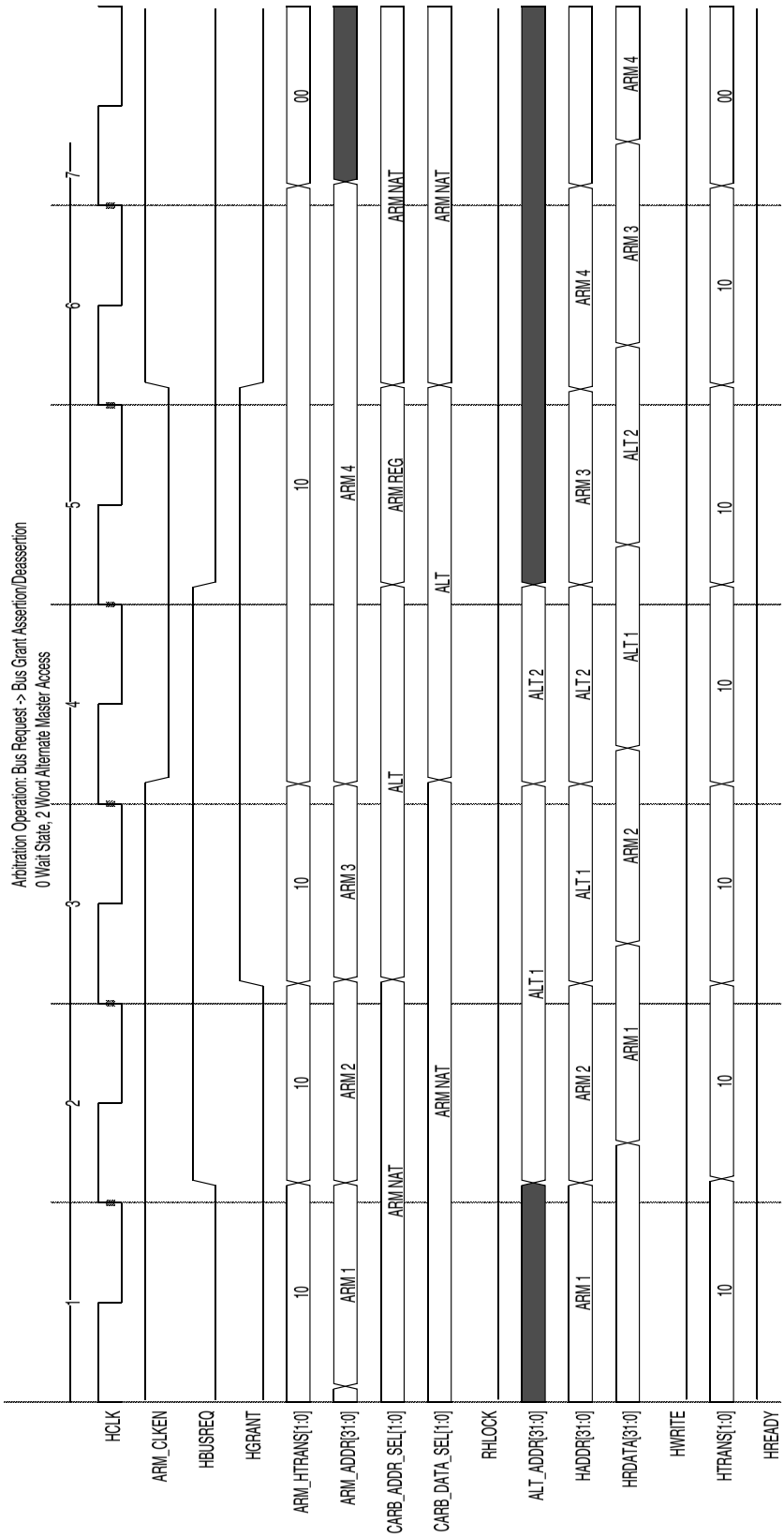


Figure 40-7. 0 wait state, 2 word alternate master access

Chapter 41
R-AHB IP Interface (AIPI)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	04/10/02	Lloyd Khuc	Added the ipg_wait signal output
0.2	05/07/03		Updated for LTE specification release.

41.1 Overview

This section provides an overview of the R-AHB to IP bus interface (AIPI).

R-AHB IP Interface module (AIPI Ver1.0) acts as an interface between the R-AHB (Reduced ARM Advanced High-performance Bus) and lower bandwidth peripherals conforming to the IPbus specification version 2.0 (10 December, 1999) available from the Motorola IP repository.

41.1.1 Highlights

- All peripheral read transactions require a minimum of two system clocks (R-AHB side) and all write transactions require a minimum of three system clocks (R-AHB side).
- The AIPI only supports the Green and Sky-blue line signals. This interface is only meant for slave mode peripherals.
- The AIPI supports 8-bit, 16-bit and 32-bit IPbus peripherals. (Byte, half word and word reads and write are supported to each.)
- The AIPI supports multi-cycle accesses (16-bit operations to 8-bit peripherals and 32-bit operations to 16-bit and 8-bit peripherals).
- The AIPI supports 31 external IPbus peripherals.
- The AIPI uses one single asynchronous reset and one global clock.
- The AIPI is implemented using MUX-D scan methodology for testability.

The following IPbus signals/operations are not supported:

- Sky-blue line signals **ips_clk**, **ips_seq_access**, and **ips_test_access** are not supported.
- Green line signals **ipg_clk**, **ipg_soft_reset_b**, **ipg_hard_async_reset_b**, **ipg_hard_sync_reset_b**, **ipg_powseq_dr_off**, **ipg_stop_ack** and **ipg_clk_en** are not supported.

Please note the following:

- The **ips_module_en[0]** signal is not available since the AIPI registers occupy this location in memory.
- An external mux is required since only one 32-bit data bus, one **ips_xfr_err** and one **ips_xfr_wait** input is available.
- Green line signals **ipg_clk** and **ipg_hard_async_reset_b** will be provided elsewhere in the system.

41.1.2 General

The AIPI is the interface between the R-AHB and on-chip IPbus v2.0 peripherals as shown in Figure 41-1.

IPbus peripherals are modules that contain readable/writable control and status registers. The R-AHB master reads and writes these registers through the AIPI. The AIPI generates module enables, the module address, transfer attributes, byte enables and write data as inputs to the IPbus peripherals. The AIPI captures read data (qualified by **ips_xfr_wait**) from the IPbus v2.0 interface and drives it on the R-AHB. The AIPI terminates the transfer by asserting **aipi_hready_out**.

The register maps of all IPbus peripherals are located on 4096 byte boundaries. Each IPbus peripheral is allocated one 4-Kilobyte block (minimum block size) of the memory map, configured as 1024 32-bit internal registers (or 2048 16-bit internal registers, or 4096 8-bit internal registers), activated by one of 31 module enables from the AIPI. Up to 31 IPbus peripherals may be implemented, occupying contiguous blocks of 4 Kilobytes, for a total of 124 Kilobytes. The exact address assignment for the IPbus peripherals is system dependent, and is defined in the system specification. Each IPbus peripheral will select its internal registers based on the address driven on the **ips_addr** signals.

The AIPI is responsible for telling the IPbus peripherals if the access is in supervisor or user mode. The AIPI may block user mode accesses to certain IPbus peripherals or it may allow the individual IPbus peripherals to determine if user mode accesses are allowed. Please see Section 41.4, “AIPI Registers,” , for more information.

The AIPI will support multi-cycle accesses to IPbus peripherals when the R-AHB master requests data transfers that are larger than the targeted IPbus peripheral’s data bus width. Table 41-1, *R-AHB to IPbus v2.0 Interface Operation (Big Endian Only)* provides more information on both single-cycle and multi-cycle accesses.

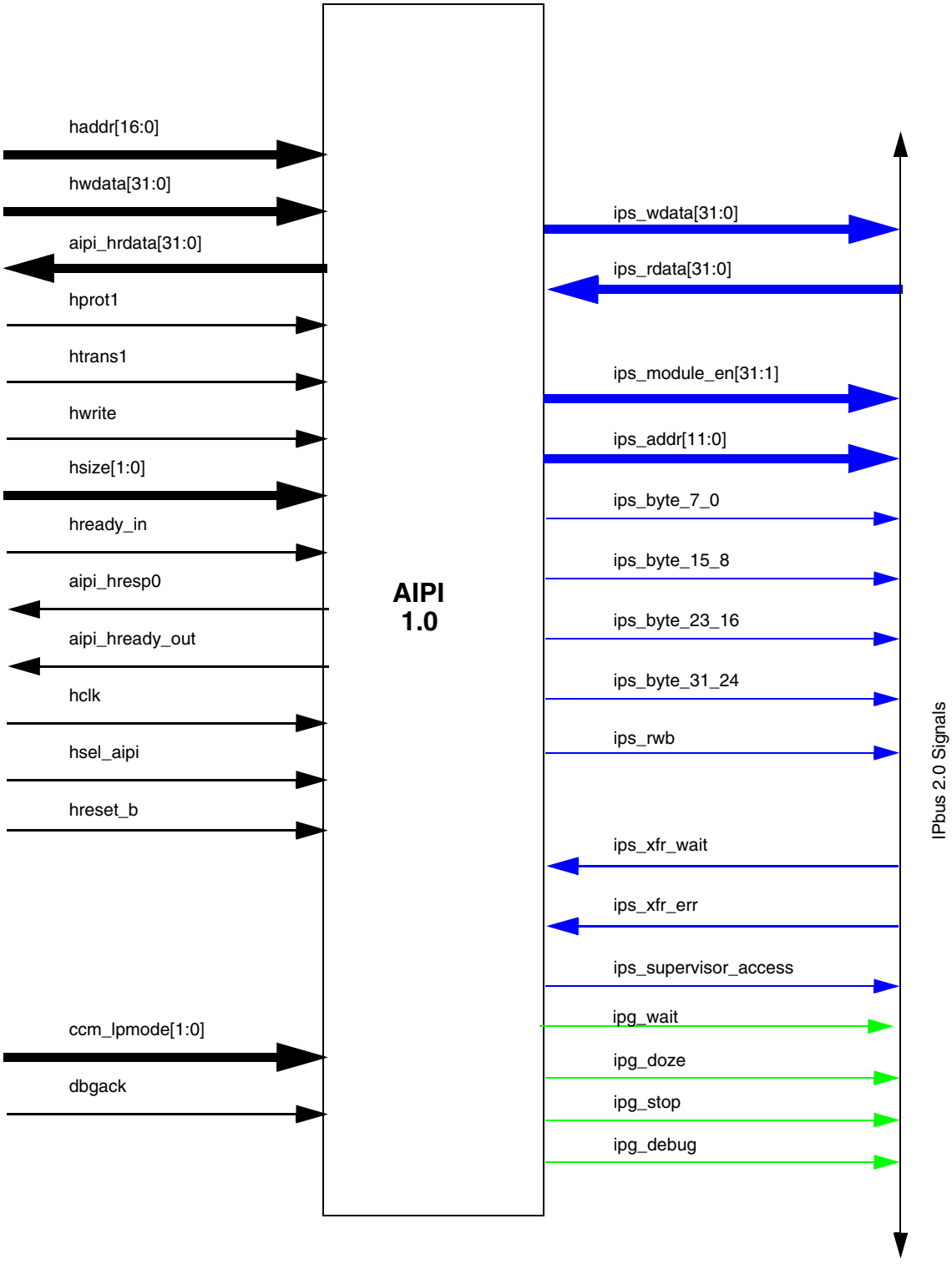


Figure 41-1. AIPI interface

Table 41-1. R-AHB to IPbus v2.0 Interface Operation (Big Endian Only)

Transfer Size	haddr		IPbus size	ips_addr		Active bus section (R-AHB to IPbus)			
	[1]	[0]		[1]	[0]	R-AHB[31:24]	R-AHB[23:16]	R-AHB[15:8]	R-AHB[7:0]
Byte	0	0	8 bit	0	0	ips_data[7:0]			
	0	1		0	1		ips_data[7:0]		
	1	0		1	0			ips_data[7:0]	
	1	1		1	1				ips_data[7:0]
	0	0	16 bit	0	X	ips_data[15:8]	-	-	-
	0	1				-	ips_data[7:0]	-	-
	1	0		1	X	-	-	ips_data[15:8]	-
	1	1				-	-	-	ips_data[7:0]
	0	0	32 bit	X	X	ips_data[31:24]	-	-	-
	0	1				-	ips_data[23:16]	-	-
	1	0		X	X	-	-	ips_data[15:8]	-
	1	1				-	-	-	ips_data[7:0]
Half Word	0	NA	8 bit	0	0	ips_data[7:0]			
					1		ips_data[7:0]		
	1			1	0			ips_data[7:0]	
					1				ips_data[7:0]
	0		16 bit	0	X	ips_data[15:8]	ips_data[7:0]	-	-
	1			1	X	-	-	ips_data[15:8]	ips_data[7:0]
	0		32 bit	X	X	ips_data[31:24]	ips_data[23:16]	-	-
	1			X	X	-	-	ips_data[15:8]	ips_data[7:0]
Word	NA	NA	8 bit	0	0	ips_data[7:0]			
					1		ips_data[7:0]		
				1	0			ips_data[7:0]	
					1				ips_data[7:0]
			16 bit	0	X	ips_data[15:8]	ips_data[7:0]	-	-
				1	X	-	-	ips_data[15:8]	ips_data[7:0]
			32 bit	X	X	ips_data[31:24]	ips_data[23:16]	ips_data[15:8]	ips_data[7:0]

41.2 AIPI 1.0 Interface Signals

41.2.1 R-AHB Master Interface signals

41.2.1.1 Address Bus (haddr[16:0])

These input signals are the address bus from the R-AHB. The AIPI will decode **haddr[16:12]** to determine which of the IPbus peripherals (or the AIPI itself) is being accessed. The rest of the address bits are passed on to the IPbus peripherals to allow decoding of their registers. The upper bits of the address bus are not needed because there is a centralized address decoder on the platform that generates **hsel_aipi** which is set when the AIPI is being accessed.

Table 41-2. IPbus Peripheral Location Decode[1]

haddr[16:12]	ips_module_en	haddr[16:12]	ips_module_en	haddr[16:12]	ips_module_en
00000	AIPI registers	01011	11	10110	22
00001	1	01100	12	10111	23
00010	2	01101	13	11000	24
00011	3	01110	14	11001	25
00100	4	01111	15	11010	26
00101	5	10000	16	11011	27
00110	6	10001	17	11100	28
00111	7	10010	18	11101	29
01000	8	10011	19	11110	30
01001	9	10100	20	11111	31
01010	10	10101	21		

1. Assumes **hsel_aipi** is asserted.

41.2.1.2 Write Data Bus (hwdata[31:0])

These input signals are the write data bus from the R-AHB master. This bus is used to transfer data from the R-AHB master to the bus slaves during write operations.

41.2.1.3 Read Data Bus (aipi_hrdata[31:0])

These output signals are the read data bus going to the read data mux to be sent on to the R-AHB master. This bus is used to transfer data from the AIPI to the R-AHB master during read operations.

41.2.1.4 Transfer Type (htrans1)

This input indicates the type of the current transfer, which can be sequential/nonsequential or idle/busy. A sequential/nonsequential transfer is indicated when this signal is HIGH and a idle/busy transfer is indicated when this signal is LOW.

41.2.1.5 Transfer Direction (hwrite)

This input indicates a write transfer when HIGH and a read transfer when LOW.

41.2.1.6 Protection Control (hprot1)

This input indicates a privileged transfer when HIGH and a user transfer when LOW.

41.2.1.7 Transfer Size (hsize[1:0])

These input signals indicate the size of the transfer, which is byte (8-bit), halfword (16-bit), or word (32-bit). Table 41-3, *Transfer Size Encoding* shows the encoding of this bit field

Table 41-3. Transfer Size Encoding

hsize[1:0]	Transfer Size
0 0	Byte
0 1	Half-word (2 Bytes)
1 0	Word (4 Bytes)
1 1	Reserved

41.2.1.8 AIPI Transfer Done (aipi_hready_out)

This output signal indicates to the R-AHB master when an AIPI transfer has finished on the bus. This signal may be driven LOW to extend a transfer.

41.2.1.9 Transfer Done In (hready_in)

This input signal indicates to the AIPI when any of the slaves has finished their transfer on the bus.

41.2.1.10 Transfer Response (aipi_hresp0)

This output signal provides the R-AHB master with additional information on the status of the transfer. If **aipi_hresp0** is HIGH, then an error has occurred. If **aipi_hresp0** is LOW, then the transfer was completed successfully.

41.2.1.11 CCM Low Power Mode(ccm_lpmode[1:0])

These input signals from the Clock Control Module (CCM) indicate to the AIPI what power mode the R-AHB master is in. Possible choices are STOP, WAIT, DOZE, and NORMAL. Table 41-4, *R-AHB Master Low Power Mode Encoding* shows the encoding of this bit field. The AIPI simply decodes these signals and passes on **ipg_doze, ipg_stop** and **ipg_wait** to the IPbus peripherals.

Table 41-4. R-AHB Master Low Power Mode Encoding

ccm_lpmd[1:0]	State
0 0	Stop
0 1	Wait

Table 41-4. R-AHB Master Low Power Mode Encoding

ccm_lpmd[1:0]	State
1 0	Doze
1 1	Normal

41.2.1.12 Debug Mode (dbgack)

The R-AHB master asserts this signal to indicate that debug mode has been entered. The AIPI buffers this signal and broadcasts it to the IPbus peripherals.

41.2.2 IPbus v2.0 Interface Signals

41.2.2.1 IP Address Bus (ips_addr[11:0])

These outputs from the AIPI provide the address to the various IPbus peripherals to allow them to decode and select from the 4-Kilobyte space allocated to each IPbus peripheral. This address is registered from the R-AHB. IPbus peripherals that have less than 1K 32-bit (or 2K 16-bit or 4k 8-bit) registers do not have to decode all of the address bits (the registers may be multiply-mapped).

41.2.2.2 IP Read Data Bus (ips_rdata[31:0])

This input bus is the read data path between the AIPI and the IPbus peripherals. The AIPI captures the data from the IPbus peripherals on reads and drives it to the R-AHB on **aipi_hrdata[31:0]**.

Since there are 31 read data busses coming from the IPbus peripherals but only one read data bus input to the AIPI, the integrator must externally mux all the peripheral read data busses based on the **ips_module_en** outputs of the AIPI. Also note that 16-bit IPbus peripherals must connect to the **ips_rdata[15:0]** portion of the IP read data bus and 8-bit IPbus peripherals must connect to the **ips_rdata[7:0]** portion of the IP read data bus.

41.2.2.3 IP write data bus

This output bus is the write data path between the AIPI and the IPbus peripherals. The AIPI drives data on writes from **hwdata[31:0]** to **ips_wdata[31:0]**. This write data bus is common to all the IPbus peripherals.

Please note, 16-bit IPbus peripherals must connect to **ips_wdata[15:0]** and 8-bit IPbus peripherals must connect to **ips_wdata[7:0]**.

41.2.2.4 ips_module_en[31:1]

The AIPI will assert one (and only one) module enable at a time. The module enable will be driven off the rising edge of **hclk** and will remain valid until the rising edge of **hclk** when the **ips_xfr_wait** signal is negated.

The **haddr[16:12]** bits of the R-AHB address are decoded to determine which module select is to be generated. If the selected IPbus peripheral location is occupied, then corresponding **ips_module_en** is asserted, otherwise an error response is returned to the R-AHB and no IPbus activity will occur.

The **ips_module_en[0]** signal is not available externally as this peripheral location is occupied by the AIPI registers. See Section 41.4, “AIPI Registers,” for more information about the AIPI registers.

41.2.2.5 ips_rwb

This signal is used to indicate to the peripheral that current access is either a read or write access. A logic high indicates a read and a logic low indicates a write. It is asserted after the rising edge of **hclk** along with the **ips_module_en**. This signal remains asserted for the duration of the peripheral access.

41.2.2.6 ips_byte_31_24, ips_byte_23_16, ips_byte_15_8, ips_byte_7_0

Byte enables are generated based on the **hsize[1:0]** bits and lower bits of address, i.e. **haddr[1:0]**. These byte enables are used to select the corresponding bytes.

41.2.2.7 ips_xfr_wait

Each connected IPbus peripheral has a separate **ips_xfr_wait** signal. The **ips_xfr_wait** signal is sampled by the AIPI on the rising edge of **hclk**. It should be asserted by the selected peripheral to indicate that the current access cannot be completed in the given cycle and a wait state is needed.

Since there are 31 **ips_xfr_wait** signals coming from the peripherals and only one **ips_xfr_wait** input to the AIPI, the integrator must externally mux all the peripheral **ips_xfr_wait** signals based on the **ips_module_en** outputs of the AIPI. The integrator should insure that if a vacant IPbus peripheral location is selected, the **ips_xfr_wait** input to the AIPI is driven to a logic low.

41.2.2.8 ips_xfr_err

Each IPbus peripheral has a separate **ips_xfr_err** signal connected to interface module. This error signal is asserted by the peripheral to indicate to the AIPI that the current access is not valid and needs to be terminated with an abort. This signal is sampled by the AIPI at the positive edge of **hclk** when the **ips_xfr_wait** signal is negated.

Since there are 31 **ips_xfr_err** signals coming from the peripherals and only one **ips_xfr_err** input to the AIPI the integrator must externally mux all the peripheral **ips_xfr_err** signals based on the **ips_module_en** outputs of the AIPI. The integrator should insure that if a vacant IPbus peripheral location is selected, the **ips_xfr_err** input to the AIPI is driven to a logic high (forcing an error for any access to an unoccupied IPbus peripheral location).

41.2.2.9 ips_supervisor_access

This signal is asserted at the positive edge of **hclk** to indicate to the peripheral that the current access is in supervisor mode.

41.2.2.10 ipg_debug

This output indicates that the peripheral should enter debug mode. This signal is simply a buffered version of the **dbgack** input.

41.2.2.11 ipg_stop

This output indicates that system clocks are shut down. This signal is simply a decoded and buffered version of the **ccm_lpmode[1:0]** inputs. Table 40-4, R-AHB Master Low Mode Encoding shows the encoding of this bit field.

41.2.2.12 ipg_wait

This output indicates that system are in the wait mode. This signal is simply a decoded and buffered version of the **ccm_lpmode[1:0]** inputs. Table 40-4, R-AHB Master Low Mode Encoding shows the encoding of this bit field.

41.2.2.13 ipg_doze

This output indicates for modules to enter low power operation. This signal is simply a decoded and buffered version of the **ccm_lpmode[1:0]** signals. Table 40-4, R-AHB Master Low Mode Encoding shows the encoding of this bit field.

41.2.2.14 ipg_hard_async_reset_b

This signal is not supported, the user should connect to the **hreset_b** input of the platform (which has the same behavior as **ipg_hard_async_reset_b**) to the **ipg_hard_async_reset_b** input to their module (if this input is present).

41.2.2.15 ipg_clk/ips_clk

These signals are not supported by the AIPI. The user should connect these inputs to the same source that is driving the **hclk** input to the platform.

41.3 AIPI Pin List

Table 41-5, *AIPI Pin List* lists all the pins in the AIPI module.

Table 41-5. AIPI Pin List

Pin Name	Direction	Description
R-AHB Master Interface signals		
haddr[16:0]	Input	Address Bus. These input signals are the address bus from the R-AHB. The AIPI will decode **haddr[16:12]** to determine which of the IPbus peripherals (or the AIPI itself) is being accessed. The rest of the address bits are passed on to the IPbus peripherals to allow decoding of their registers. The upper bits of the address bus are not needed because there is a centralized address decoder on the platform that generates **hsel_aipi** which is set when the AIPI is being accessed.
hwdata[31:0]	Input	Write Data Bus. These input signals are the write data bus from the R-AHB master. This bus is used to transfer data from the R-AHB master to the bus slaves during write operations.
aipi_hrdata[31:0]	Output	Read Data Bus. These output signals are the read data bus going to the read data mux to be sent on to the R-AHB master. This bus is used to transfer data from the AIPI to the R-AHB master during read operations.
htrans1	Input	Transfer Type. This input indicates the type of the current transfer, which can be sequential/nonsequential or idle/busy. A sequential/nonsequential transfer is indicated when this signal is HIGH and a idle/busy transfer is indicated when this signal is LOW.
hwrite	Input	Transfer Direction. This input indicates a write transfer when HIGH and a read transfer when LOW.
hprot1	Input	Protection Control. This input indicates a privileged transfer when HIGH and a user transfer when LOW.
hsize[1:0]	Input	Transfer Size. These input signals indicate the size of the transfer, which is byte (8-bit), halfword (16-bit), or word (32-bit). Table 41-3, ***Transfer Size Encoding*** shows the encoding of this bit field
aipi_hready_out	Output	AIPI Transfer Done. This output signal indicates to the R-AHB master when an AIPI transfer has finished on the bus. This signal may be driven LOW to extend a transfer.
hready_in	Input	Transfer Done In. This input signal indicates to the AIPI when any of the slaves has finished their transfer on the bus.
aipi_hresp0	Output	Transfer Response. This output signal provides the R-AHB master with additional information on the status of the transfer. If **aipi_hresp0** is HIGH, then an error has occurred. If **aipi_hresp0** is LOW, then the transfer was completed successfully.

Table 41-5. AIPI Pin List

Pin Name	Direction	Description
ccm_lpmode[1:0]	Input	CCM Low Power Mode. These input signals from the Clock Control Module (CCM) indicate to the AIPI what power mode the R-AHB master is in. Possible choices are STOP, WAIT, DOZE, and NORMAL. Table 41-4, ***R-AHB Master Low Power Mode Encoding*** shows the encoding of this bit field. The AIPI simply decodes these signals and passes on **ipg_doze ipg_stop**and **ipg_wait** to the IPbus peripherals.
dbgack	Input	Debug Mode. The R-AHB master asserts this signal to indicate that debug mode has been entered. The AIPI buffers this signal and broadcasts it to the IPbus peripherals.
IPbus v2.0 Interface Signals		
ips_addr[11:0]	Output	IP Address Bus. These outputs from the AIPI provide the address to the various IPbus peripherals to allow them to decode and select from the 4-Kilobyte space allocated to each IPbus peripheral. This address is registered from the R-AHB. IPbus peripherals that have less than 1K 32-bit (or 2K 16-bit or 4k 8-bit) registers do not have to decode all of the address bits (the registers may be multiply-mapped).
ips_rdata[31:0]	Input	IP Read Data Bus. This input bus is the read data path between the AIPI and the IPbus peripherals. The AIPI captures the data from the IPbus peripherals on reads and drives it to the R-AHB on **aipi_hrdata[31:0]**. Since there are 31 read data busses coming from the IPbus peripherals but only one read data bus input to the AIPI, the integrator must externally mux all the peripheral read data busses based on the **ips_module_en** outputs of the AIPI. Also note that 16-bit IPbus peripherals must connect to the **ips_rdata[15:0]** portion of the IP read data bus and 8-bit IPbus peripherals must connect to the **ips_rdata[7:0]** portion of the IP read data bus.
ips_wdata[31:0]	Output	IP write data bus.This output bus is the write data path between the AIPI and the IPbus peripherals. The AIPI drives data on writes from **hwdata[31:0]** to **ips_wdata[31:0]**. This write data bus is common to all the IPbus peripherals. Please note, 16-bit IPbus peripherals must connect to **ips_wdata[15:0]** and 8-bit IPbus peripherals must connect to **ips_wdata[7:0]**.
ips_module_en[31:1]	Output	The AIPI will assert one (and only one) module enable at a time. The module enable will be driven off the rising edge of **hclk** and will remain valid until the rising edge of **hclk** when the **ips_xfr_wait** signal is negated. The **haddr[16:12]** bits of the R-AHB address are decoded to determine which module select is to be generated. If the selected IPbus peripheral location is occupied, then corresponding **ips_module_en** is asserted, otherwise an error response is returned to the R-AHB and no IPbus activity will occur. The **ips_module_en[0]** signal is not available externally as this peripheral location is occupied by the AIPI registers. See Section 41.4, “AIPI Registers,” for more information about the AIPI registers.

Table 41-5. AIPI Pin List

Pin Name	Direction	Description
ips_rwb	Output	This signal is used to indicate to the peripheral that current access is either a read or write access. A logic high indicates a read and a logic low indicates a write. It is asserted after the rising edge of **hclk** along with the **ips_module_en**. This signal remains asserted for the duration of the peripheral access.
ips_byte_[31:24]	Output	Byte enables are generated based on the **hsize[1:0]** bits and lower bits of address, i.e. **haddr[1:0]**. These byte enables are used to select the corresponding bytes.
ips_byte_[23:16]	Output	Byte enables are generated based on the **hsize[1:0]** bits and lower bits of address, i.e. **haddr[1:0]**. These byte enables are used to select the corresponding bytes.
ips_byte_[15:8]	Output	Byte enables are generated based on the **hsize[1:0]** bits and lower bits of address, i.e. **haddr[1:0]**. These byte enables are used to select the corresponding bytes.
ips_byte_[7:0]	Output	Byte enables are generated based on the **hsize[1:0]** bits and lower bits of address, i.e. **haddr[1:0]**. These byte enables are used to select the corresponding bytes.
ips_xfr_wait	Input	Each connected IPbus peripheral has a separate **ips_xfr_wait** signal. The **ips_xfr_wait** signal is sampled by the AIPI on the rising edge of **hclk**. It should be asserted by the selected peripheral to indicate that the current access cannot be completed in the given cycle and a wait state is needed. Since there are 31 **ips_xfr_wait** signals coming from the peripherals and only one **ips_xfr_wait** input to the AIPI, the integrator must externally mux all the peripheral **ips_xfr_wait** signals based on the **ips_module_en** outputs of the AIPI. The integrator should insure that if a vacant IPbus peripheral location is selected, the **ips_xfr_wait** input to the AIPI is driven to a logic low.
ips_xfr_err	Input	Each IPbus peripheral has a separate **ips_xfr_err** signal connected to interface module. This error signal is asserted by the peripheral to indicate to the AIPI that the current access is not valid and needs to be terminated with an abort. This signal is sampled by the AIPI at the positive edge of **hclk** when the **ips_xfr_wait** signal is negated. Since there are 31 **ips_xfr_err** signals coming from the peripherals and only one **ips_xfr_err** input to the AIPI the integrator must externally mux all the peripheral **ips_xfr_err** signals based on the **ips_module_en** outputs of the AIPI. The integrator should insure that if a vacant IPbus peripheral location is selected, the **ips_xfr_err** input to the AIPI is driven to a logic high (forcing an error for any access to an unoccupied IPbus peripheral location).
ips_supervisor_access	Output	This signal is asserted at the positive edge of **hclk** to indicate to the peripheral that the current access is in supervisor mode.
ipg_debug	Output	This output indicates that the peripheral should enter debug mode. This signal is simply a buffered version of the **dbgack** input.

Table 41-5. AIPI Pin List

Pin Name	Direction	Description
ipg_stop	Output	This output indicates that system clocks are shut down. This signal is simply a decoded and buffered version of the **ccm_lpmode[1:0]** inputs. Table 41-4, ***R-AHB Master Low Power Mode Encoding*** shows the encoding of this bit field
ipg_wait	Output	This output indicates that system are in the wait mode. This signal is simply a decoded and buffered version of the **ccm_lpmode[1:0]** inputs. Table 41-4, ***R-AHB Master Low Power Mode Encoding*** shows the encoding of this bit field
ipg_doze	Output	This output indicates for modules to enter low power operation. This signal is simply a decoded and buffered version of the **ccm_lpmode[1:0]** signals. Table 41-4, ***R-AHB Master Low Power Mode Encoding*** shows the encoding of this bit field
ipg_hard_async_reset_b	NA	This signal is not supported, the user should connect to the **hreset_b** input of the platform (which has the same behavior as **ipg_hard_async_reset_b**) to the **ipg_hard_async_reset_b** input to their module (if this input is present).

41.4 AIPI Registers

There are three registers that reside inside the AIPI. These registers occupy the IPbus peripheral location decoded when **haddr[16:12]** are all cleared, the IPbus peripheral location that would normally be assigned to **ips_module_en[0]** external to the AIPI. All three registers are 32-bit registers and can only be accessed in supervisor mode. Additionally, these registers can only be read from or written to by a 32-bit access.

The registers are shadowed throughout the space allocated by **ips_module_en[0]**.

Two system clocks are required for read accesses and three system clocks are required for write accesses to the AIPI registers.

41.4.1 Peripheral Size Registers[1:0]

These registers are used to tell the AIPI what size of IPbus peripheral is in each IPbus peripheral location. Peripheral locations that are not occupied should have their corresponding bits in the PSRs (Peripheral Size Registers) programmed to 1 in each register.

The least significant bit in the PSRs is a read only bit as it governs the AIPI registers themselves. They are set and cleared appropriately to indicate the registers are 32 bits.

PSR0	Peripheral Size Register 0															**Addr ips_module_en[0] + 0**
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

PSR1	Peripheral Size Register 1															**Addr ips_module_en[0] + 4**
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The PSRs work together to indicate the size of the IPbus peripheral occupying the corresponding **ips_module_en** location, or to indicate there is no IPbus peripheral occupying the corresponding **ips_module_en** location. A good example of how the PSRs work is the AIPI registers themselves. When **haddr[16:12]** is decoded to select the AIPI registers, {PSR1[bit0], PSR0[bit0]} returns a value of 10, indicating that the AIPI registers are word width registers. Table 41-6, *PSR Data Bus Size Encoding* shows how to program the PSR registers based on the size or availability of an IPbus peripheral.

Table 41-6. PSR Data Bus Size Encoding

PSR[1:0] bits		IPbus peripheral size[x] (ips_module_en[x])
PSR1[x]	PSR0[x]	
0	0	8-bit
0	1	16-bit
1	0	32-bit
1	1	Unoccupied

41.4.2 Peripheral Access Register

This register is used to tell the AIPI whether or not the IPbus peripheral corresponding to the bit location in this register may be accessed in user mode. If the peripheral may be accessed in supervisor mode only and a user mode access is attempted an abort will be generated and no IPbus activity will occur. If the peripheral may be accessed in user mode then the **ips_supervisor_access** bit will reflect whether the attempted access is in supervisor or user mode and the peripheral itself can decide whether to accept a user access (if one is attempted) or issue an error response.

The least significant bit in the PAR is a read only bit as it governs the AIPI registers themselves. It is set to indicate supervisor access only.

PAR	Peripheral Access Register[1]															**Addr ips_module_en[0] + 8**
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

1. A “1” indicates the corresponding peripheral is a supervisor access only peripheral. A “0” indicates the decision is left up to the peripheral (the AIPI will allow user accesses).

41.5 Interface Timings

41.5.1 Read Cycles

Two clock read accesses are possible with the AIPI when the requested access size is equal to or smaller than the size of the targeted IPbus peripheral. If the requested access size is larger than that of the targeted IPbus peripheral (i.e.,a 32 bit access to a 16 bit peripheral) then a minimum of three clocks are required to complete the access. The difference can be seen by examining Figure 41-2 and Figure 41-4.

In a read cycle all address, control signals and **ips_module_en** are launched off the same edge of **hclk** in which the AIPI becomes selected. The read cycle is terminated when **ips_xfr_wait** is negated at the rising edge of **hclk**. On the cycle following the negation of **ips_xfr_wait** the AIPI presents the valid data to the R-AHB and asserts **aipi_hready_out** to indicate to the R-AHB that the cycle has terminated.

When the AIPI is performing a multi-cycle access the **aipi_hready_out** signal will not be asserted until after the second or fourth (depending on whether a two cycle or four cycle access is required) **hclk** rising edge when **ips_xfr_wait** is negated, indicating that two successful read operations have occurred to the targeted IPbus peripheral. Please see Figure 41-4 and Figure 41-5 for clarification.

Once the AIPI asserts **aipi_hready_out** it remains asserted until the rising edge of **hclk** when **hready_in** is asserted.

41.5.2 Write Cycles

Three clock write accesses are possible with the AIPI when the requested access size is equal to or smaller than the size of the targeted IPbus peripheral. If the requested access size is larger than that of the targeted IPbus peripheral (i.e., a 32 bit access to a 16 bit peripheral) then a minimum of four clocks are required to complete the access. The difference can be seen by examining Figure 41-6 and Figure 41-8.

In a write cycle all address and control signals are launched off the same edge of **hclk**, which is the same edge of **hclk** in which the AIPI becomes selected. **ips_wdata** and **ips_module_en** are launched off the following edge of **hclk**, insuring excellent write data timing to the targeted IPbus peripheral. The write cycle is terminated when **ips_xfr_wait** is negated at the rising edge of **hclk**. On the cycle following the negation of **ips_xfr_wait** the AIPI asserts **aipi_hready_out** to indicate to the R-AHB that the cycle has terminated.

When the AIPI is performing a multi-cycle access the **aipi_hready_out** signal will not be asserted until after the second **hclk** rising edge when **ips_xfr_wait** is negated, indicating that two successful write operations have occurred to the targeted IPbus peripheral.

Once the AIPI asserts **aipi_hready_out** it remains asserted until the rising edge of **hclk** when **hready_in** is asserted.

41.5.3 Aborted Cycles

When a cycle is aborted and the abort is initiated by the AIPI itself or the targeted IPbus peripheral the AIPI will follow a standard procedure. The AIPI will either fail to initiate or will immediately terminate any IPbus activity that is ongoing. The AIPI will then assert **aipi_hresp0** (indicating an error has occurred), the following clock cycle the AIPI will assert **aipi_hready_out** and the AIPI will hold both **aipi_hresp0** and **aipi_hready_out** asserted until **hready_in** is asserted, at which time the AIPI will negate both **aipi_hresp0** and **aipi_hready_out**.

There are several conditions which will cause the AIPI to abort the current operation and report an error. The first is the case in which the targeted IPbus peripheral asserts **ips_xfr_err** while **ips_xfr_wait** is negated. In this case the AIPI will immediately terminate access to the targeted IPbus peripheral and follow the abort procedure described above. Whether or not the current IPbus access is a multi-cycle access or a single cycle access has no bearing on the behavior of the AIPI. The AIPI will respond identically in both cases. Please refer to Figure 41-9, Figure 41-10 and Figure 41-11 for additional clarification.

The second case that will cause an error response to the R-AHB is when a user mode accesses is attempted to an IPbus peripheral who's corresponding PAR bit indicates it is a supervisor only peripheral. In this case the AIPI will not initiate any IPbus activity but will instead respond immediately by following the abort procedure described above. This can be seen in Figure 41-12.

The third case that will cause an error response to the R-AHB is when an access is attempted to a location at which the PSRs indicate there is no IPbus peripheral. In this case the AIPI will not initiate any IPbus activity but will instead respond immediately by following the abort procedure described above. This can be seen in Figure 41-12.

The AIPI will also terminate the current IPbus peripheral access activity if **hready_in** is asserted before the end of the current IPbus access. This will not result in an error condition being reported as this behavior is initiated by the R-AHB. If the R-AHB is once again requesting access to an IPbus peripheral a new access to the targeted peripheral will be initiated, otherwise all IPbus activity will terminate.

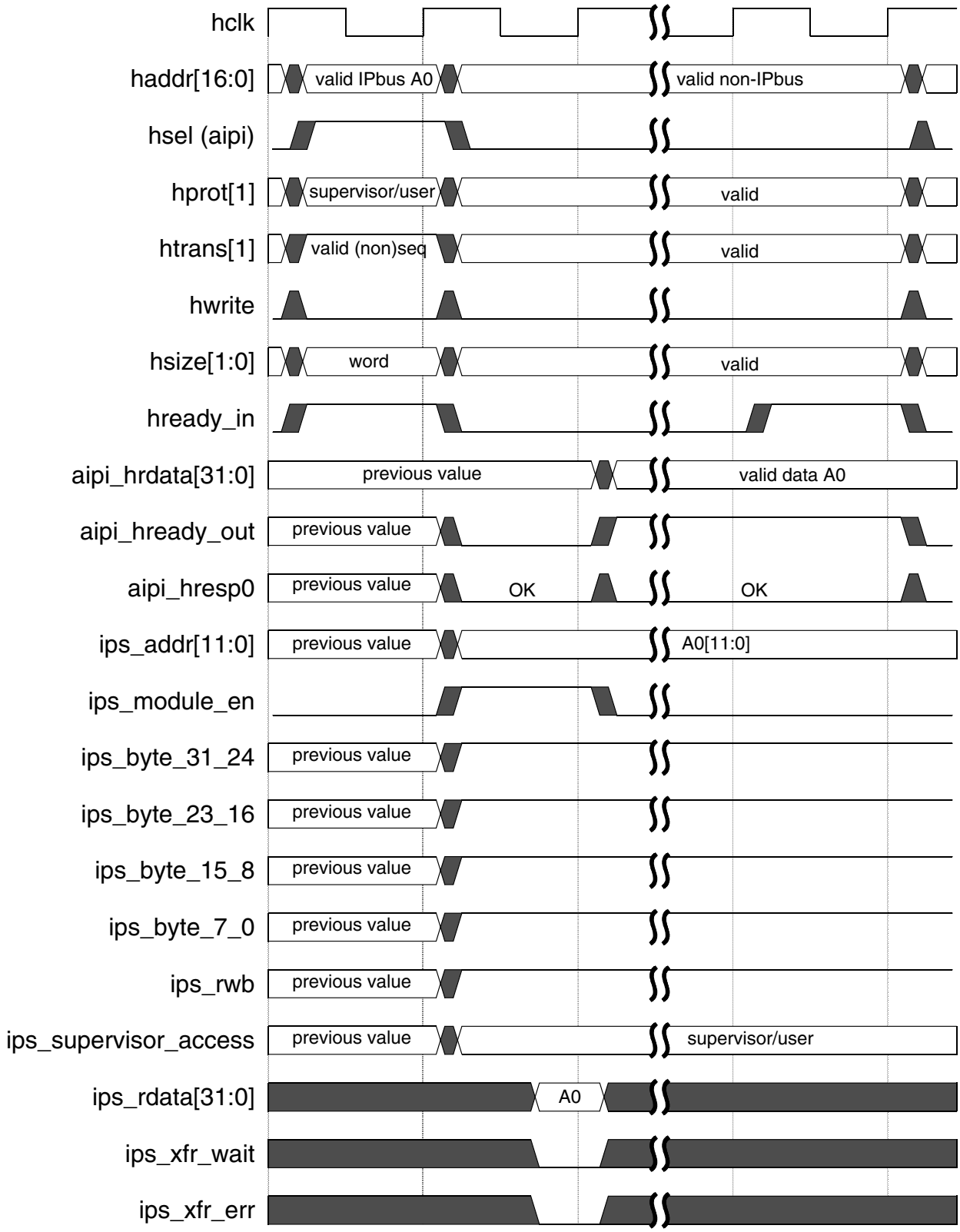


Figure 41-2. Two clock read access (32-bit IPbus)

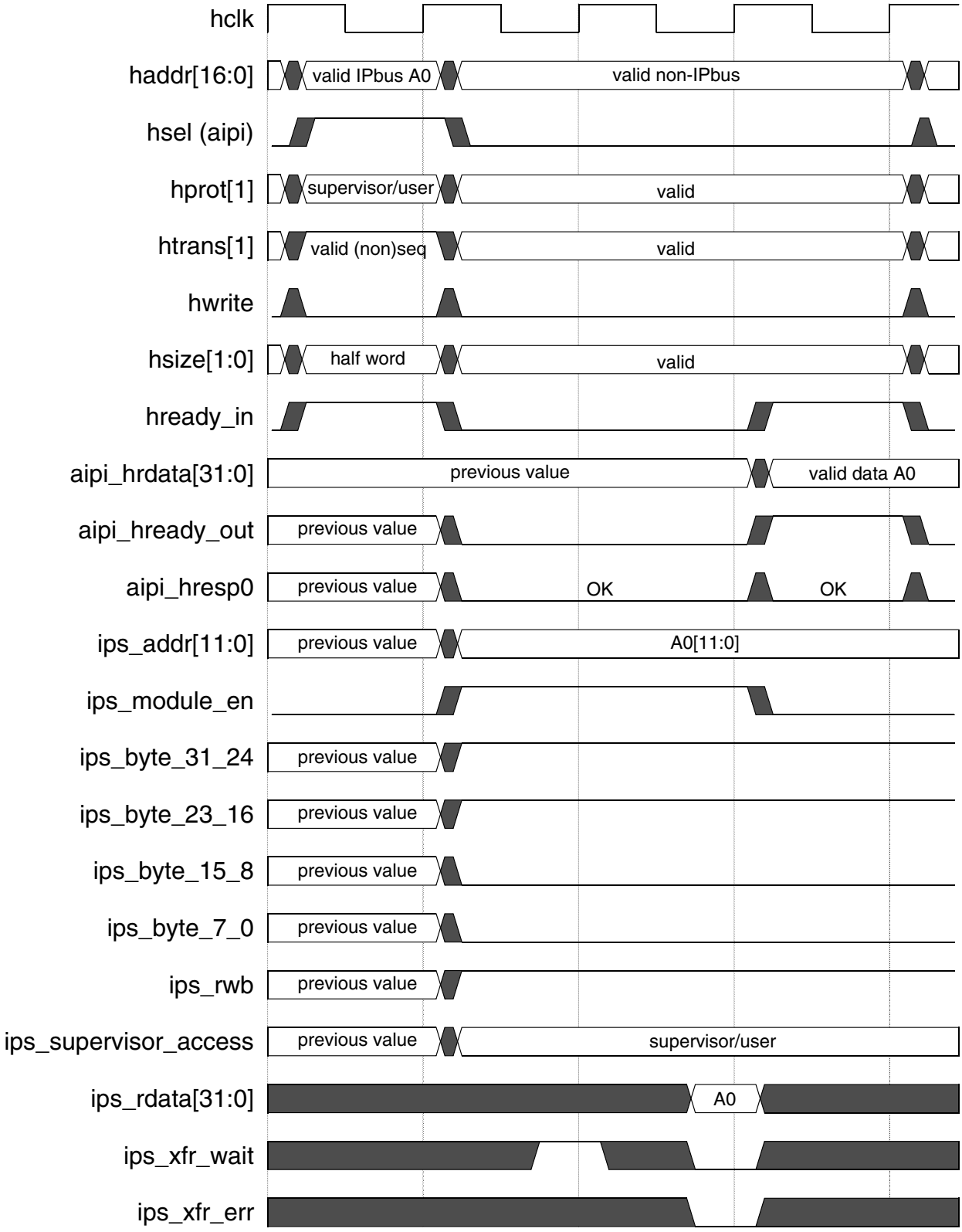


Figure 41-3. Three clock read access (32-bit IPbus)

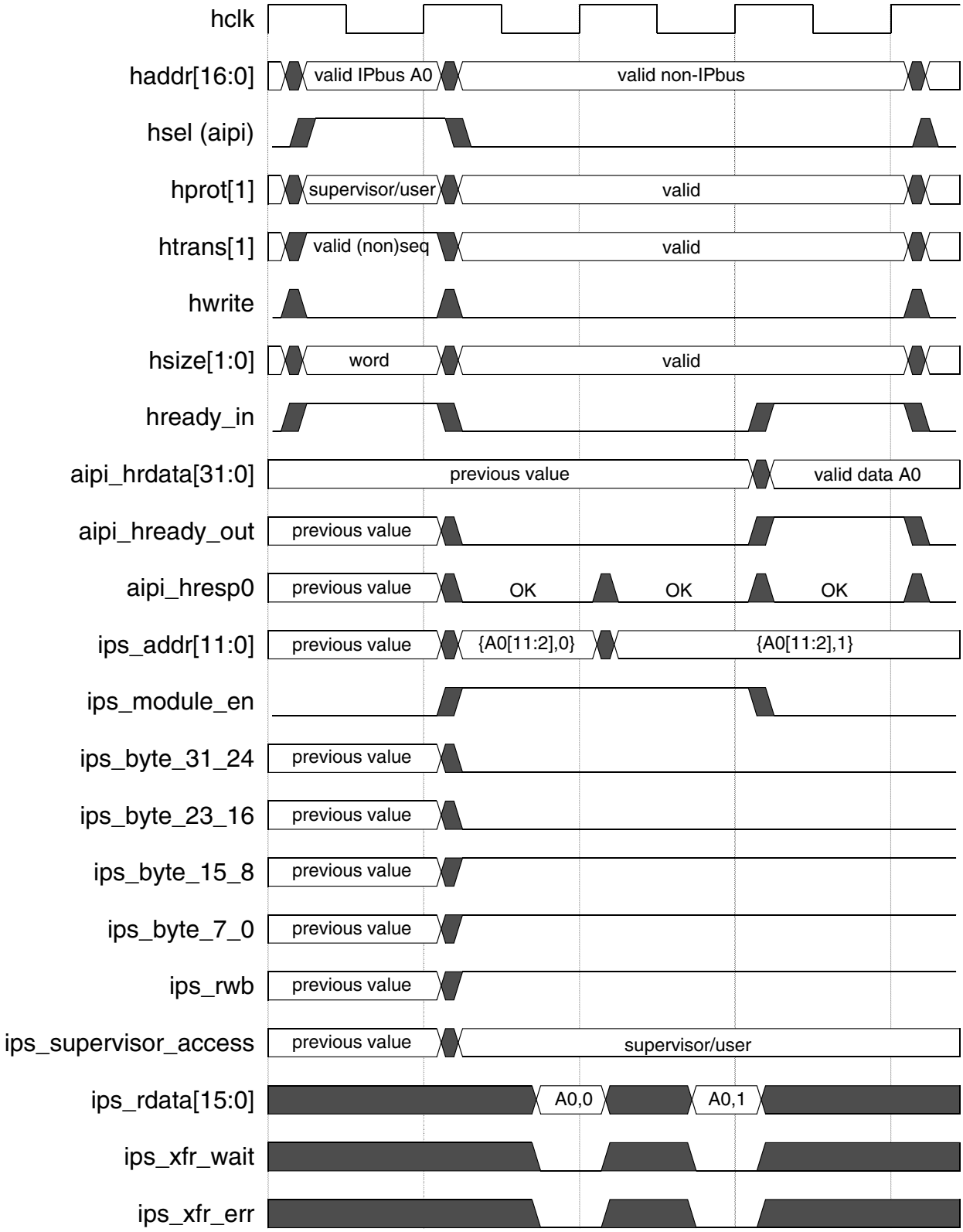


Figure 41-4. Three clock read access (16-bit IPbus)

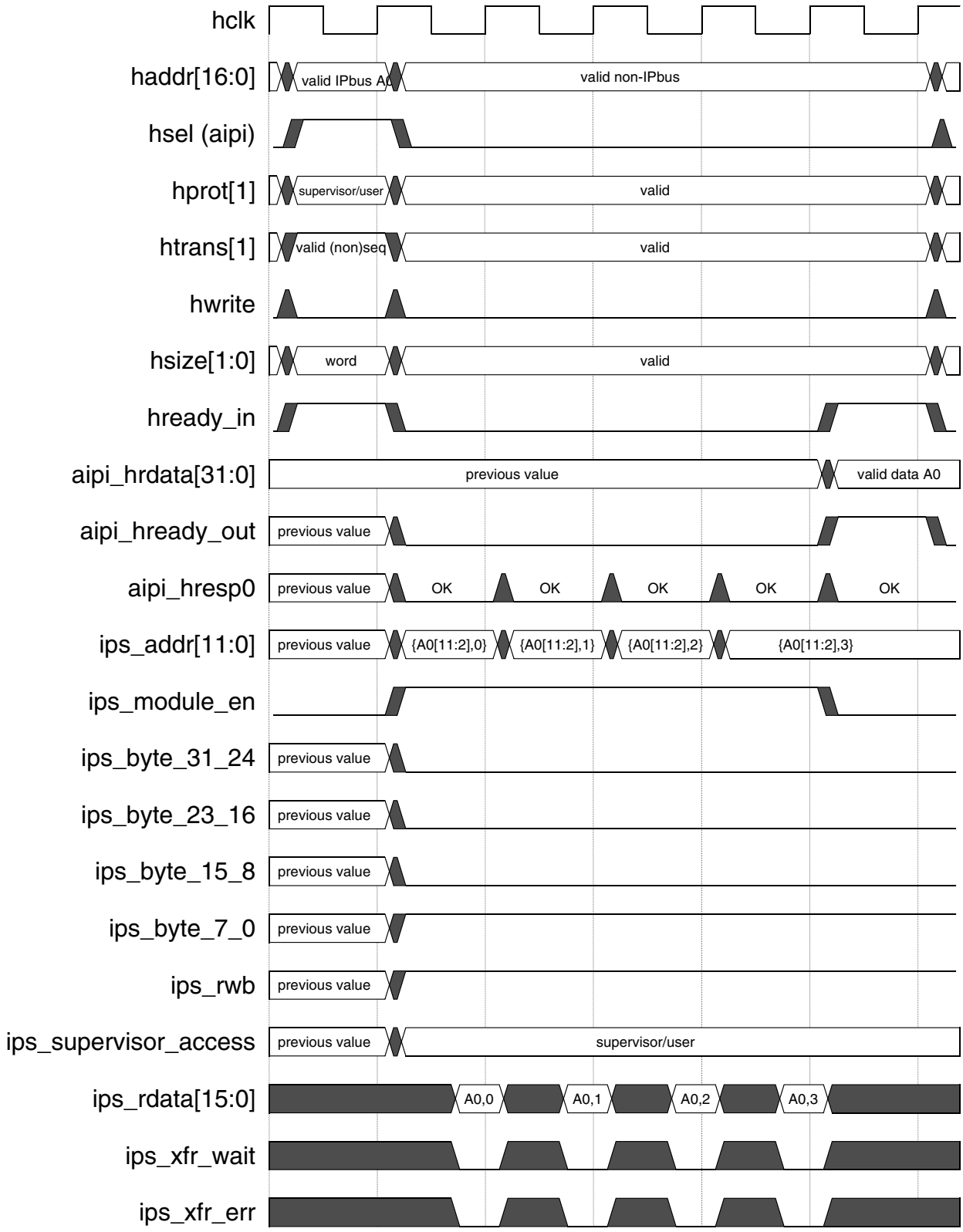


Figure 41-5. Five clock read access (8-bit IPbus)

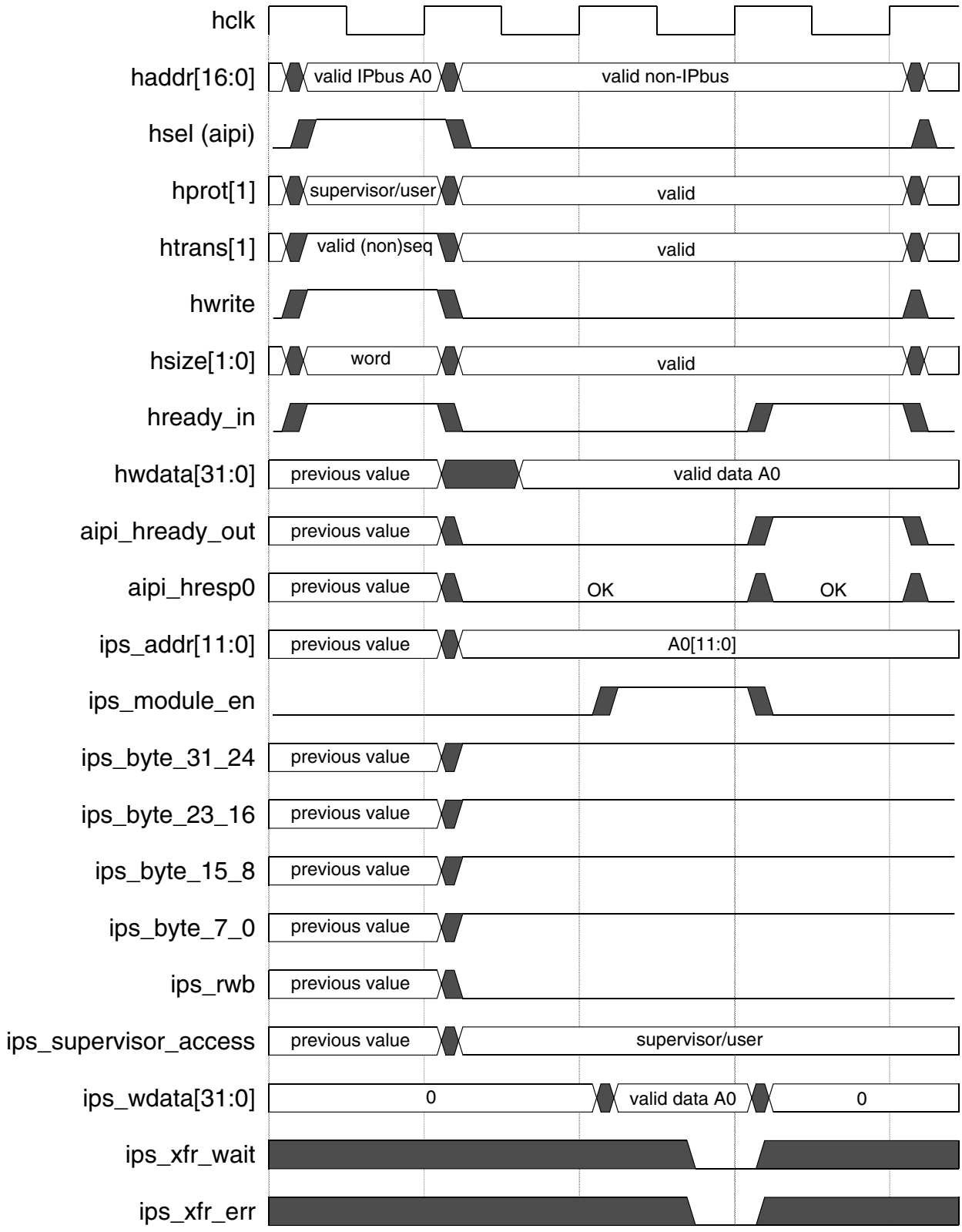


Figure 41-6. Three clock write access (32-bit IPbus)

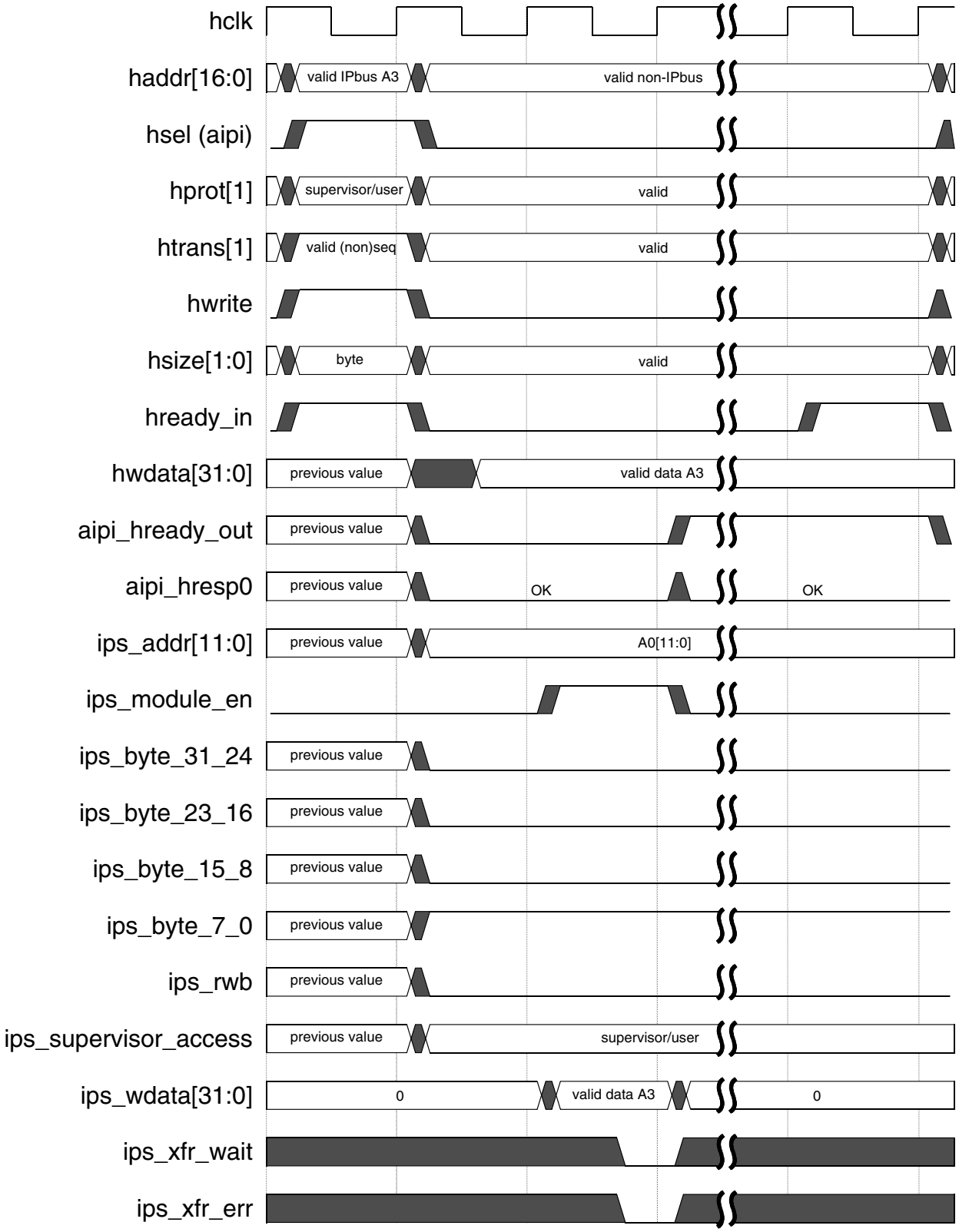


Figure 41-7. Three clock write access (16-bit/32-bit IPbus)

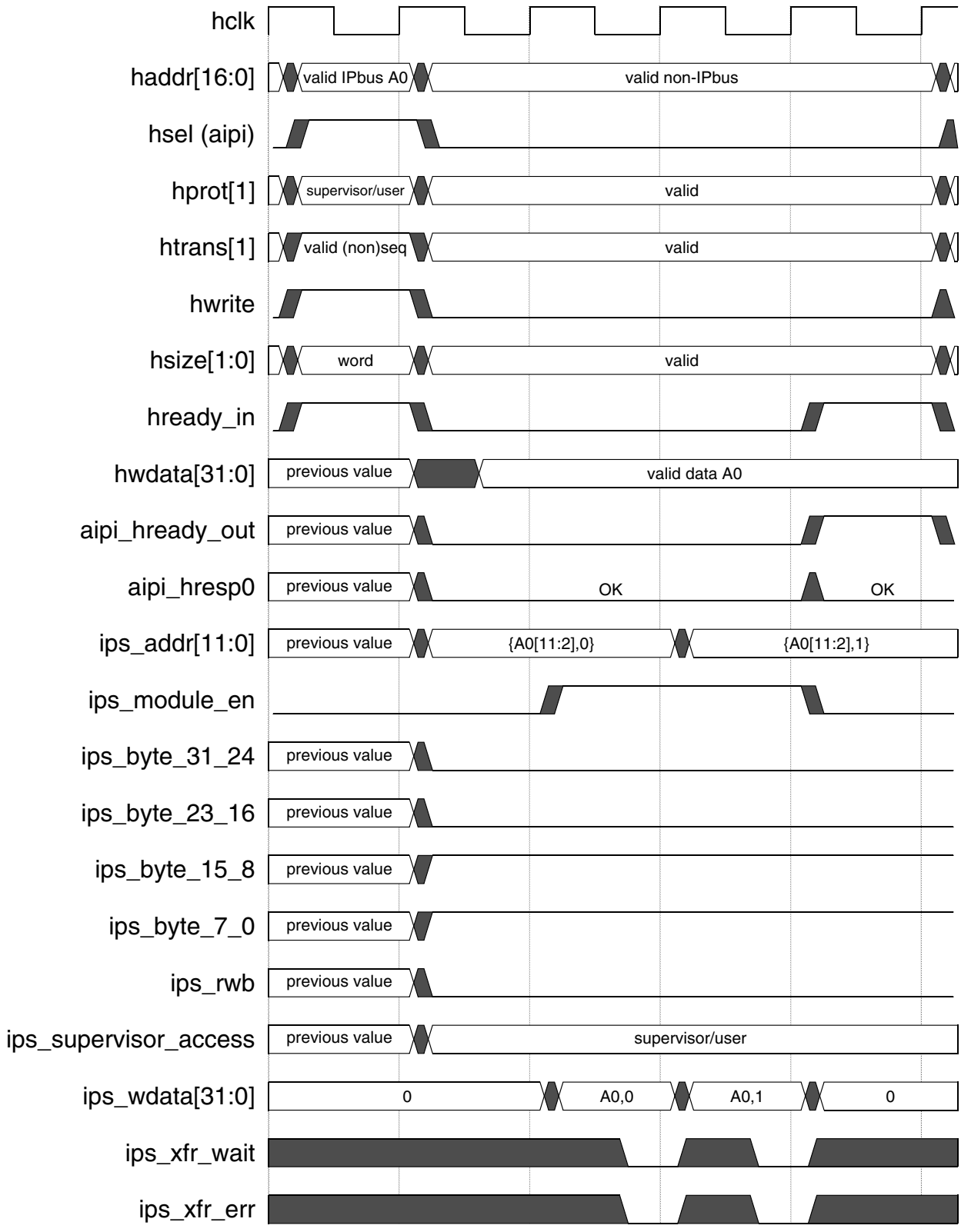


Figure 41-8. Four clock write access (16-bit IPbus)

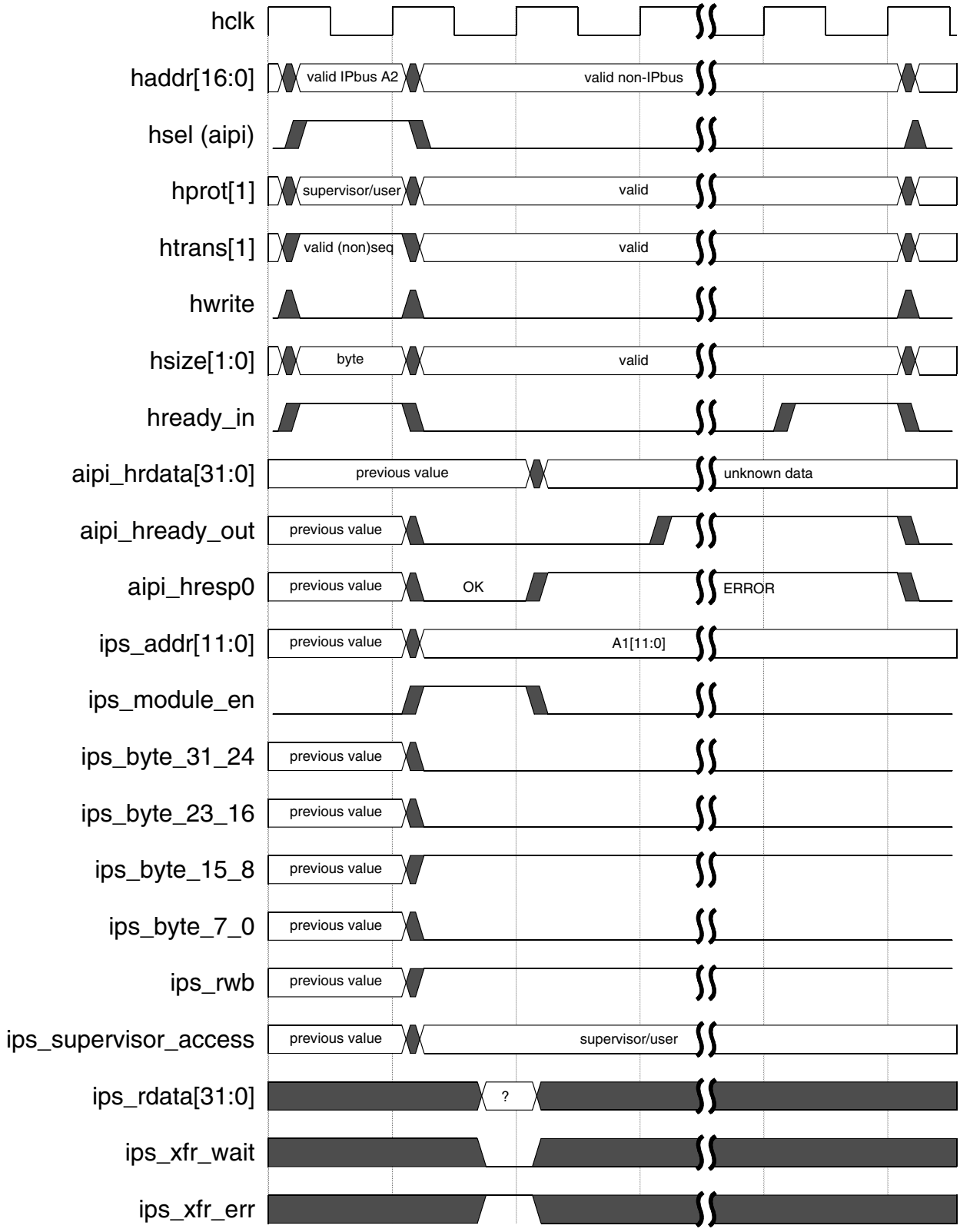


Figure 41-9. Two clock read with error (16-bit/32-bit IPbus)

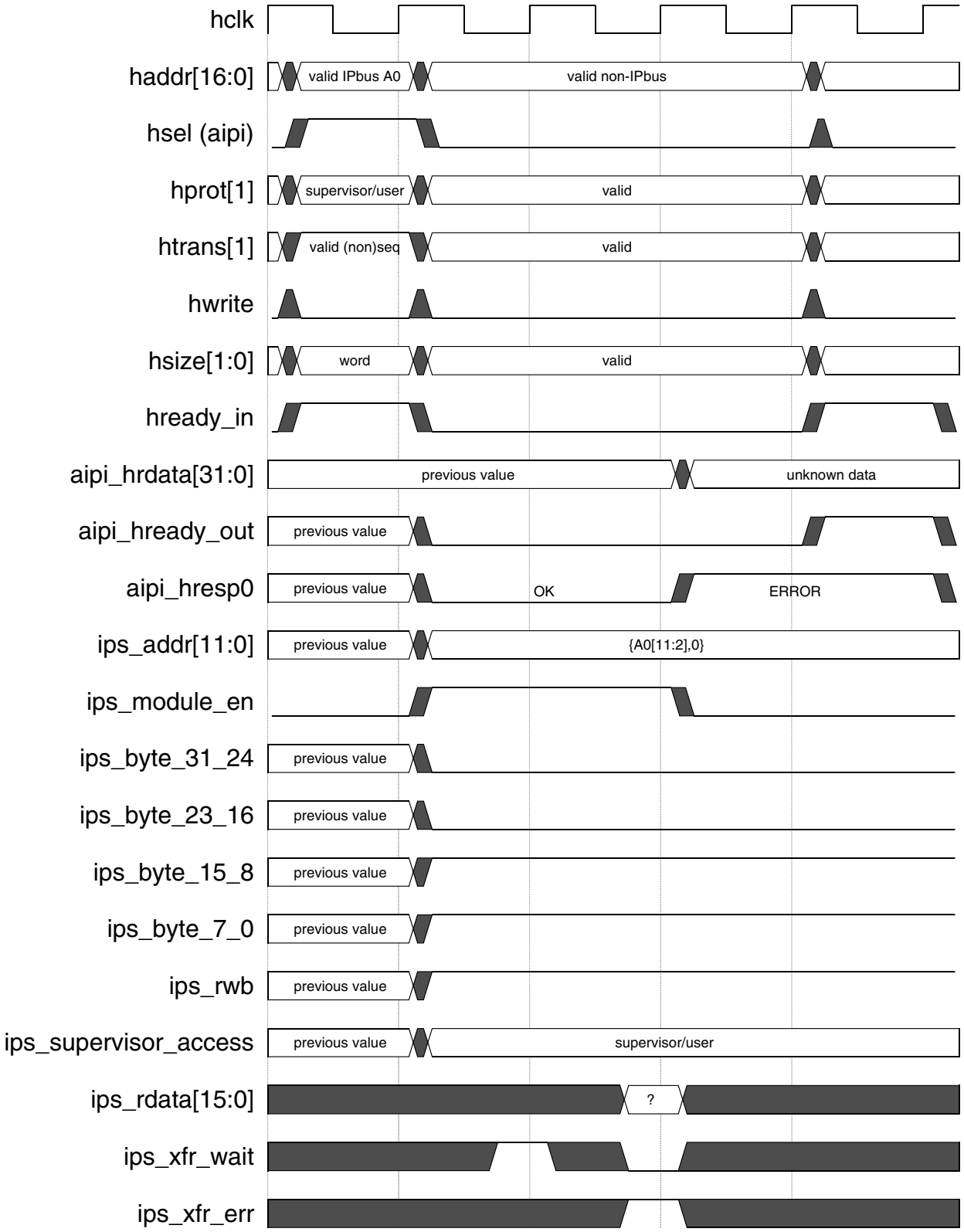


Figure 41-10. Three clock read with error (16-bit IPbus)

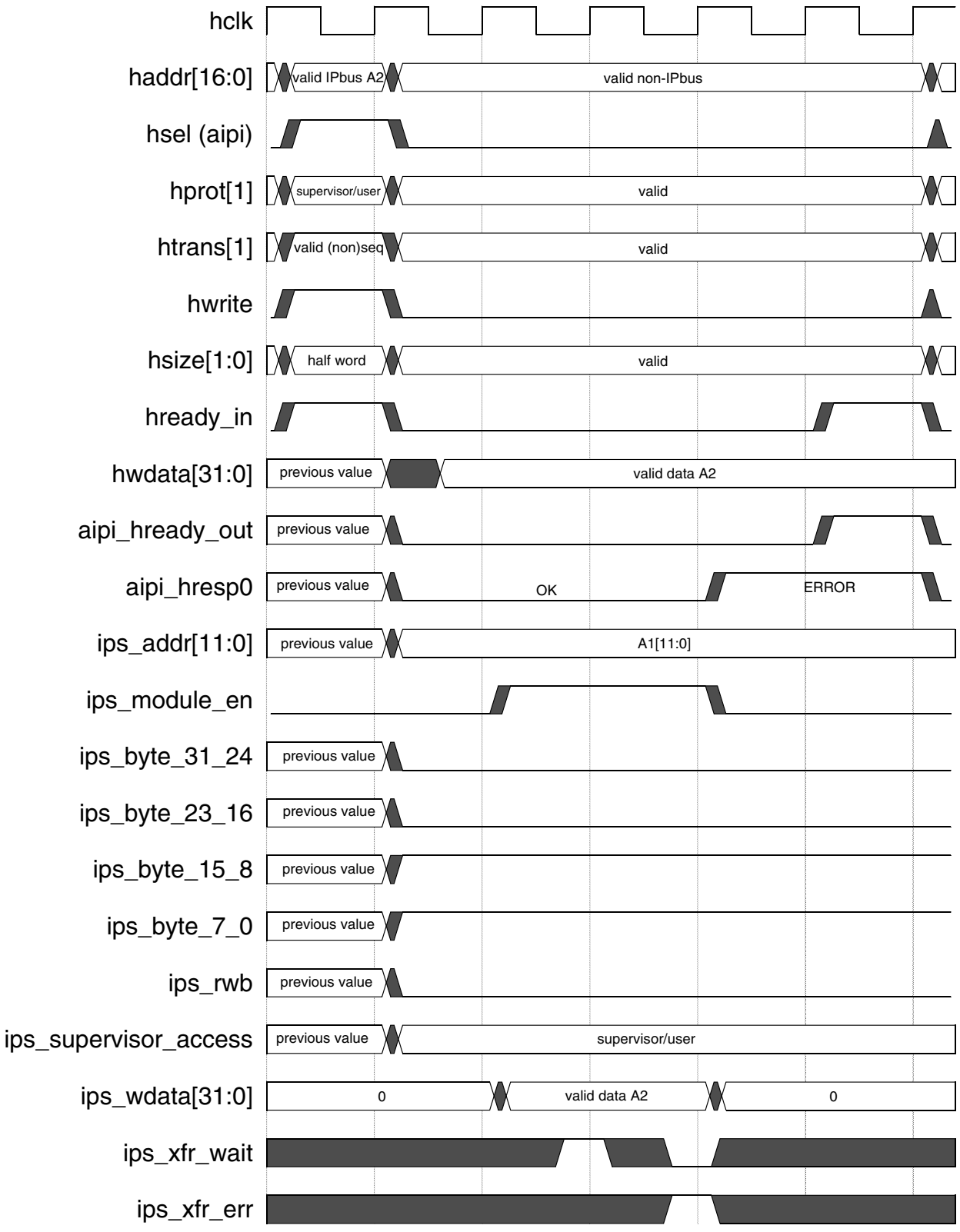


Figure 41-11. Four clock write with error (16-bit/32-bit IPbus)

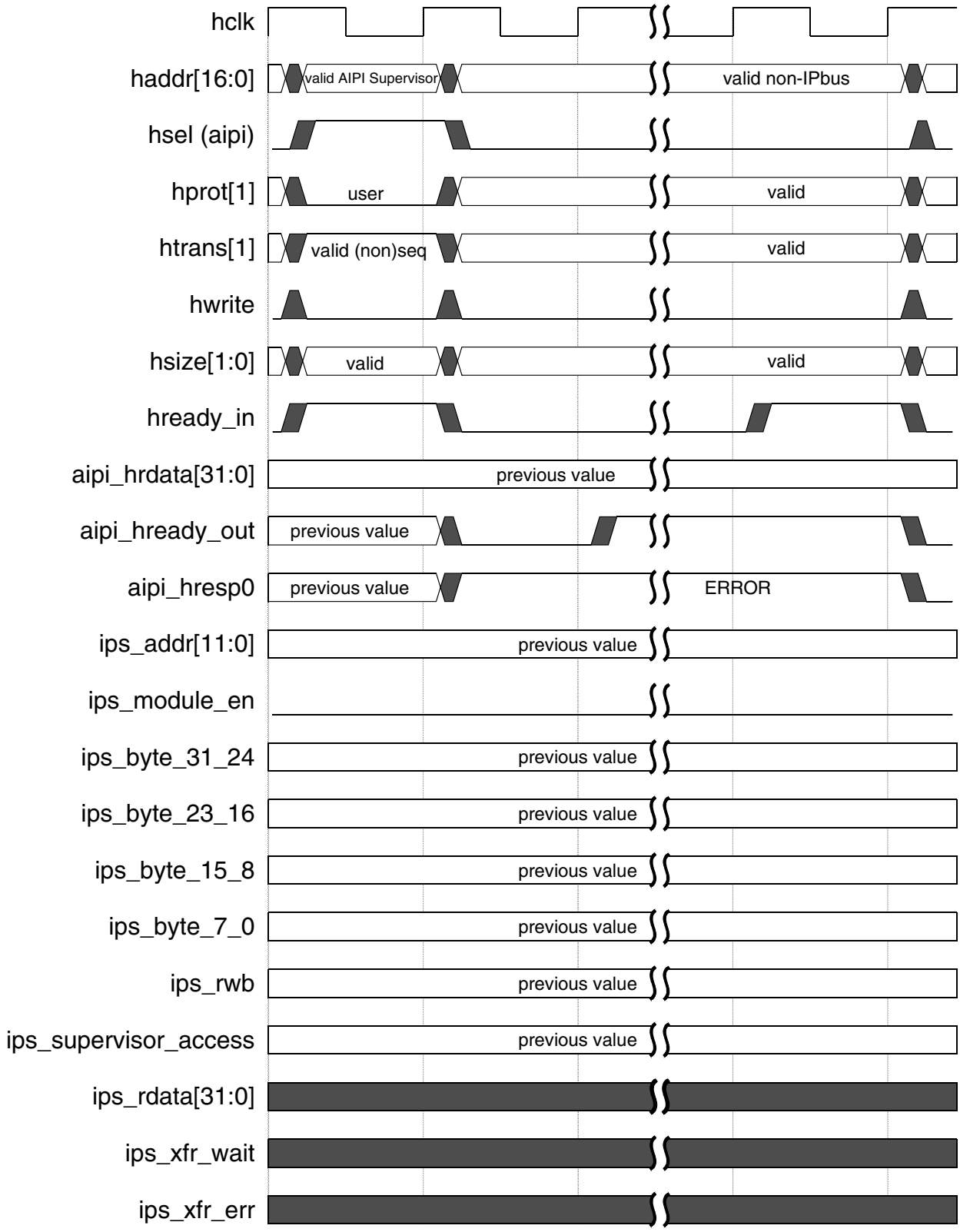


Figure 41-12. AIPI access protection/occupation error

41.6 AIPI AC Timing Information

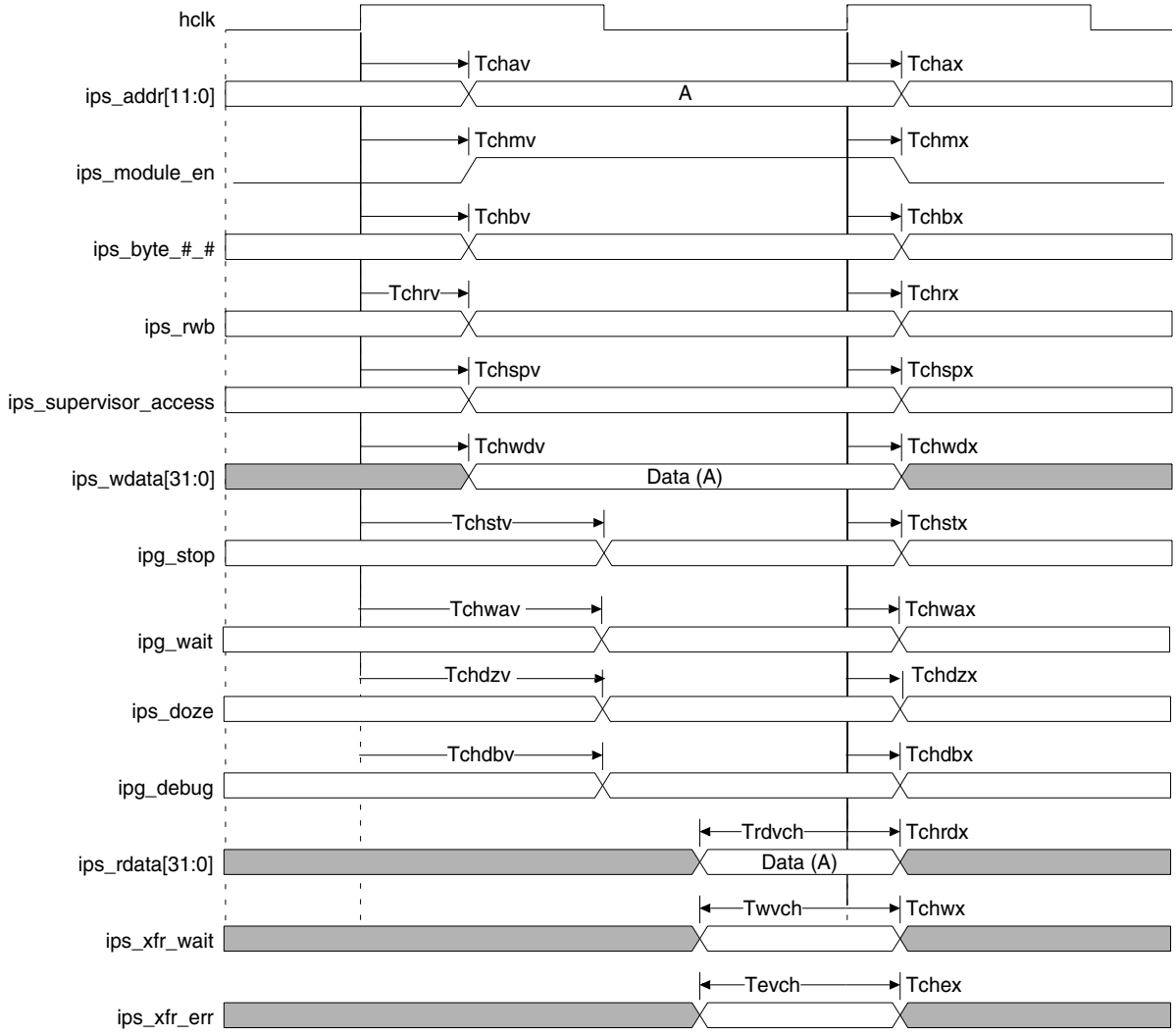


Figure 41-13. AIPI IPbus v2.0 Bus Timing Parameters

Table 41-7. AIPI IPbus v2.0 AC Timing Parameters[1]

Parameter	Description	Timing (ns)
Tclk	hclk minimum period	18.0
Tchav	hclk rise to ips_addr valid	TBD
Tchax	ips_addr hold from hclk rise	TBD
Tchmv	hclk rise to ips_module_en valid	TBD
Tchmx	ips_module_en hold from hclk rise	TBD
Tchbv	hclk rise to ips_byte_#_# valid	TBD
Tchbx	ips_byte_#_# hold from hclk rise	TBD
Tchrv	hclk rise to ips_rwb valid	TBD
Tchrx	ips_rwb hold from hclk rise	TBD
Tchspv	hclk rise to ips_supervisor_access valid	TBD
Tchspx	ips_supervisor_access hold from hclk rise	TBD
Tchwdv	hclk rise to ips_wdata valid	TBD
Tchwdx	ips_wdata valid hold from hclk rise	TBD
Tchstv	hclk rise to ipg_stop valid	TBD
Tchstx	ipg_stop hold from hclk rise	TBD
Tchwav	hclk rise to ipg_wait valid	TBD
Tchwax	ipg_wait hold from hclk rise	TBD
Tchdzv	hclk rise to ipg_doze valid	TBD
Tchdzx	ipg_doze hold from hclk rise	TBD
Tchdbv	hclk rise to ipg_debug valid	TBD
Tchdbx	ipg_debug hold from hclk rise	TBD
Trdvch	ips_rdata setup to hclk rise	TBD
Tchrdx	ips_rdata hold from hclk rise	TBD
Twvch	ips_xfr_wait setup to hclk rise	TBD
Tchwx	ips_xfr_wait hold from hclk rise	TBD
Tevch	ips_xfr_err setup to hclk rise	TBD
Tchex	ips_xfr_err hold from hclk rise	TBD

1. The requirements for AC timing from targeted IPbus v2.0 modules can be found in the IPbus v2.0 specification. If targeted modules meet the IPbus v2.0 timing specification they will work with the AIPI. However, it would be preferable for all IPbus modules to try and meet IPbus v3.0 (proposed) AC timing.

Chapter 42
Alternate Master Arbiter (AMARB)

Revision History Table

Revision	Date	Author	Changes
0.0	10/01/01	Rob Greenwood and Shannon Osgood	Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
1.0	2/25/02	Mindy Hsu	Added HAC as alternate master
1.1	3/29/02	Mindy Hsu	Added another bit and changed amarb_master encoding scheme
1.2	4/18/02	Mindy Hsu	Changed Scan Signal Name and update state machine and timing diagram for 2 clock cycle delays before sampling again
1.3	05/07/03		Updated for LTE specification release.

42.1 Introduction

The external alternate master arbiter allows multiple alternate masters, such as the DMAC, GEM, TCM, or HACC to gain control of the ARM7 high performance (R-AHB) bus. The alternate master arbiter is also used during low power modes to gate the clock to the ARM7 core. Figure 42-1 on page 42-3 shows a block diagram of external arbiter and its connections in the system. The alternate master arbiter interfaces with the ARM7 platform core bus arbiter (CARB). It muxes the address, attributes, and write data of the alternate masters and outputs these signals to the ARM7 platform. Inside the ARM7 platform, these signals are muxed with the ARM7 address, attribute, and write data signals. The muxes in the ARM7 platform are controlled by the CARB. Read data for the alternate masters comes directly from the ARM7 platform.

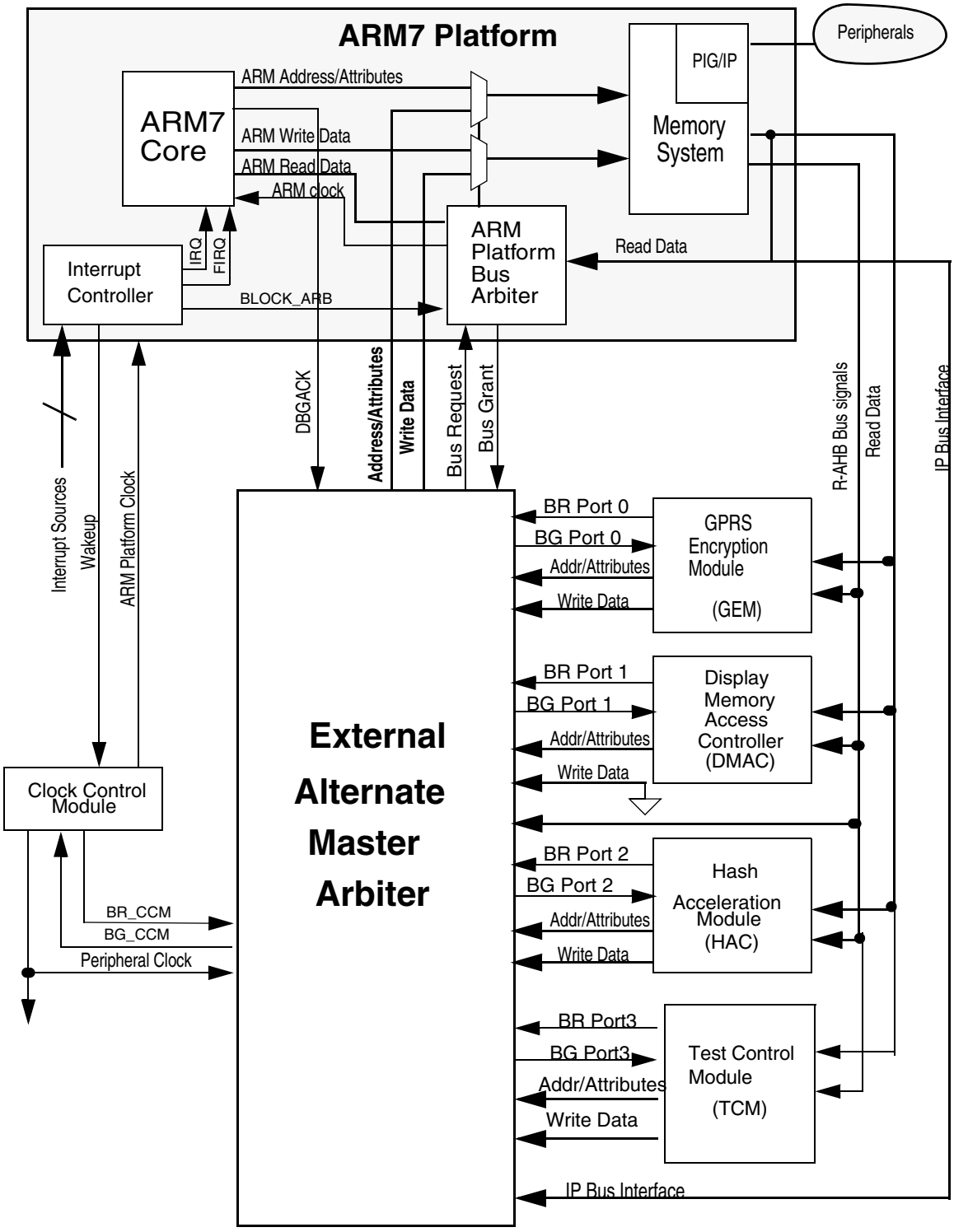


Figure 42-1. External Arbiter Block Diagram

42.2 AMARB Module Pin List

Table 42-1 is a list of the AMARB module pins.

Table 42-1. AMARB Module Pin List

Pin Name	Direction	Description
br_port0	Input	Port 0 Bus Request
br_port1	Input	Port 1 Bus Request
br_port2	Input	Port 2 Bus Request
br_port3	Input	Port3 Bus Request
br_ccm	Input	Clock Control Module Bus Request
wdata_port0[31:0]	Input	Port 0 Write Data Bus
wdata_port1[31:0]	Input	Port 1 Write Data Bus
wdata_port2[31:0]	Input	Port 2 Write Data Bus
wdata_port3[31:0]	Input	Port3 Write Data Bus
addr_port0[31:0]	Input	Port 0 Address Bus
addr_port1[31:0]	Input	Port 1Address Bus
addr_port2[31:0]	Input	Port 2 Address Bus
addr_port3[31:0]	Input	port3 Address Bus
size_port0[1:0]	Input	Port 0 Size Attribute
size_port1[1:0]	Input	Port 1Size Attribute
size_port2[1:0]	Input	Port 2 Size Attribute
size_port3[1:0]	Input	Port 3 Size Attribute
trans_port0[1:0]	Input	Port 0 Transfer Type Attribute
trans_port1[1:0]	Input	Port 1 Transfer Type Attribute
trans_port2[1:0]	Input	Port 2 Transfer Type Attribute
trans_port3[:0]	Input	Port 3 Transfer Type Attribute

Table 42-1. AMARB Module Pin List (Continued)

Pin Name	Direction	Description
prot_port0[1:0]	Input	Port 0 Protection Attribute
prot_port1[1:0]	Input	Port 1Protection Attribute
prot_port2[1:0]	Input	Port 2 Protection Attribute
Prot_port3[1:0]	Input	Port 3 Protection Attribute
write_port0	Input	Port 0 Write Signal
write_port1	Input	Port 1 Write Signal
write_port2	Input	Port 2 Write Signal
write_port3	Input	Port 3 Write Signal
bg_port0	Output	Bus Grant Signal Output to Port 0
bg_port1	Output	Bus Grant Signal Output to Port 1
bg_port2	Output	Bus Grant Signal Output to Port 2
bg_port3	Output	Bus Grant Signal Ouput to Port3
bg_ccm	Output	Bus Grant Signal Output to the Clock Control Module
AMARB Interface Signals		
carb_bg	Input	Bus Grant from ARM7 Platform Core Arbiter
hready	Input	HREADY signal from ARM7 Platform memory system
dbgack	Input	Debug acknowledge from ARM7 platform
amarb_br	Output	Bus Request Signal from Alternate Master Arbiter to Core Arbiter
amarb_haddr[31:0]	Output	Alternate Master Arbiter Address Output to Core Arbiter
amarb_hwdata[31:0]	Output	Alternate Master Arbiter Write Data Output to Core Arbiter
amarb_hsize[1:0]	Output	Alternate Master Arbiter Size Attribute Output to Core Arbiter
amarb_htrans[1:0]	Output	Alternate Master Arbiter Transfer Attribute Output to Core Arbiter

Table 42-1. AMARB Module Pin List (Continued)

Pin Name	Direction	Description
amarb_hprot[1:0]	Output	Alternate Master Arbiter Protection Attribute Output to Core Arbiter
amarb_hwrite	Output	Alternate Master Arbiter Write Signal Output to Core Arbiter
amarb_master[2:0]	Output	Alternate Master Arbiter state used for AEIM show cycles mode
IP BUS Interface Signals		
ips_wdata[7:0]	Input	Write data bus
ips_module_en	Input	Module select
ips_addr[4:2]	Input	Address bus
ips_byte_7_0	Input	Byte [7:0] enable
ips_rwb	Input	Module read/write signal
ips_supervisor_access	Input	Supervisor access input
ips_rdata[7:0]	Output	16-bit read data bus from arbiter
ips_xfr_wait	Output	Arbiter wait output
ips_xfr_err	Output	Arbiter transfer error output
Clocking/Scan Signals		
clk	Input	System Peripheral Clock from Clock Control Module
reset_b	Input	System Reset Signal
ipt_clk_se	Input	Scan enable

42.3 AMARB Programming Model

42.3.1 Register Summary

The AMARB module has six registers: a control register (ARB_CONTROL), an interval control register (ARB_INTERVAL), a software priority level control registers (SOFT_PRIORITY_LEVEL), and three port priority level control registers (PORT0_PRIORITY_LEVEL, PORT1_PRIORITY_LEVEL, and PORT2_PRIORITY_LEVEL). Each of the registers is described below.

Figure 0-6.

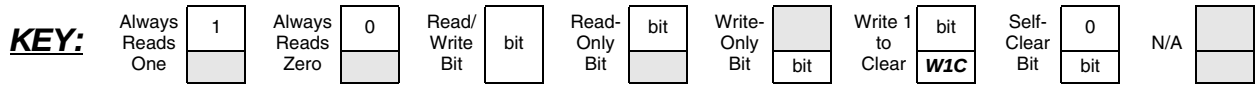


Table 42-2. AMARB Register Summary

Figure 0-7.

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARB_CONTROL ($2484_7000)	R	0																															ARB_EN
	W																																
ARB_INTERVAL ($2484_7004)	R	0																								INTERVAL[7:0]							
SOFT_PRIORITY_LEVEL ($2484_7008)	R	0																											SOFT_PRIORITY [4:0]				
	W																																
PORT0_PRIORITY_LEVEL ($2484_700c)	R	0																								P0_DIS_DBUG	0		PORT0_PRIORITY [4:0]				
	W																																
PORT1_PRIORITY_LEVEL ($2484_7010)	R	0																								P1_DIS_DBUG	0		PORT1_PRIORITY [4:0]				
	W																																
PORT2_PRIORITY_LEVEL ($2484_7014)	R	0																								P2_DIS_DBUG	0		PORT2_PRIORITY [4:0]				
	W																																
PORT3_PRIORITY_LEVEL ($2484_7018)	R	0																								P3_DIS_DBUG	0		PORT3_PRIORITY [4:0]				
	W																																

42.3.2 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various AMARB registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

ARB_CONTROL Arbiter Control Register **$2484_7000**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
																ARB_EN
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 42-3. Arbiter Control Register Description

Name	Description	Settings
ARB_EN Bit 0	**Arbiter Enable—** This bit enables the external alternate master arbiter. This bit is set out of reset. When cleared, the clock to the arbiter is gated off and no alternate master will be granted access to the R-AHB system bus. Do not disable the AMARB while the alternate masters are busy and have the potential to request control of the bus. Throttling of the alternate master access activity should be done using the priority and interval controls built into the AMARB.	0 = Arbiter disabled 1 = Arbiter enabled

NOTE:

Alternate master devices should not clear the ARB_EN bit of the Arbiter Control Register. This will disable the AMARB state machine which could lead to unpredictable system behavior.

ARB_INTERVAL Arbiter Interval Register **$2484_7004**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									INTERVAL[7:0]							
TYPE:	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 42-4. Arbiter Interval Register Description

Name	Description	Settings
INTERVAL [7:0] Bits 7-0	**Bus Handoff Interval—** The bus handoff interval bits specify the minimum number of clock cycles that the arbiter will wait before evaluating the next alternate master's bus request signal. This pause occurs at the conclusion of a bus handoff event. This feature can be used to allow the core to have control of the system bus for a minimum number of cycles between alternate master accesses.	0 = Wait 0 additional cycles 1 = Wait 1 additional cycle . . 255 = Wait 255 additional cycles

The Software Priority Level Register sets the priority for the current software context.

SOFT_PRIORITY_LEVEL Software Priority Level Register **$2484_7008**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
												SOFT_PRIORITY[4:0]				
TYPE:	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 42-5. Software Priority Level Register Description

Name	Description	Settings
SOFT_PRIORITY[4:0] Bits 4-0	**Software Priority Level**—These bits set the priority level of the current software process. The priority level of a requesting alternate master must be greater than that of the software priority in order to be granted control of the system bus.	00000 – Current software process has lowest priority 00001 . . . 11110 11111 - Current software process has highest priority

PORT0_PRIORITY_LEVEL — Port 0 Priority Level Register — $2484_700c

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									P0_DIS_DBUG			PORT0_PRIORITY[4:0]				
TYPE:	r	r	r	r	r	r	r	r	rw	r	r	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 42-6. Port 0 Priority Level Register Description

Name	Description	Settings
P0_DIS_DBUG Bit 7	**Port 0 Debug Disable** —When set, This bit disables alternate master acesses from Port 0 when DBGACK is asserted by the ARM7 core. Assertion of DBGACK will not terminate alternate master activity that has already initiated. It will, however, prevent new Port 0 bus requests when DBGACK=1.	0 – Port 0 alternate master accesses not affected by DBGACK 1 - Port 0 alternate master bus requests are disabled when DBGACK is asserted
PORT0_PRIORITY[4:0] Bits 4-0	**Port 0 Priority Level**—These bits set the priority level of the alternate master connected to port 0. The priority level of a requesting alternate master must be greater than that of the software priority in order to be granted control of the system bus.	00000 – Port 0 alternate master has its lowest priority and will never be selected 00001 . . . 11110 11111 - Port 0 alternate master has its highest priority

PORT1_PRIORITY_LEVEL — Port 1 Priority Level Register — $2484_7010

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									P1_DIS_DBG			PORT1_PRIORITY[4:0]				
TYPE:	r	r	r	r	r	r	r	r	rw	r	r	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 42-7. Port 1 Priority Level Register Description

Name	Description	Settings
P1_DIS_DBUG Bit 7	**Port 1 Debug Disable** —When set, this bit disables alternate master acesses from Port 1 when DBGACK is asserted by the ARM7 core. Assertion of DBGACK will not terminate alternate master activity that has already initiated. It will, however, prevent new Port 1 bus requests when DBGACK=1.	0 – Port 1 alternate master accesses not affected by DBGACK 1 - Port 1 alternate master bus requests are disabled when DBGACK is asserted
PORT1_PRIORITY[4:0] Bits 4-0	**Port 1 Priority Level**—These bits set the priority level of the alternate master connected to port 1. The priority level of a requesting alternate master must be greater than that of the software priority in order to be granted control of the system bus.	00000 – Port 1 alternate master has its lowest priority and will never be selected 00001 . . . 11110 11111 - Port 1 alternate master has its highest priority

PORT2_PRIORITY_LEVEL — Port 2 Priority Level Register — $2484_7014

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									P2_DIS_DBG			PORT2_PRIORITY[4:0]				
TYPE:	r	r	r	r	r	r	r	r	rw	r	r	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 42-8. Port 2 Priority Level Register Description

Name	Description	Settings
P2_DIS_DBUG Bit 7	**Port 2 Debug Disable** —When set, this bit disables alternate master acesses from Port 2 when DBGACK is asserted by the ARM7 core. Assertion of DBGACK will not terminate alternate master activity that has already initiated. It will, however, prevent new Port 2 bus requests when DBGACK=1.	0 – Port 2 alternate master accesses not affected by DBGACK 1 - Port 2 alternate master bus requests are disabled when DBGACK is asserted
PORT2_PRIORITY[4:0] Bits 4-0	**Port 2 Priority Level**—These bits set the priority level of the alternate master connected to port 2. The priority level of a requesting alternate master must be greater than that of the software priority in order to be granted control of the system bus.	00000 – Port 2 alternate master has its lowest priority and will never be selected 00001 . . . 11110 11111 - Port 2 alternate master has its highest priority

PORT3_PRIORITY_LEVEL — Port 3Priority Level Register — $2484_7018

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									P3_DIS_DBG			PORT3_PRIORITY[4:0]				
TYPE:	r	r	r	r	r	r	r	r	rw	r	r	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 42-9. Port 3Priority Level Register Description

Name	Description	Settings
P3_DIS_DBUG Bit 7	**Port 3Debug Disable** —When set, this bit disables alternate master acesses from Port 3 when DBGACK is asserted by the ARM7 core. Assertion of DBGACK will not terminate alternate master activity that has already initiated. It will, however, prevent new Port 3bus requests when DBGACK=1.	0 – Port 3alternate master accesses not affected by DBGACK 1 - Port 3alternate master bus requests are disabled when DBGACK is asserted
PORT3_PRIORITY[4:0] Bits 4-0	**Port 3Priority Level**—These bits set the priority level of the alternate master connected to port 2. The priority level of a requesting alternate master must be greater than that of the software priority in order to be granted control of the system bus.	00000 – Port 3alternate master has its lowest priority and will never be selected 00001 . . . 11110 11111 - Port 3alternate master has its highest priority

42.4 Arbiter Operation

The alternate master arbiter is a simple round-robin arbiter which monitors bus requests on its ports. Each alternate master has a bus request and bus grant signal that connects to the arbiter. Alternate masters also drive their address, attributes, and write data signals to the arbiter. The arbiter contains muxes which select the address, attributes, and write data signals from the active alternate master. The output of these muxes connect to the ARM7 platform. The ARM7 platform contains its own core arbiter (CARB) which arbitrates control of the ARM7 high performance (R-AHB) bus between the ARM7 core and all other alternate masters.

When an alternate master has control of the R-AHB, the clock to the ARM7 core is gated off via the ARM7 CLKEN signal. This provides a convenient method for gating off the ARM7 clock during low power modes. Low power modes are controlled by the Clock Control Module (CCM). The alternate master arbiter provides a separate bus request and bus grant port for the CCM. The CCM bus request port has a lower priority than any other alternate master and is not programmable. Low power mode operation of the alternate master arbiter is described below.

42.4.1 Alternate Master Bus Handoff

Each alternate master has a bus request and bus grant signal that connects to the arbiter. Each alternate master also has a programmable priority. The alternate master priority is compared to the priority level programmed for the current software process. An alternate master is allowed access to the system during normal modes, only if its programmed priority is higher than the software priority. During low power modes, the priority levels are ignored. Since software is not running in low power modes, an alternate master is granted access to the bus regardless of its priority level.

The arbiter uses a simple round-robin scheme. The arbiter scans each alternate master bus request port. If an alternate master asserts bus request and its priority level is higher than the current software process, the arbiter will pass its bus request to the ARM7 platform. Also, the bus grant signal is passed from the ARM7 platform to the requesting alternate master. When the transaction is completed, the alternate master arbiter will begin scanning the bus request ports and control of the R-AHB is passed back to the ARM7 core.

Figure 42-2 on page 42-18 shows the state diagram for the alternate master arbiter. When the system is in normal mode and no alternate master requests the bus, the arbiter cycles in the SAMPLE port states. When an alternate master requests the bus and has sufficient priority, the arbiter enters one of the GRANT states. When both the bus request and the bus grant negate, the arbiter will exit the GRANT state and begin sampling bus request ports.

Figure 42-3 on page 42-19 shows the logical connections of the various alternate master bus requests and bus grants

Figure 42-4 on page 42-20 shows a simple bus handoff to Port 0. This example shows a simple zero wait state case (no low power modes or interrupts considered).When Port 0 asserts its bus request signal, the alternate master arbiter state machine is sampling Port 2. One cycle later, the state machine samples Port 0. This timing diagram assumes that the priority level on Port 0 is configured to be higher than software's priority level. This results in the arbiter granting the bus request to Port 0 when it sees the Port 0 bus request asserted while sampling the port. The arbiter state machine enters the GRANT state for Port 0 and stays there until the Port 0 request and core bus grant are negated. After the alternate master access has completed, the arbiter begins scanning the ports, looking for a valid bus request.

42.4.2 Low Power Mode Operation

When an alternate master has control of the R-AHB, the clock to the ARM7 core is gated off via the ARM7 CLKEN signal. This provides a convenient method for gating off the ARM7 clock during low power modes. Low power modes are controlled by the Clock Control Module (CCM). The alternate master arbiter provides a separate bus request and bus grant port for the CCM. The CCM bus request port has a lower priority than any other alternate master and is not programmable.

When the CCM enters a low power mode, it will assert the BR_CCM signal to the alternate master arbiter. The arbiter immediately passes on this bus request to the ARM7 core arbiter. After the current memory cycle completes, the core arbiter will generate the bus grant signal output to the AMARB. The AMARB will enter its LOW POWER HOLD state and stay there until the data phase of the last ARM7 access is complete, or the CCM bus request negates. If the CCM bus request negates, the AMARB will begin sampling ports, starting with port 0. The AMARB will enter its LOW POWER state when the CCM bus request is asserted and the data phase of the last ARM7 access is completed. The bus grant signal from the CARB is passed on to the CCM when the AMARB enters its LOW POWER state. Once the CCM sees its bus grant signal assert (BG_CCM = 1), it can gate off the clock to the ARM7 platform and the memory system.

During WAIT mode, the clock output from the CCM to the alternate masters will toggle. Therefore, the alternate masters may request access to the system bus. If the arbiter is in its LOW POWER state and an alternate master requests the bus, the AMARB will leave the LOW POWER state and enter SAMPLE HOLD state (2 cycle delays) before begin sampling ports. The bus grant signal to the CCM (BG_CCM) will negate, enabling the ARM7 platform and memory system clock. The bus request signal output to the core arbiter will stay asserted. This will continue to gate off the clock to the ARM7 core. When a valid bus request signal from an alternate master is identified, the bus will be granted to that alternate master. Completion of the alternate master accesses will take the AMARB back to the LOW POWER HOLD, and LOW POWER states Figure 42-7 on page 42-23 shows the timing associated with alternate master accesses during wait mode.

Figure 42-6 on page 42-22 shows a simple case of entry/exit to/from wait mode. In the diagram, there are no pending bus requests from any of the alternate masters. The CCM asserts it bus request signal and negates it sometime later.

In the other low power modes, the clock connected to the AMARB and alternate masters is gated off. The clock is gated off a fixed number of cycles after the CCM receives the bus grant signal from the alternate master arbiter.

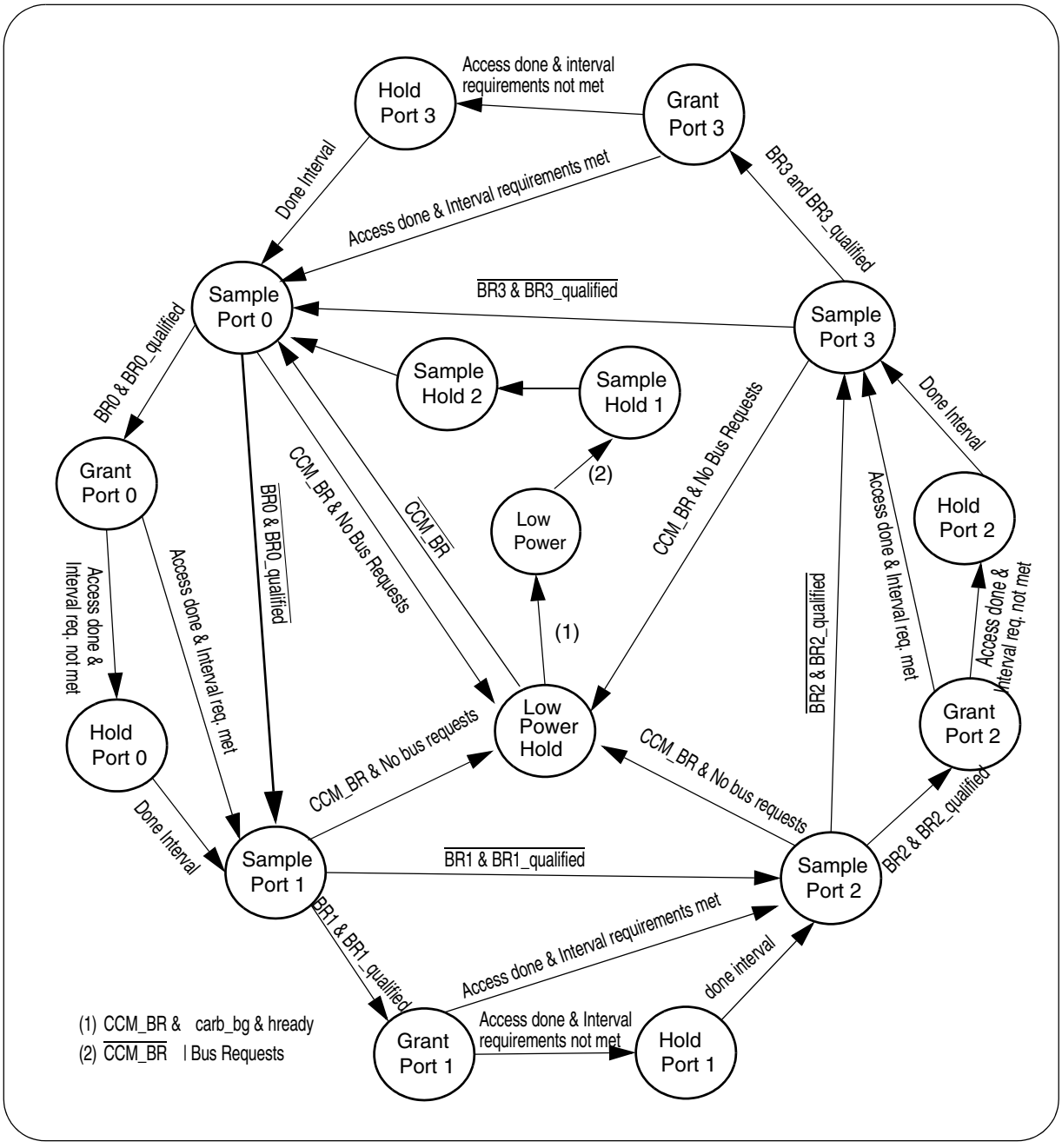
42.4.3 Bus Handoff Interval Control

The alternate master contains eight control bits in the Arbiter Interval Register (ARB_INTERVAL) that can be used to specify a minimum number of clocks that must elapse between bus handoff. A counter in the AMARB loads the INTERVAL[7:0] value and begins counting after the current alternate master access has completed. The counter counts down to 0 before bus request sampling begins on the other arbiter ports. This feature can be used to allow the processor a minimum number of bus cycles between bus handoffs. Out of reset, these bits are cleared to 0, which causes the arbiter to wait 0 cycles before sampling the next alternate master port. Figure 42-8 on page 42-24 shows a bus handoff to port 0 with the interval control set at 10. In this case, the arbiter waits 10 clock cycles before resuming port sampling.

It should be noted that the bus is always granted back to the ARM7 core between alternate master accesses. Figure 42-5 on page 42-21 shows simultaneous bus requests on port 0 and port 1 of the AMARB. In this example, the INTERVAL[7:0] value is equal to 0. The AMARB begins sampling ports immediately after the bus access by the alternate master on port 0 has completed.

During low power modes, the INTERVAL value is ignored. Since the ARM7 core is stalled, there is no reason to delay bus requests from alternate masters.

Figure 42-2. AMARB State Diagram



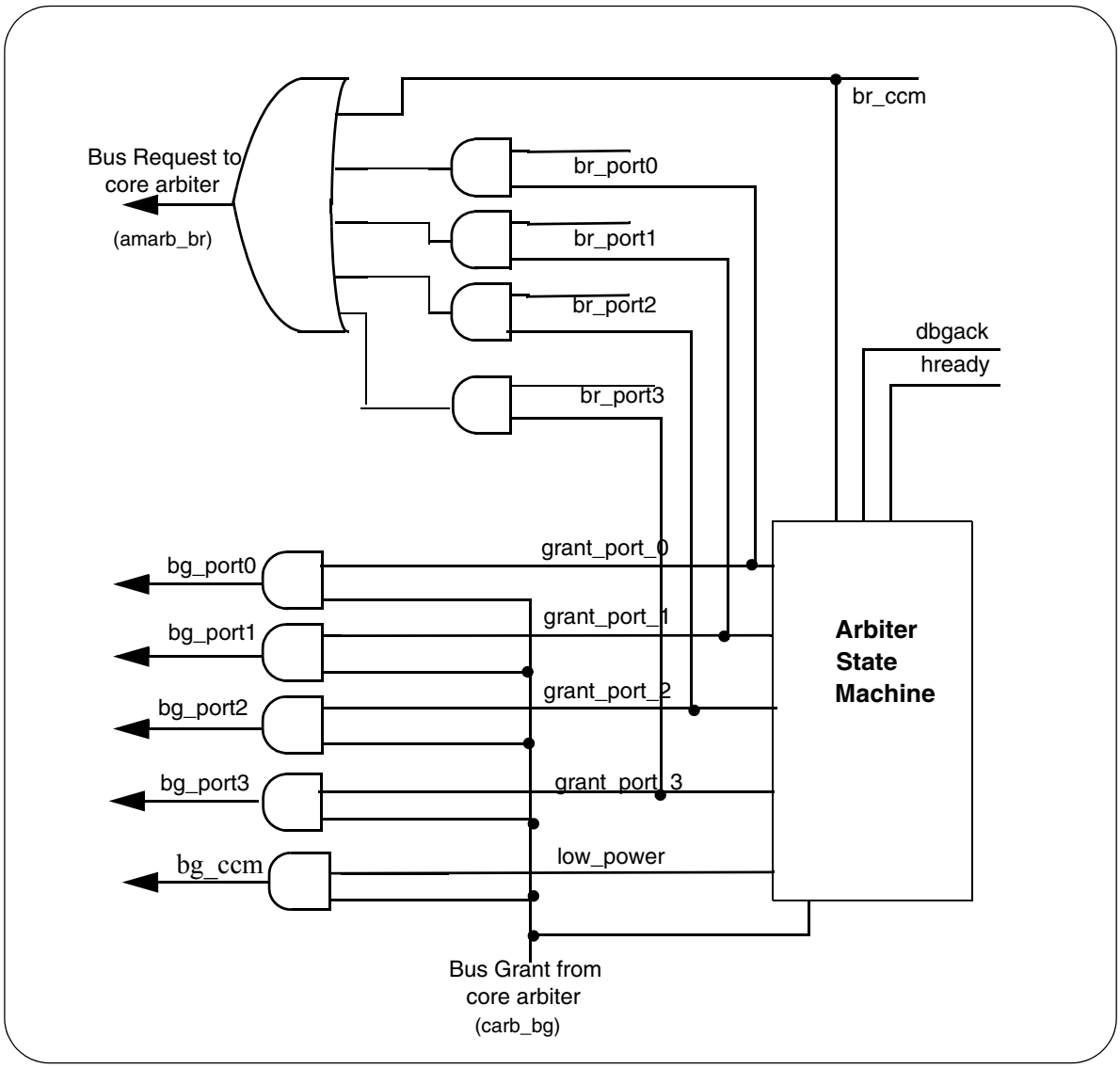


Figure 42-3. AMARB I/O Block Diagram

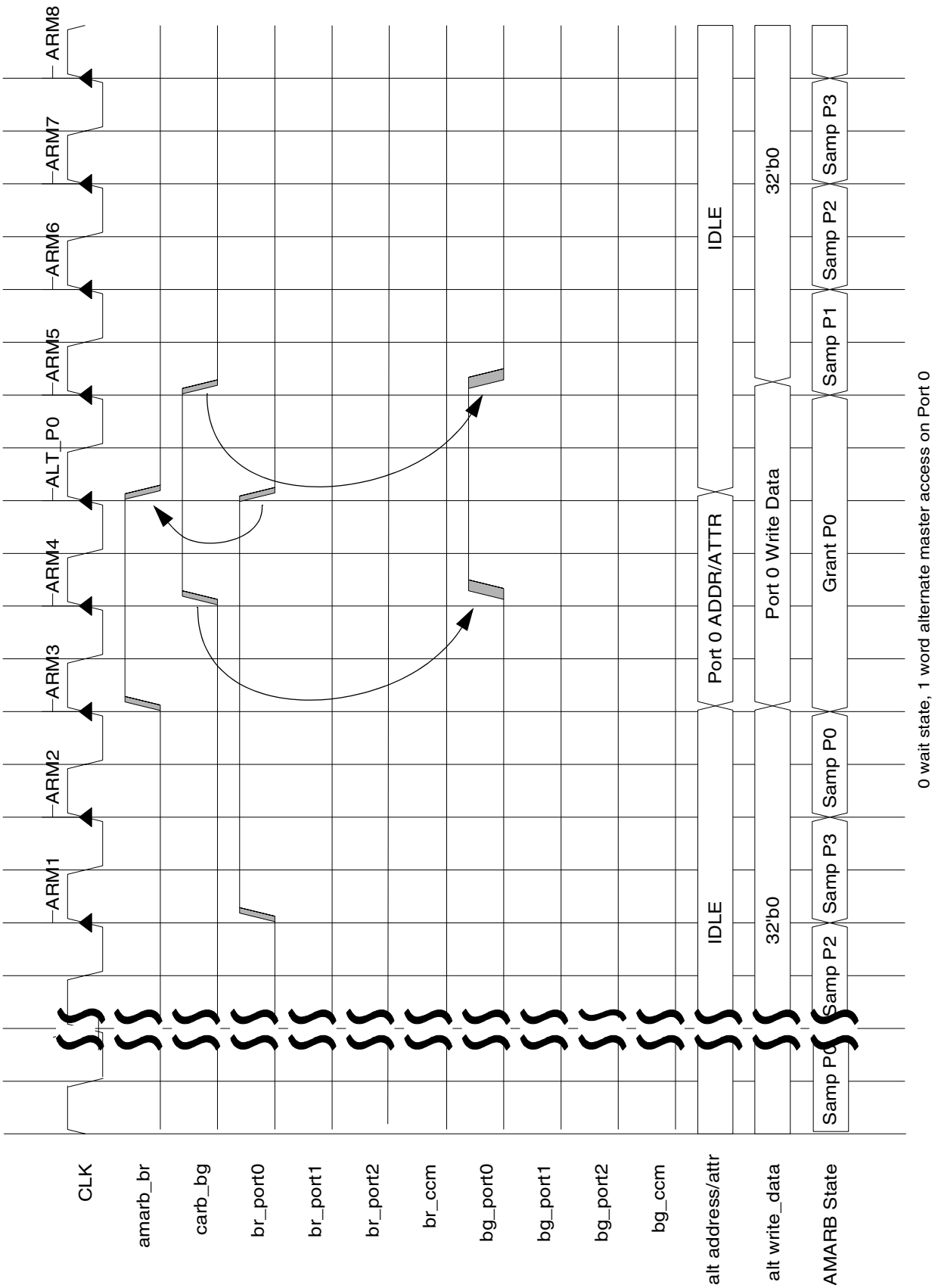


Figure 42-4. Alternate Master Access from Port 0

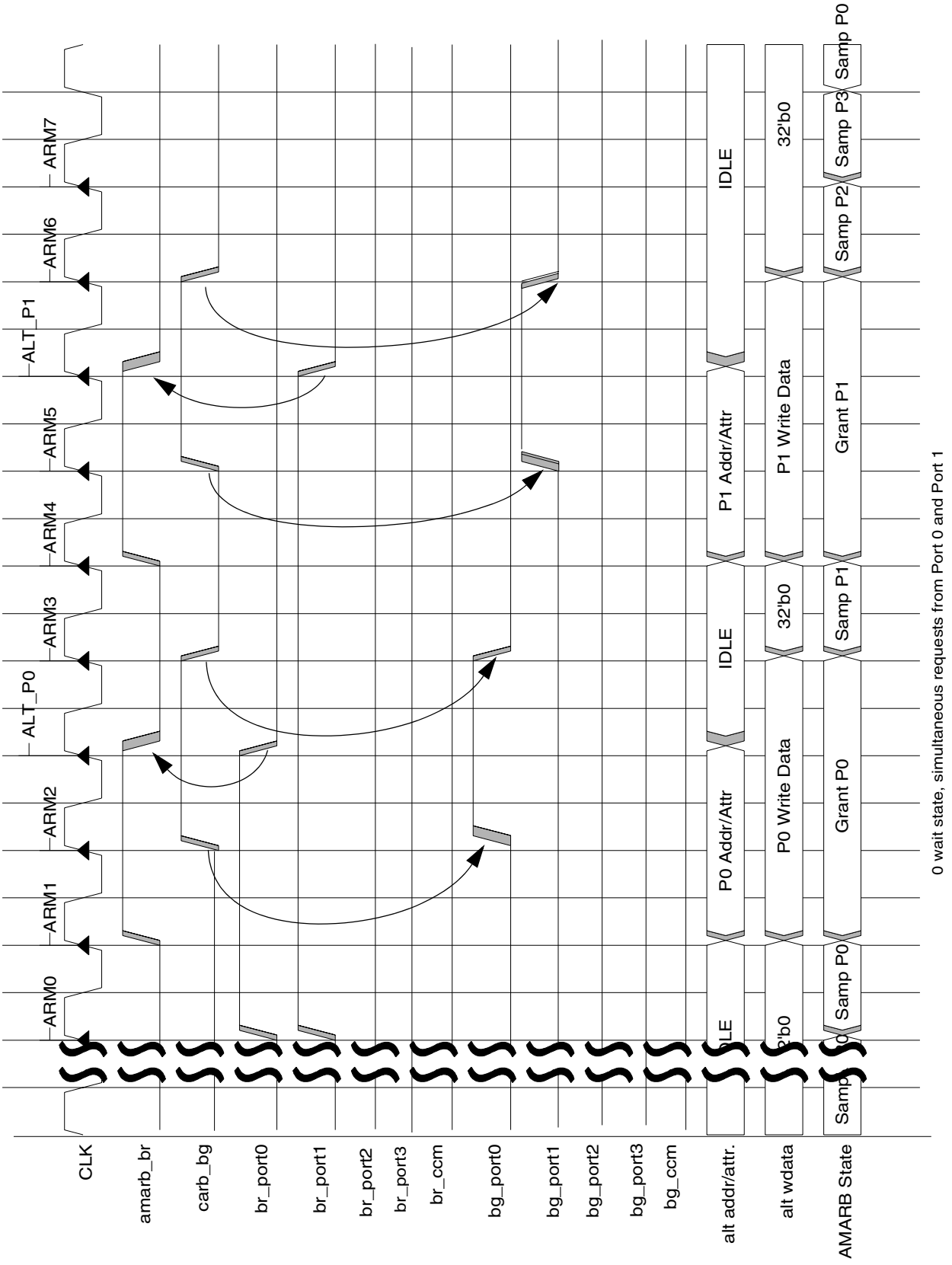


Figure 42-5. Simultaneous Requests from Port 0 and Port 1

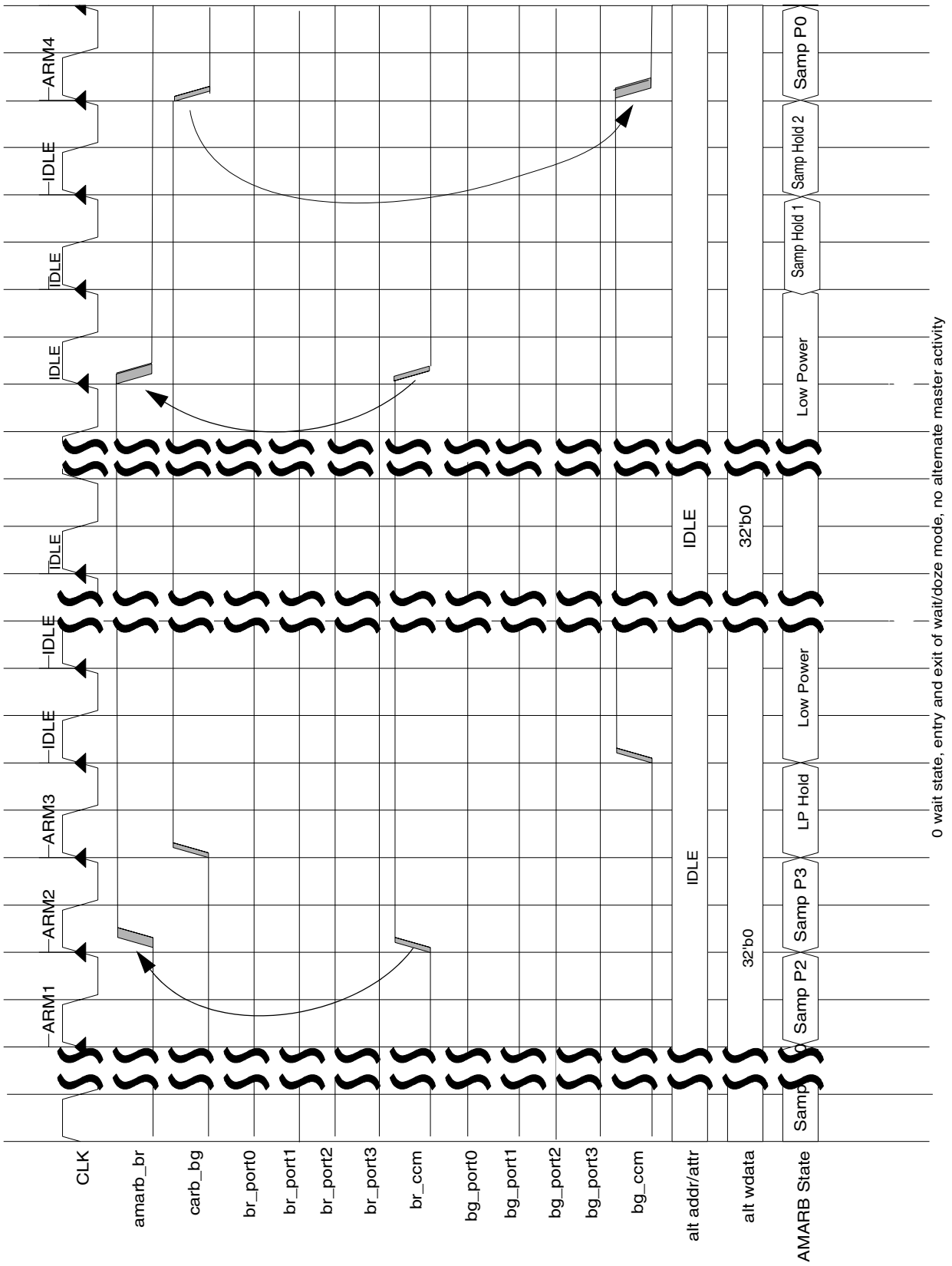


Figure 42-6. Wait Mode Entry and Exit

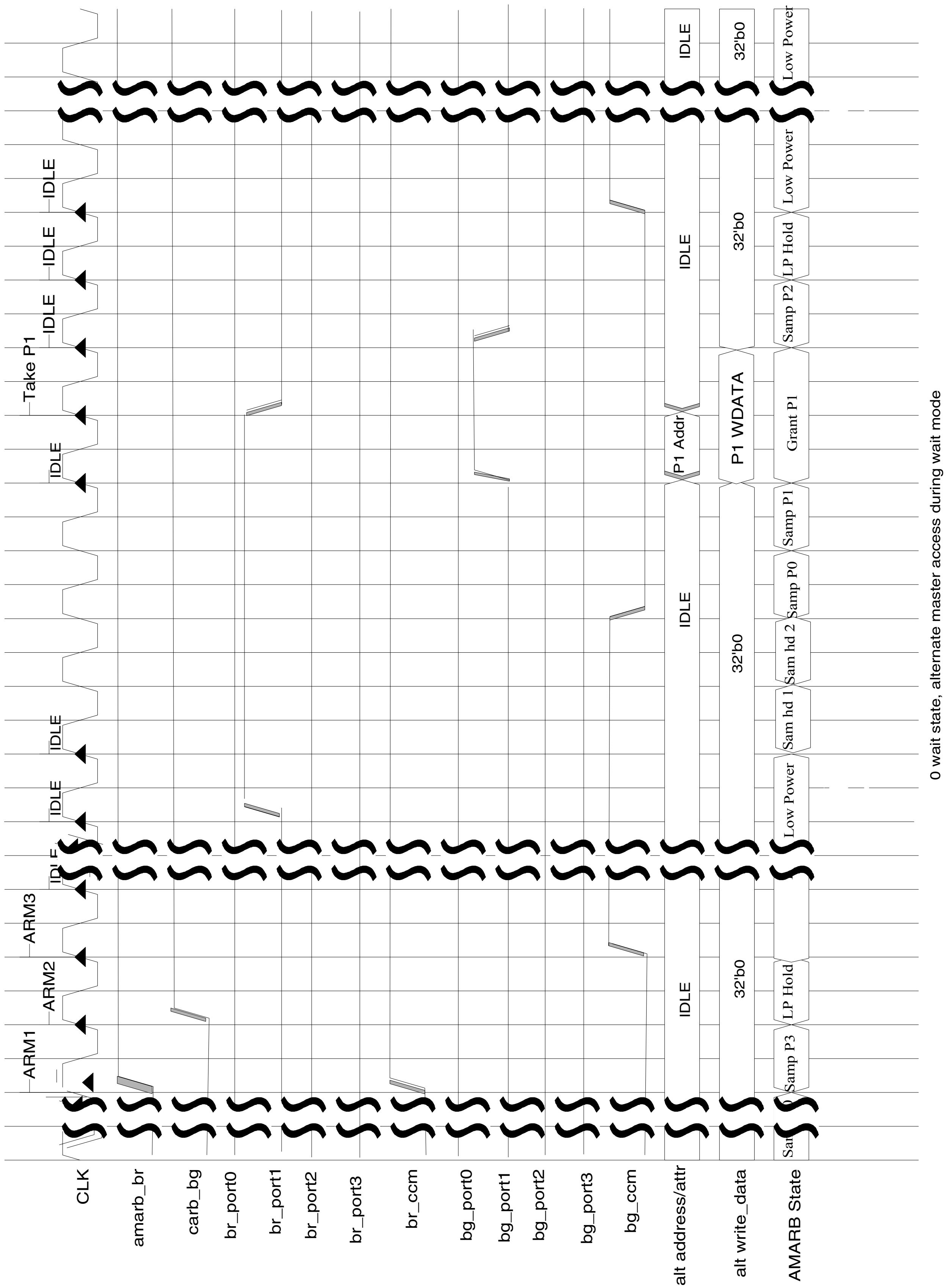


Figure 42-7. Alternate Master Access During Wait Mode

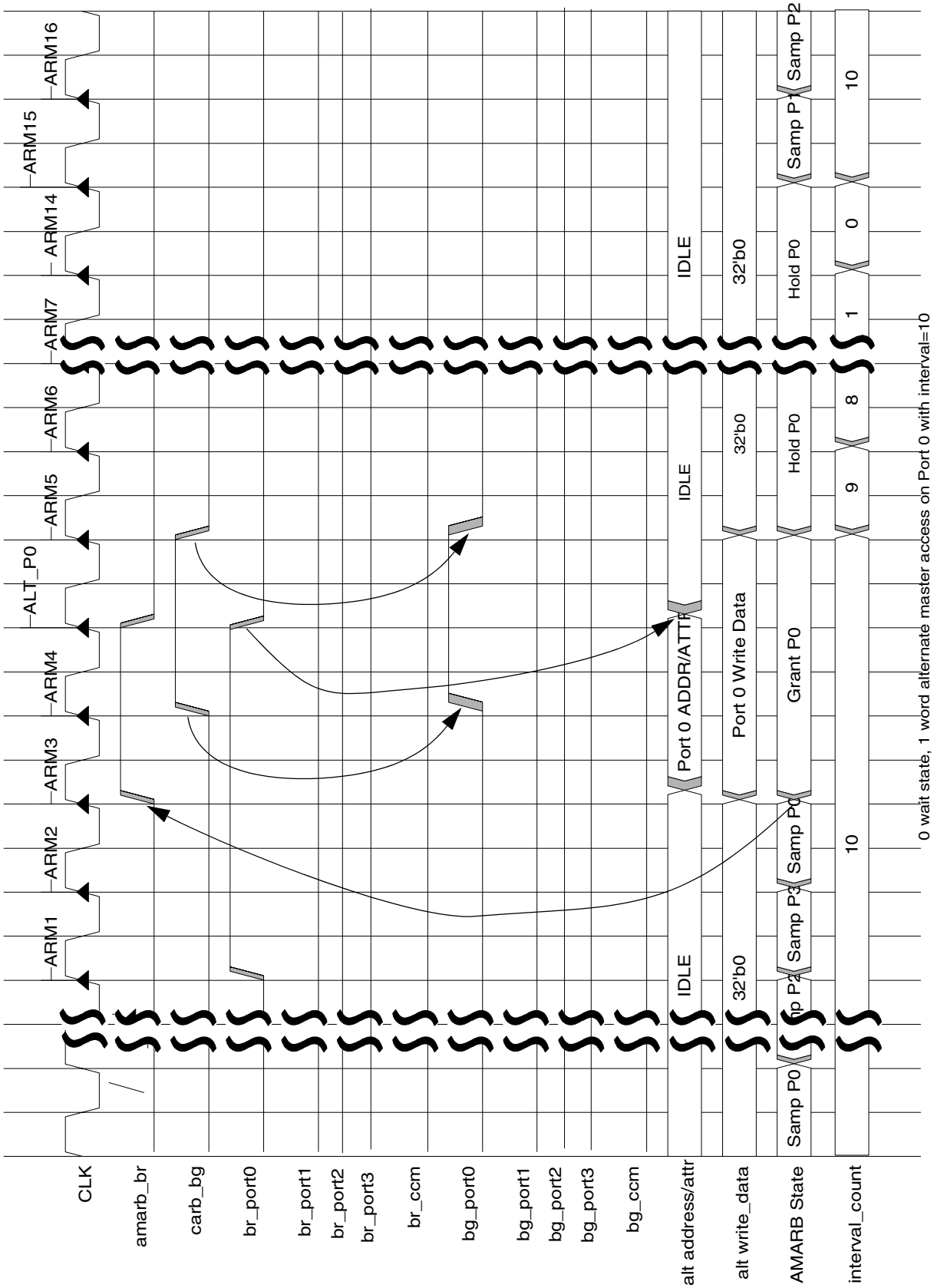


Figure 42-8. Bus Handoff Interval Control

42.4.4 Bus Control Monitoring

The AMARB provides a three-bit output signal that indicates the source of the alternate master bus request. This signal is used by the ARM platform core arbiter (CARB) and ARM platform external interface module (AEIM) to generate the QSTAT signals output during show cycles mode. Table 42-10 describes the *amarb_master[2:0]* encoding from the AMARB.

Table 42-10. amarb_master[2:0] Encoding

amarb_master[2:0]	Interpretation
000	No Alternate Master has the bus (default)
001	port 0 has control of alternate master bus (GEM)
010	port 1 has control of alternate master bus (DMAC)
011	port 2 has control of alternate master bus (HAC)
100	bus request from CCM
101*	port 3 has control of alternate master bus (TCM)

This encoding scheme maps with CARB hmaster [2:0] that go out to AEIM block to indicate the state of R-AHB. The default value of *amarb_master[2:0]* = 000. This value is output when there is no alternate master activity.

Note * The value amarb_master [2:0] = 101 for TCM can only be decoded as default/reserved value for QSTAT (due to already populated signal capacity). There is no unique QSTAT value that will support that TCM has control of alternate master bus.

42.5 Arbitration Protocol

Standard ARM AMBA AHB arbitration protocol is not used on Neptune. Instead, Neptune uses unique R-AHB arbitration protocol that is compatible with the M*CORE MLB alternate master devices that were previously designed for Patriot. The Neptune bus arbitration protocol relies on the concept of memory access “phases”. An access on the R-AHB bus can be broken down into an *address phase* followed by a *data phase*. The bus arbitration protocol’s use of these phases is described below.

Figure 42-9 on page 42-27 and Figure 42-10 on page 42-28 show simplified timing diagrams of the R-AHB arbitration protocol. The top 5 signals in the figures show the behavior of R-AHB signals during an access by an alternate master. The bottom 5 signals in the figures show the bus signals relevant to the alternate master.

Figure 42-9 shows the bus protocol used for an alternate master burst access of 2 words. In this case, the memory accesses are zero wait-state. When the alternate master wants control of the bus, it must assert its *bus request* signal after the rising edge of the clock. At the same time, the alternate master should present a stable set of address and access attribute signals. Although the write data is not needed by the memory system until after the access is “taken”, it is recommended that the alternate master presents the write data on the bus at the time the of the bus request. The *bus grant* response from the R-AHB occurs as the data phase of the current access completes. In Figure 42-9, it is the data phase of the ARM7 access to address “ARM0” that completes when *bus request* asserts. When the *bus request* signal from the alternate master and the *bus grant* signal from the system arbiters are both asserted, control of the R-AHB address/attribute bus is switched from the ARM7 core to the alternate master. “ALT0” from the alternate master address bus is muxed onto the R-AHB address bus. The alternate master’s access to ALT0 is taken by the memory

system on the next rising edge of the clock coincident with the assertion of *hready*. Since Figure 42-9 shows a zero wait-state case, *hready* is asserted every clock cycle. Thus, the ALT0 access by the alternate master is taken by the memory system on the next rising edge of the clock after *bus grant* is asserted. This completes the address phase of the ALT0 alternate master access.

When the address phase of a DMA access is done, the alternate master must indicate to the system whether it will retain control of the R-AHB or hand control back to the ARM7 core. Alternate master control of the bus is retained by continued assertion of the bus request signal. According to the bus arbitration protocol, the *bus request* signal must be asserted by the alternate master until the completion of the address phase of the last access done by the alternate master.

Figure 42-9 shows an example of a system bus handoff that lasts for 2 alternate master accesses. The alternate master continues to assert the *bus request* signal after completion of the ALT0 address phase. The address phase of an access is followed immediately by its data phase. The data phase of the current access continues until the *hready* signal is asserted at the rising edge of the clock. For a zero wait-state access, the data phase of is one clock cycle long. At the end of the data phase, data is transferred to/from the alternate master. In the case of a read, the value on the *hrdata* bus is latched by the alternate master. In the case of a write, the value on the *hwdata* bus is latched by the memory system. Data written by the alternate master must be presented on the *alt_hwdata* bus during the data phase of the access.

The data phase of the current access is coincident with the address phase of the next access. Therefore when each alternate master access is taken, the alternate master must update its address bus for the next access, or give up control of the bus by negating the *bus request* signal. In this example, the alternate master negates *bus request* after the address phase of the ALT1 access. The *bus grant* signal continues to be asserted by the system bus arbiter (CARB) until the data phase of the last alternate master access is complete.

Figure 42-10 illustrates a 2 word, 1 wait-state burst by an alternate master. Since the memory transfers are configured for 1 wait-state, the address and data phases of the accesses are two cycles long. *hready* is negated for the first cycle of the address and data phases. Alternate master must monitor the *hready* signal to determine the boundaries of the bus access phases.

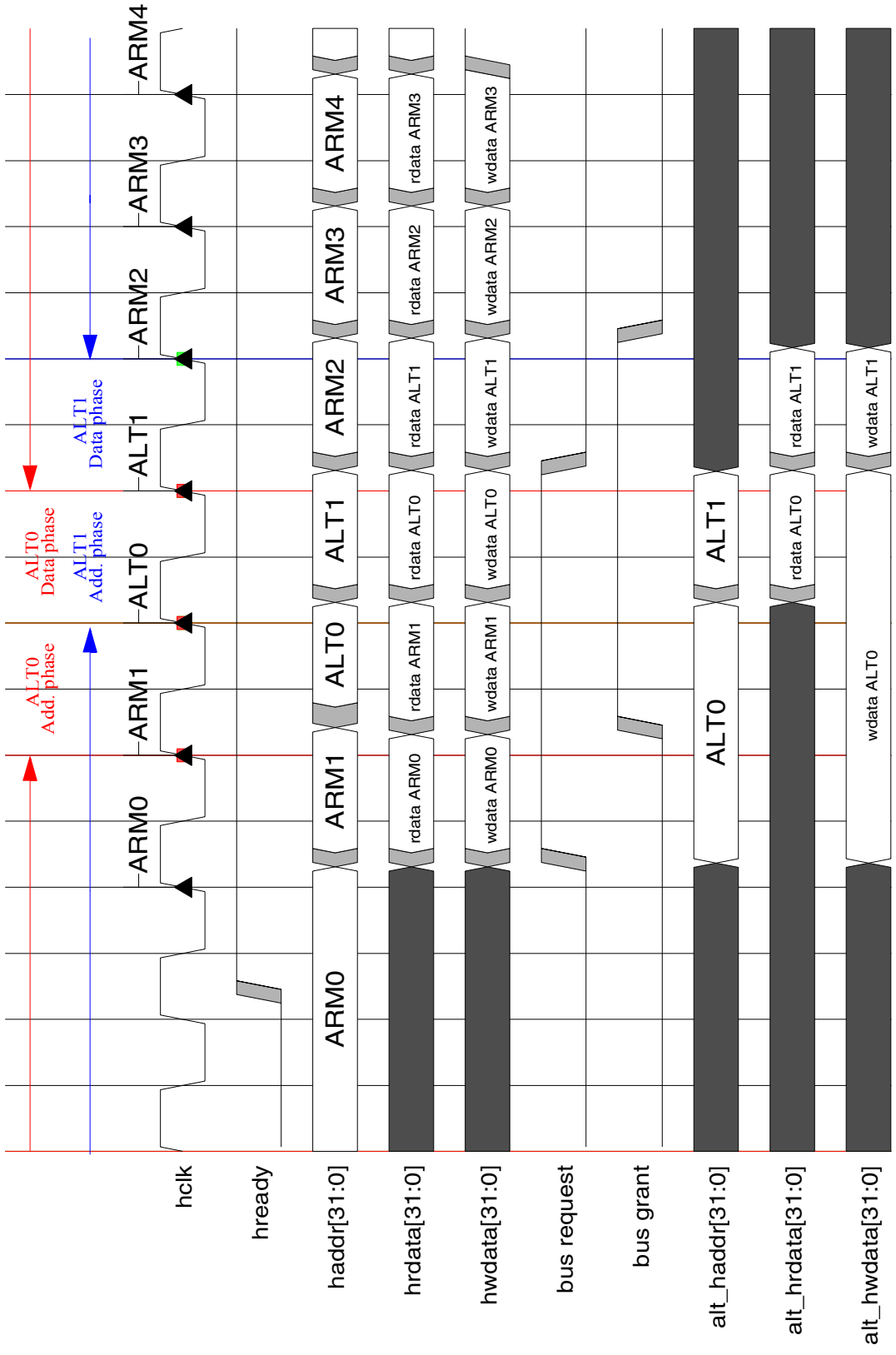


Figure 42-9. Neptune Bus Arbitration (0 wait states, 2 word access)

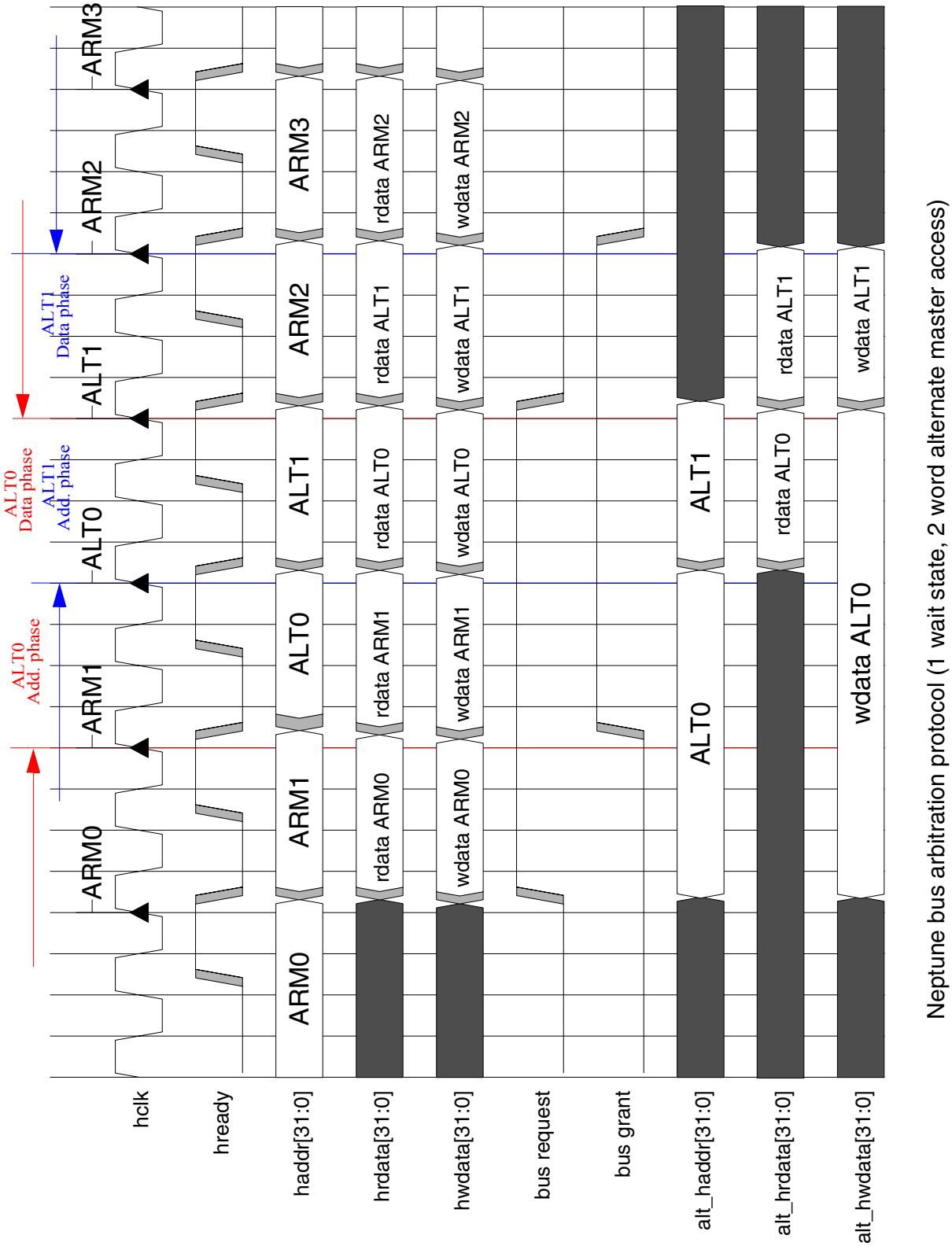


Figure 42-10. Neptune Bus Arbitration (1 wait-state, 2 word access)

Chapter 43
ARM7TDMI-S Interrupt Controller (AITC)

Revision History Table

Revision	Date	Author	Changes
0.0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	2/25/02	Pritam Kabe	Updated for secure operations: 1) Added NIDIS and FIDIS bits in INTCNTL register 2) Added aitc_disabled port 3) Updated the block diagram
0.2	3/21/02	Pritam Kabe	Updated for the new Arbitration control logic: Two new bits ABFLAG and ABFEN in the control register.
0.3	05/07/03		Updated for LTE specification release.

43.1 Introduction

The ARM7TDMI-S Interrupt Controller is a 32-bit peripheral which collects interrupt requests from up to 64 sources and provides an interface to the ARM7TDMI-S core. The AITC includes hardware acceleration of normal and fast interrupts. The AITC includes software controlled priority levels for normal interrupts.

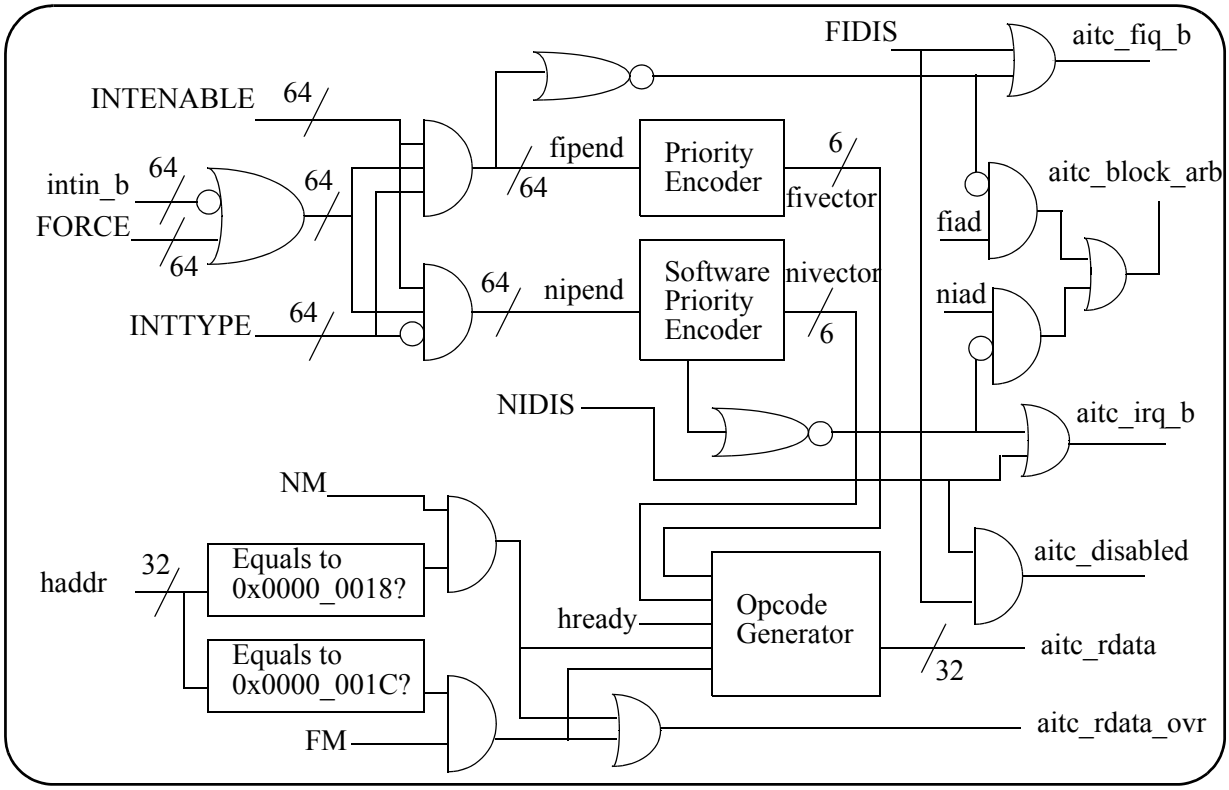


Figure 43-1. AITC Block Diagram

The AITC performs the following functions:

- Supports up to 64 interrupt sources
- Supports fast and normal interrupts
- Selects normal or fast interrupt request for any interrupt source
- Supports hardware accelerated vectoring to service routines for all interrupts
- Indicates pending interrupt sources via a register for normal and fast interrupts
- Indicates highest priority interrupt number via register (can be used a table index)
- Independently enable or disable any interrupt source
- Provides a mechanism for software to schedule an interrupt
- Provides three vector table modes for hardware acceleration: high memory, low memory with a table pointer, and complete software control
- Supports up to 16 software controlled priority levels for normal interrupts and priority masking
- Single bit disabling of all normal interrupts and of all fast interrupts, used in enabling of secure operations

The interrupt controller consists of a set of control registers and associated logic to perform interrupt masking, priority support, and hardware acceleration of normal interrupts.

The interrupt source registers (INTSRCH / INTSRCL) are a pair of 32-bit status registers with a single interrupt source associated with each of the 64 bits. An interrupt line or set of interrupt lines are routed from each interrupt source to the INTSRCH or INTSRCL register. This allows up to 64 distinct interrupt sources in an implementation. Interrupt requests may be forced to be asserted by way of the interrupt force registers (INTFRCH / INTFRCL). Each bit in this register is logically “OR-ed” with the corresponding hardware request line prior to feeding the INTSRCH or INTSRCL register inputs.

There is a corresponding set of interrupt enable registers (INTENABLEH / INTENABLEL), also 32-bits wide which allow individual bit masking of the INTSRCH / INTSRCL registers. There is also a corresponding set of interrupt type register (INTTYPEH / INTTYPEL) which selects whether an interrupt source will generate a normal or fast interrupt to the ARM7TDMI-S core.

There is a corresponding set of normal interrupt pending registers (NIPNDH / NIPNDL) which indicate pending normal interrupt requests. These registers are equivalent to the logical AND of the interrupt source registers (INTSRCH / INTSRCL), the interrupt enable registers (INTENABLEH / INTENABLEL), and the NOT of the interrupt type registers (INTTYPEH / INTTYPEL). (Refer to “AITC Block Diagram” on page 2) The NIPNDH / NIPNDL register bits are bit-wise “NOR-ed” together to form the nIRQ signal routed to the ARM7TDMI-S core. This core input signal is maskable by the normal interrupt disable bit (I bit) in the processor status register (CPSR). The normal interrupt vector register (NIVECSR) indicates the vector index of highest priority pending normal interrupt.

There is a corresponding set of fast interrupt pending registers (FIPNDH / FIPNDL) which indicate pending fast interrupt requests. These registers are equivalent to the logical AND of the interrupt source registers (INTSRCH / INTSRCL), the interrupt enable registers (INTENABLEH / INTENABLEL), and the interrupt type registers (INTTYPEH / INTTYPEL). (Refer to “AITC Block Diagram” on page 2) The FIPNDH / FIPNDL register bits are bit-wise “NOR-ed” together to form the nFIQ signal routed to the ARM7TDMI-S core. This core input signal is maskable by the fast interrupt disable bit (F bit) in the CPSR. The fast interrupt vector register (FIVECSR) indicates the vector index of highest priority pending fast interrupt.

The AITC includes hardware acceleration of normal interrupt service routine entry. If enabled and a normal interrupt occurs, the ARM7TDMI-S core will execute the command at location 0x0000_0018. When the core fetches this memory location, the AITC module will mask the normal read data bus (RDATA) and will generate an opcode. The RDATA will be overwritten until the memory subsystem asserts HREADY; which keeps the memory subsystem synchronized with the ARM7TDMI-S core.

The AITC supports two vector table modes: one is high memory and one in low memory. If the AITC is in high memory vector table mode, this opcode is “LDR PC, [PC,#-(288-4*(vector index)]”. This causes the ARM7TDMI-S core to load the Program Counter (PC) with a vector from a table of 64 vectors located at 0xFFFF_FF00 to 0xFFFF_FFFF; more specifically the PC is loaded with the vector located at 0xFFFF_FF00 + 4*(vector index). If the AITC is in low memory vector table mode, this opcode is “LDR PC, [PC, #((table pointer)+4*(vector index)-32]”. This causes the ARM7TDMI-S core to load the PC with a vector from a table of 64 vectors beginning at (table pointer) and ending at (table pointer)+0xFF; more specifically the PC is loaded with the vector located at (table pointer) + 4*(vector index). This hardware mechanism alleviates the need for software to determine which interrupt source caused the interrupt to be asserted.

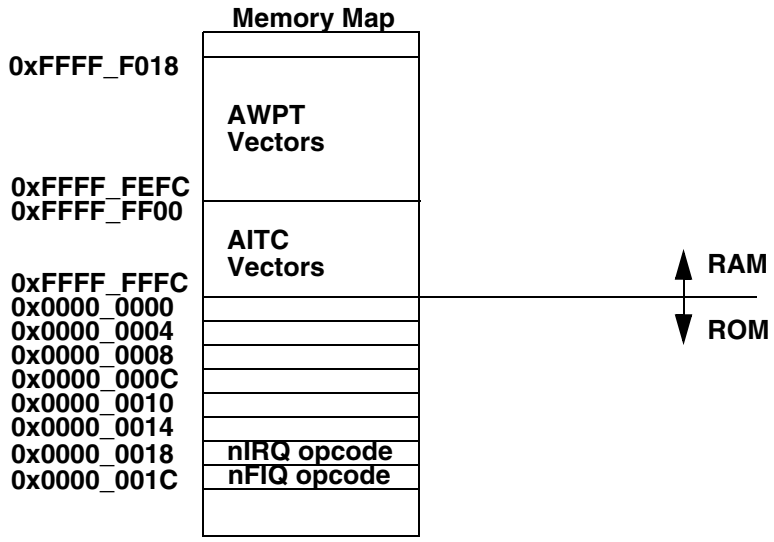


Figure 43-2. Example Memory Map for Opcode Patching With High AITC Vectors

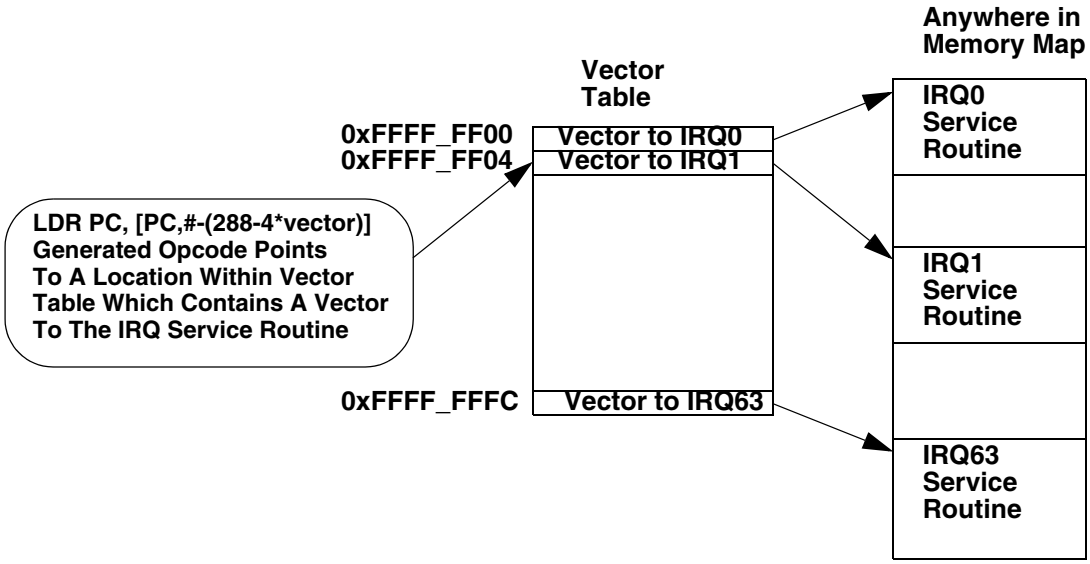


Figure 43-3. AITC Vector Table Located In High Memory

The AITC includes hardware acceleration of fast interrupt service routine entry. If enabled and a fast interrupt occurs, the ARM7TDMI-S core will execute the command at location 0x0000_001C. When the core fetches this memory location, the AITC module will mask the normal read data bus (RDATA) and will generate an opcode. If the AITC is in high memory vector table mode, this opcode is "LDR PC, [PC,#-(292-4*(vector index)]". If the ATIC is in low memory vector table mode, this opcode is "LDR PC, [PC, #((table pointer)+4*(vector index)-36]". The fast interrupt vector register (FIVECSR) indicates the vector index of highest priority pending fast interrupt.

All interrupt controller registers are readable and writable in privileged mode only. Writes attempted to read-only registers will be ignored. These registers can be only modified using 32-bit writes.

The INTFRCH / INTFRCL registers are provided for software generation of interrupts. By enabling interrupts for these bit positions, software can force an interrupt request. This register can also be used to debug hardware interrupt service routines by providing an alternate method of interrupt assertion.

The interrupt requests are prioritized in the following sequence:

1. Fast interrupt requests, in order of highest number
2. Normal interrupt requests, in order of highest priority level, then highest source number with the same priority

The AITC provides 16 software controlled priority levels for normal interrupts. Every interrupt can be placed in any priority level. The AITC also provides a normal interrupt priority level mask (NIMASK) which disables any interrupt with a priority level lower than or equal to the mask. If a level 0 normal interrupt and a level 1 normal interrupt are asserted at the same time, the level 1 normal interrupt will be selected assuming that NIMASK has not disabled level 1 normal interrupts. If two level 1 normal interrupts are asserted at the same time, the level 1 normal interrupt with the highest source number will be selected, also assuming that NIMASK has not disabled level 1 normal interrupts.

43.2 AITC Module Pin List

Table 43-1 is a list of the AITC module pins.

Table 43-1. AITC Module Pin List

<table>
<tr><th>Pin Name</th><th>Direction</th><th>Description</th></tr>
<tr><td colspan="3">Clock and Reset Signals</td></tr>
<tr><td>hclk</td><td>In</td><td>Clock. This signal is the main clock for the AITC. It should be connected to the same source as the mcu_clk signal provided to the ARM7TDMI-S core.</td></tr>
<tr><td>hreset_b</td><td>In</td><td>Reset. This active-low input signal resets the AITC. It should be tied to the same source as the nRESET signal to ARM7TDMI-S core.</td></tr>
<tr><td colspan="3">ARM7TDMI-S Native Bus Signals</td></tr>
<tr><td>haddr[31:0]</td><td>In</td><td>Address Bus. This bus specifies the address for the memory access. It is used to index the register file and in automatic vectoring of the prefetch abort.</td></tr>
<tr><td>hprot[1:0]</td><td>In</td><td>Transfer Protection Mode. This bus encodes information about the transfer. This is used to detect whether opcode fetches or data accesses are occurring and whether the access is a privileged one.
<table>
<tr><th colspan="2">hprot</th><th rowspan="2">Mode</th><th rowspan="2">Opcode / Data</th></tr>
<tr><th>1</th><th>0</th></tr>
<tr><td>0</td><td>0</td><td>User</td><td>Opcode</td></tr>
<tr><td>0</td><td>1</td><td>User</td><td>Data</td></tr>
<tr><td>1</td><td>0</td><td>Privileged</td><td>Opcode</td></tr>
<tr><td>1</td><td>1</td><td>Privileged</td><td>Data</td></tr>
</table></td></tr>
<tr><td>hready</td><td>In</td><td>Memory Transfer Acknowledge. This signal indicates that the memory controller has completed the current access. The signal is used to stretch the RDATA masking and overwriting. It is also used for the ABORT pipeline mirror within the Watch Point Engine.</td></tr>
</table>

Table 43-1. AITC Module Pin List (Continued)

<table>
<tr><th>Pin Name</th><th>Direction</th><th>Description</th></tr>
<tr><td>hsel</td><td>In</td><td>AITC Module Select. This signal indicates that the ARM7TDMI-S core is accessing a AITC register. This signal is decoded from the MSBs of the haddr bus.</td></tr>
<tr><td>hsize[1:0]</td><td>In</td><td>Transfer Size. This bus encodes the size of the transfer.
<table>
<tr><th colspan="2">hsize</th><th rowspan="2">Transfer Width</th></tr>
<tr><th>1</th><th>0</th></tr>
<tr><td>0</td><td>0</td><td>Byte</td></tr>
<tr><td>0</td><td>1</td><td>Half-word</td></tr>
<tr><td>1</td><td>0</td><td>Word</td></tr>
<tr><td>1</td><td>1</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>htrans[1:0]</td><td>In</td><td>Transfer Type. This bus encodes the type of the transfer using the R-AHB.
<table>
<tr><th colspan="2">htrans</th><th rowspan="2">Cycle Type</th><th rowspan="2">Transfer Type</th></tr>
<tr><th>1</th><th>0</th></tr>
<tr><td>0</td><td>0</td><td>I</td><td>Internal Cycle</td></tr>
<tr><td>0</td><td>1</td><td>C</td><td>Coprocessor Cycle</td></tr>
<tr><td>1</td><td>0</td><td>N</td><td>Nonsequential Cycle</td></tr>
<tr><td>1</td><td>1</td><td>S</td><td>Sequential Cycle</td></tr>
</table></td></tr>
<tr><td>hwdata[31:0]</td><td>In</td><td>Write Data Bus. This bus specifies the write data for the memory access. It is used to program the register file.</td></tr>
<tr><td>hwrite</td><td>In</td><td>Write Enable. This input signal specifies the direction of the transfer. When asserted, the memory access is a write cycle.</td></tr>
<tr><td colspan="3">Interrupt Controller Signals</td></tr>
<tr><td>aitc_block_arb</td><td>Out</td><td>Alternate Master Arbitration Block. This active-high signal indicates that a normal or fast interrupt is asserted and the AITC has been programmed to disable bus arbitration.</td></tr>
<tr><td>aitc_disabled</td><td>Out</td><td>AITC Disabled. This active high signal indicates that the AITC has been completely disabled. This is the “AND-ing” of the NIDIS and FIDIS bits of the INTCNTL register. This signal is used to enable secure operations.</td></tr>
<tr><td>aitc_fiq_b</td><td>Out</td><td>Fast Interrupt. This active-low signal provides a asynchronous fast interrupt request (nFIQ) to the Clock Control Module (CCM).</td></tr>
<tr><td>aitc_fiq_sync_b</td><td>Out</td><td>Synchronized Fast Interrupt. This active-low signal provides a synchronized fast interrupt request (nFIQ) to the ARM7TDMI-S core. This signal is intended to be used by any block which needs a synchronized version of the fast interrupt.</td></tr>
</table>

Table 43-1. AITC Module Pin List (Continued)

Pin Name	Direction	Description
aitc_hready	Out	AITC Transfer Acknowledge. This active-high signal indicates that AITC has completed a register access. This signal is asserted during the data portion of a register access to the AITC.
aitc_irq_b	Out	Normal Interrupt. This active-low signal provides a asynchronous normal interrupt request (nIRQ) to the Clock Control Module (CCM).
aitc_irq_sync_b	Out	Synchronized Normal Interrupt. This active-low signal provides a synchronized normal interrupt request (nIRQ) to the ARM7TDMI-S core. This signal is intended to be used by any block which needs a synchronized version of the normal interrupt.
aitc_rdata[31:0]	Out	AITC Read Data Bus. This bus is the read data bus from the AITC. It either contains the output from a register read, output from the opcode generator, or output from the data fix table.
aitc_rdata_ovr	Out	AITC Read Data Bus Overwrite. This active-high signal that the normal read data bus should be overwritten by the AITC. When asserted, the AHB_MUX block should multiplex off the normal read data bus from the memory controller and use the AITC's read data bus.
dsm_int_holdoff	In	Deep Sleep Interrupt Hold Off. This active-high signal indicates that the Deep Sleep Module (DSM) has disabled interrupt generation. When asserted, the AITC will disable the synchronized versions of the normal and fast interrupt outputs.
intin_b[63:0]	In	Interrupt Requests. These active-low signals indicate that an interrupt is being requested by a peripheral device to the interrupt controller. The interrupt controller will recognize an interrupt is asserted on the rising edge of the clock. The interrupt controller does not latch and hold the interrupt and it is the peripheral's responsibility to keep the interrupt request asserted until the software acknowledges and clears the interrupt request.

43.3 Interrupt Controller Programming Model

43.3.1 Register Summary

The AITC module has 26 registers. All of these registers are single cycle access as the AITC sits on the native bus of the ARM7TDMI-S core.

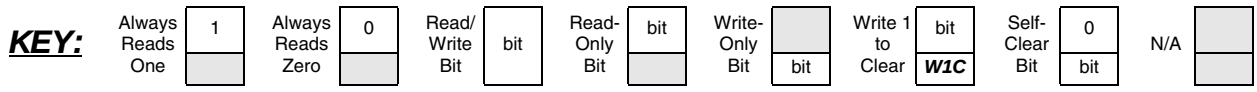


Table 43-2. AITC Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTCNTL ($2989_0000)	R	0	0	0	0	0	0	ABFLAG	ABFEN	0	NIDIS	FIDIS	NIAD	FIAD	NM	FM	MD	0	0	0	0	POINTER										0	0
	W																																
NIMASK ($2989_0004)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NIMASK				
	W																																
INTENNUM ($2989_0008)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																											ENNUM				
INTDISNUM ($2989_000C)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																											DISNUM				
INTENABLEH ($2989_0010)	R	INTENABLE[63:32]																															
	W																																
INTENABLEL ($2989_0014)	R	INTENABLE[31:0]																															
	W																																
INTTYPEH ($2989_0018)	R	INTTYPE[63:32]																															
	W																																
INTTYPEL ($2989_001C)	R	INTTYPE[31:0]																															
	W																																
NIPRIORITY7 ($2989_0020)	R	NIPR63				NIPR62				NIPR61				NIPR60				NIPR59				NIPR58				NIPR57				NIPR56			
	W																																
NIPRIORITY6 ($2989_0024)	R	NIPR55				NIPR54				NIPR53				NIPR52				NIPR51				NIPR50				NIPR49				NIPR48			
	W																																
NIPRIORITY5 ($2989_0028)	R	NIPR47				NIPR46				NIPR45				NIPR44				NIPR43				NIPR42				NIPR41				NIPR40			
	W																																
NIPRIORITY4 ($2989_002C)	R	NIPR39				NIPR38				NIPR37				NIPR36				NIPR35				NIPR34				NIPR33				NIPR32			
	W																																
NIPRIORITY3 ($2989_0030)	R	NIPR31				NIPR30				NIPR29				NIPR28				NIPR27				NIPR26				NIPR25				NIPR24			
	W																																
NIPRIORITY2 ($2989_0034)	R	NIPR23				NIPR22				NIPR21				NIPR20				NIPR19				NIPR18				NIPR17				NIPR16			
	W																																
NIPRIORITY1 ($2989_0038)	R	NIPR15				NIPR14				NIPR13				NIPR12				NIPR11				NIPR10				NIPR9				NIPR8			
	W																																
NIPRIORITY0 ($2989_003C)	R	NIPR7				NIPR6				NIPR5				NIPR4				NIPR3				NIPR2				NIPR1				NIPR0			
	W																																
NIVECSR ($2989_0040)	R	NIVECTOR																NIPRILVL															
	W																																
FIVECSR ($2989_0044)	R	FIVECTOR																															
	W																																

Table 43-2. AITC Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTSRCH ($2989_0048)	R	INTIN[63:32]																															
	W																																
INTSRCL ($2989_004C)	R	INTIN[31:0]																															
	W																																
INTFRCH ($2989_0050)	R	FORCE[63:32]																															
	W																																
INTFRCL ($2989_0054)	R	FORCE[31:0]																															
	W																																
NIPNDH ($2989_0058)	R	NIPEND[63:32]																															
	W																																
NIPNDL ($2989_005C)	R	NIPEND[31:0]																															
	W																																
FIPNDH ($2989_0060)	R	FIPEND[63:32]																															
	W																																
FIPNDL ($2989_0064)	R	FIPEND[31:0]																															
	W																																

43.3.2 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various AITC registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0.
  - **1**: Will reset to a logic 1.
  - **?**: The reset state is unknown.

The interrupt control register (INTCNTL) controls the hardware acceleration done by the AITC. Both normal interrupts and fast interrupts can be enabled to jump directly to the interrupt service routine. For fast interrupts, it may be faster to begin to fast interrupt routine at 0x0000_001C instead of jumping to a service routine.

The vector table can be sourced in high memory, 0xFFFF_FF00 to 0xFFFF_FFFF, or in low memory. If the vector table is located in low memory (MD=1), a register has been provided to control where the vector table is located.

This register is located on the ARM7TDMI-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode. This register can be only modified using 32-bit writes.

INTCNTL Interrupt Control Register **$2989_0000**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
							ABFLAG	ABFEN		NIDIS	FIDIS	NIAD	FIAD	NM	FM	MD
TYPE	r	r	r	r	r	r	r / w1c	rw	r	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
					POINTER											
TYPE	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-3. INTCNTL Descriptions

Name	Description	Settings
Bits 31-23	Reserved	N/A
ABFLAG Bit 25	**Core Arbitration Prioritization Risen Flag —** This status bit indicates that the AITC is asserting the aitc_block_arb signal to rise the priority level of the ARM core in the bus arbitration logic. This signal is directly tied to the aitc_block_arb output signal. When the ABFEN bit is cleared, this bit will indicate the current value of the aitc_block_arb signal to the ARM core. This bit will be set if the nFIQ is asserted and the FIAD bit is set or if the nIRQ is asserted and the NIAD bit is set. When the ABFEN bit is set, this bit will keep the “1” until written to with a “1”. This bit will remain set if either of the above conditions which normally set the ABFLAG bit is true.	0 = AITC is not affecting the bus arbitration priority levels 1 = AITC is rising ARM core priority level in bus arbitration logic.
ABFEN Bit 24	**ABFLAG Sticky Enable —** This bit controls whether the ABFLAG bit is “sticky”. This allows the arbitration logic to keep the ARM core at the higher priority level until the ABFLAG bit is written.	0 = ABFLAG bit is normal 1 = ABFLAG bit is “sticky” and requires a write of “1” to clear the bit.
Bit 23	Reserved	N/A

Table 43-3. INTCNTL Descriptions (Continued)

Name	Description	Settings
NIDIS Bit 22	**Normal Interrupt Disable--** This bit, when set, disables the generation of the normal interrupt signal. This bit is similar to the I bit of the ARM7TDMI-S core. This bit along with the FIDIS bit is used to enable secure operations.	0 = Does not affect the normal interrupt generation 1 = Disable all normal interrupts
FIDIS Bit 21	**Fast Interrupt Disable--** This bit, when set, disables the generation of the fast interrupt signal. This bit is similar to the F bit of the ARM7TDMI-S core. This bit along with the NIDIS bit is used to enable secure operations.	0 = Does not affect the fast interrupt generation 1 = Disable all fast interrupts
NIAD Bit 20	**Normal Interrupt Arbiter Disable—** This bit, when asserted, prevents the assertion of bus request to the core when the normal interrupt signal (nIRQ) is asserted. If an alternate master has ownership of the bus when a normal interrupt occurs, the bus will be given back to the processor core ***after*** the DMA device has completed its accesses. Therefore, the IRQ_DIS bit does not affect alternate master accesses that are in progress. In order to prevent an alternate master from accessing the bus during an interrupt service routine, the interrupt flag should not be cleared until the end of the service routine. Another option is to use the ABFEN and ABFLAG bits.	0 = Disregard the normal interrupt flag when evaluating bus requests 1 = Normal interrupt flag prevents alternate masters from accessing the system bus
FIAD Bit 19	**Fast Interrupt Arbiter Disable—** This bit, when asserted, prevents the assertion of bus request to the core when the fast interrupt signal (nFIQ) is asserted. If an alternate master has ownership of the bus when a fast interrupt occurs, the bus will be given back to the processor core ***after*** the DMA device has completed its accesses. Therefore, the IRQ_DIS bit does not affect alternate master accesses that are in progress. In order to prevent an alternate master from accessing the bus during an interrupt service routine, the interrupt flag should not be cleared until the end of the service routine. Another option is to use the ABFEN and ABFLAG bits.	0 = Disregard the fast interrupt flag when evaluating bus requests 1 = Fast interrupt flag prevents alternate masters from accessing the system bus
NM Bit 18	**Normal Interrupt Mode Control —** Selects the hardware acceleration mode for normal interrupt.	0 = AITC will not generate nIRQ opcode, i.e. complete software control 1 = Normal interrupt hardware acceleration enabled, i.e. generate opcode at 0x0000_0018 to jump to interrupt service routine

Table 43-3. INTCNTL Descriptions (Continued)

Name	Description	Settings
FM Bit 17	**Fast Interrupt Mode Control —** Selects the hardware acceleration mode for fast interrupt.	0 = AITC will not generate nFIQ opcode, i.e. complete software control 1 = Fast interrupt hardware acceleration enabled, i.e. generate opcode at 0x0000_001C to jump to interrupt service routine
MD Bit 16	**Interrupt Vector Table Mode —** Indicates whether the interrupt vector is located in high memory or low memory.	0 = Interrupt vector table located in high memory from 0xFFFF_FF00 to 0xFFFF_FFFF 1 = Interrupt vector table located in low memory from POINTER to POINTER+0xFF
Bits 15-12	Reserved	N/A
POINTER Bits 11-2	**Interrupt Vector Table Pointer —** Indicates start of the vector table when in low memory (MD=1).	The value stored in the 10 bits, times 4, must be set greater than or equal to 0x0000_0024 and less than or equal to 0x0000_0F00. Only word-aligned tables are allowed, and two zeros are added in the LSBs when this value is used by the AITC. The value stored here is left shifted by two bits, so the actual table vector can be directly written into the appropriate bits.
Bits 1-0	Reserved	N/A

The normal interrupt mask register (NIMASK) controls the normal interrupt mask level. All normal interrupts with a priority level lower than or equal to the NIMASK will be disabled. The priority level of normal interrupts are determined by the normal interrupt priority level registers (NIPRIORITY7, NIPRIORITY6, NIPRIORITY5, NIPRIORITY4, NIPRIORITY3, NIPRIORITY2, NIPRIORITY1, and NIPRIORITY0). The reset state of this register will not disable any normal interrupts.

Writing all 1's, or -1, to the NIMASK will set the normal interrupt mask to -1 which will not disable any normal interrupt priority levels.

This hardware mechanism can be used to create reentrant normal interrupt routines by disabling lower priority normal interrupts. Refer to Section 43.4.7, "Writing Reentrant Normal Interrupt Routines," on page 43-35 for more details on the use of the NIMASK register.

This register is located on the ARM7TDMI-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode. This register can be only modified using 32-bit writes.

NIMASK Normal Interrupt Mask Register **$2989_0004**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
												NIMASK				
TYPE	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Table 43-4. NIMASK Descriptions

Name	Description	Settings
Bits 31-5	Reserved	N/A
NIMASK Bits 4-0	**Normal Interrupt Mask —** Controls normal interrupt mask level. All normal interrupts of priority level lower than or equal to the NIMASK will be disabled.	-1 = Do not disable any normal interrupts 0 = Disable priority level 0 normal interrupts 1 = Disable priority level 1 and lower normal interrupts ... 15 = Disable all normal interrupts 16+ = Do not disable any normal interrupts

The interrupt enable number register (INTENNUM) provides a hardware accelerated enabling of interrupts. Any write to this register will enable one interrupt source. If the 6 LSBs are equal 000000, then interrupt source 0 is enabled. If the 6 LSBs equal 000001, then interrupt source 1 is enabled. And so forth. This register is decoded into a one hot mask which will be logically OR-ed with the INTENABLEH / INTENABLEL register.

This hardware mechanism alleviates the need for an atomic read/modify/write sequence to enable an interrupt source. To enable interrupts 10 and 20, the software need only preform two writes to the AITC: first write 10 to INTENNUM register, then write 20 to INTENNUM register (the order of the writes is irrelevant).

This register is located on the ARM7TDMI-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode. This register can be only modified using 32-bit writes. This register will always read back all 0s.

INTENNUM Interrupt Enable Number Register **$2989_0008**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
											ENNUM					
TYPE	r	r	r	r	r	r	r	r	r	r	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-5. INTENNUM Descriptions

Name	Description	Settings
Bits 31-6	Reserved	N/A
ENNUM Bits 5-0	**Interrupt Enable Number —** Writing to this register will enable the interrupt source associated with this value.	0 = Enable interrupt source 0 1 = Enable interrupt source 1 ... 63 = Enable interrupt source 63

The interrupt disable number register (INTDISNUM) provides a hardware accelerated disabling of interrupts. Any write to this register will disable one interrupt source. If the 6 LSBs are equal 000000, then interrupt source 0 is disabled. If the 6 LSBs equal 000001, then interrupt source 1 is disabled. And so forth. This register is decoded into a one hot mask which will be inverted and logically AND-ed with the INTENABLEH / INTENABLEL register.

This hardware mechanism alleviates the need for an atomic read/modify/write sequence to disable an interrupt source. To disable interrupts 10 and 20, the software need only preform two writes to the AITC: first write 10 to INTDISNUM register, then write 20 to INTDISNUM register (the order of the writes is irrelevant).

This register is located on the ARM7TDMI-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode. This register can be only modified using 32-bit writes. This register will always read back all 0s.

INTDISNUM Interrupt Disable Number Register **$2989_000C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
											DISNUM					
TYPE	r	r	r	r	r	r	r	r	r	r	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-6. INTDISNUM Descriptions

Name	Description	Settings
Bits 31-6	Reserved	N/A
DISNUM Bits 5-0	**Interrupt Disable Number —** Writing to this register will disable the interrupt source associated with this value.	0 = Disable interrupt source 0 1 = Disable interrupt source 1 ... 63 = Disable interrupt source 63

The interrupt enable register high (INTENABLEH) and the interrupt enable register low (INTENABLEL) are used to enable pending interrupt requests to the core. Each bit in this register corresponds to an interrupt source available in the system. The reset state of these registers are all interrupts masked.

This register can be updated by various methods: writing directly to the INTENABLEH / INTENABLEL registers, setting bits with the INTENNUM register, or clearing bits with the INTDISNUM register.

These registers are located on the ARM7TDMI-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode. These registers can be only modified using 32-bit writes.

INTENABLEH Interrupt Enable Register High $2989_0010

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	INTENABLE[63:48]															
TYPE	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	INTENABLE[47:32]															
TYPE	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTENABLEL Interrupt Enable Register Low $2989_0014

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	INTENABLE[31:16]															
TYPE	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	INTENABLE[15:0]															
TYPE	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-7. INTENABLEH / INTENABLEL Descriptions

Name	Description	Settings
INTENABLE Bits 31-0 Bits 31-0	**Interrupt Enable —** This bit enables the corresponding interrupt source to request a normal interrupt or a fast interrupt. A reset operation clears this bit. If an enable bit is set and the corresponding interrupt source is asserted, the interrupt controller will assert a normal or a fast interrupt request depending on associated INTTYPEH / INTTYPEL setting.	0 = Interrupt disabled 1 = Interrupt enabled and will generate a normal or fast interrupt upon assertion

The interrupt type register high (INTTYPEH) and the interrupt type register low (INTTYPEL) are used to select whether a pending interrupt source, when enabled with the INTENABLEH / INTENABLEL, will create a normal interrupt or a fast interrupt to the core. Each bit in this register corresponds to an interrupt source available in the system. The reset state of these registers will cause all enabled interrupt sources to generate a normal interrupt.

These registers are located on the ARM7TDMI-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode. These registers can be only modified using 32-bit writes.

INTTYPEH Interrupt Type Register High **$2989_0018**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	INTTYPE[63:48]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	INTTYPE[47:32]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTTYPEL Interrupt Type Register Low **$2989_001C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	INTTYPE[31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	INTTYPE[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-8. INTTYPEH / INTTYPEL Descriptions

Name	Description	Settings
INTTYPE Bits 31-0 Bits 31-0	**Interrupt Type —** This bit controls whether the corresponding interrupt source will request a normal interrupt or a fast interrupt. If a INTTYPE bit is set and the corresponding interrupt source is asserted, the interrupt controller will assert a fast interrupt request.	0 = Interrupt source will generate a normal interrupt (nIRQ) 1 = Interrupt source will generate a fast interrupt (nFIQ)

The normal interrupt priority level registers (NIPRIORITY7, NIPRIORITY6, NIPRIORITY5, NIPRIORITY4, NIPRIORITY3, NIPRIORITY2, NIPRIORITY1, and NIPRIORITY0) provide a software controllable prioritization of normal interrupts. Normal interrupts with a higher priority level will preempt normal interrupts with a lower priority. The reset state of these registers forces all normal interrupts to the lowest priority level.

If a level 0 normal interrupt and a level 1 normal interrupt are asserted at the same time, the level 1 normal interrupt will be selected assuming that NIMASK has not disabled level 1 normal interrupts. If two level 1 normal interrupts are asserted at the same time, the level 1 normal interrupt with the highest source number will be selected, also assuming that NIMASK has not disabled level 1 normal interrupts.

These registers are located on the ARM7TDMI-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode. These registers can be only modified using 32-bit writes.

NIPRIORITY7 Normal Interrupt Priority Level Register 7 **$2989_0020**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	NIPR63				NIPR62				NIPR61				NIPR60			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	NIPR59				NIPR58				NIPR57				NIPR56			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-9. NIPRIORITY7 Descriptions

Name	Description	Settings
NIPR63 Bits 31-28 **NIPR62** Bits 27-24 **NIPR61** Bits 23-20 **NIPR60** Bits 19-16 **NIPR59** Bits 15-12 **NIPR58** Bits 11-8 **NIPR57** Bits 7-4 **NIPR56** Bits 3-0	**Normal Interrupt Priority Level —** Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities.	0 = Lowest priority normal interrupt ... 15 = Highest priority normal interrupt

NIPRIORITY6 Normal Interrupt Priority Level Register 6 $2989_0024

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	NIPR55				NIPR54				NIPR53				NIPR52			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	NIPR51				NIPR50				NIPR49				NIPR48			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-10. NIPRIORITY6 Descriptions

Name	Description	Settings
NIPR55 Bits 31-28 **NIPR54** Bits 27-24 **NIPR53** Bits 23-20 **NIPR52** Bits 19-16 **NIPR51** Bits 15-12 **NIPR50** Bits 11-8 **NIPR49** Bits 7-4 **NIPR48** Bits 3-0	**Normal Interrupt Priority Level —** Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities.	0 = Lowest priority normal interrupt ... 15 = Highest priority normal interrupt

NIPRIORITY5 Normal Interrupt Priority Level Register 5 $2989_0028

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	NIPR47				NIPR46				NIPR45				NIPR44			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	NIPR43				NIPR42				NIPR41				NIPR40			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-11. NIPRIORITY5 Descriptions

Name	Description	Settings
NIPR47 Bits 31-28 **NIPR46** Bits 27-24 **NIPR45** Bits 23-20 **NIPR44** Bits 19-16 **NIPR43** Bits 15-12 **NIPR42** Bits 11-8 **NIPR41** Bits 7-4 **NIPR40** Bits 3-0	**Normal Interrupt Priority Level —** Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities.	0 = Lowest priority normal interrupt ... 15 = Highest priority normal interrupt

NIPRIORITY4 Normal Interrupt Priority Level Register 4 $2989_002C

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	NIPR39				NIPR38				NIPR37				NIPR36			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	NIPR35				NIPR34				NIPR33				NIPR32			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-12. NIPRIORITY4 Descriptions

Name	Description	Settings
NIPR39 Bits 31-28 **NIPR38** Bits 27-24 **NIPR37** Bits 23-20 **NIPR36** Bits 19-16 **NIPR35** Bits 15-12 **NIPR34** Bits 11-8 **NIPR33** Bits 7-4 **NIPR32** Bits 3-0	**Normal Interrupt Priority Level —** Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities.	0 = Lowest priority normal interrupt ... 15 = Highest priority normal interrupt

NIPRIORITY3 Normal Interrupt Priority Level Register 3 $2989_0030

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	NIPR31				NIPR30				NIPR29				NIPR28			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	NIPR27				NIPR26				NIPR25				NIPR24			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-13. NIPRIORITY3 Descriptions

Name	Description	Settings
NIPR31 Bits 31-28 **NIPR30** Bits 27-24 **NIPR29** Bits 23-20 **NIPR28** Bits 19-16 **NIPR27** Bits 15-12 **NIPR26** Bits 11-8 **NIPR25** Bits 7-4 **NIPR24** Bits 3-0	**Normal Interrupt Priority Level —** Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities.	0 = Lowest priority normal interrupt ... 15 = Highest priority normal interrupt

NIPRIORITY2 Normal Interrupt Priority Level Register 2 $2989_0034

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	NIPR23				NIPR22				NIPR21				NIPR20			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	NIPR19				NIPR18				NIPR17				NIPR16			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-14. NIPRIORITY2 Descriptions

Name	Description	Settings
NIPR23 Bits 31-28 **NIPR22** Bits 27-24 **NIPR21** Bits 23-20 **NIPR20** Bits 19-16 **NIPR19** Bits 15-12 **NIPR18** Bits 11-8 **NIPR17** Bits 7-4 **NIPR16** Bits 3-0	**Normal Interrupt Priority Level —** Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities.	0 = Lowest priority normal interrupt ... 15 = Highest priority normal interrupt

NIPRIORITY1 Normal Interrupt Priority Level Register 1 $2989_0038

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	NIPR15				NIPR14				NIPR13				NIPR12			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	NIPR11				NIPR10				NIPR9				NIPR8			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-15. NIPRIORITY1 Descriptions

Name	Description	Settings
NIPR15 Bits 31-28 **NIPR14** Bits 27-24 **NIPR13** Bits 23-20 **NIPR12** Bits 19-16 **NIPR11** Bits 15-12 **NIPR10** Bits 11-8 **NIPR9** Bits 7-4 **NIPR8** Bits 3-0	**Normal Interrupt Priority Level —** Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities.	0 = Lowest priority normal interrupt ... 15 = Highest priority normal interrupt

NIPRIORITY0 Normal Interrupt Priority Level Register 0 $2989_003C

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	NIPR7				NIPR6				NIPR5				NIPR4			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	NIPR3				NIPR2				NIPR1				NIPR0			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-16. NIPRIORITY0 Descriptions

Name	Description	Settings
NIPR7 Bits 31-28 **NIPR6** Bits 27-24 **NIPR5** Bits 23-20 **NIPR4** Bits 19-16 **NIPR3** Bits 15-12 **NIPR2** Bits 11-8 **NIPR1** Bits 7-4 **NIPR0** Bits 3-0	**Normal Interrupt Priority Level —** Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities.	0 = Lowest priority normal interrupt ... 15 = Highest priority normal interrupt

The normal interrupt vector and status register (NIVECSR) provides the priority of the highest pending normal interrupt and provides the vector index of the interrupt's service routine. This hardware mechanism replaces the previous necessity for core support of the FF1 command. This number can be directly used as an index into a vector table to select the highest pending normal interrupt source.

This read-only register is located on the ARM7TDMI-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode.

NIVECSR Normal Interrupt Vector and Status Register **$2989_0040**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	NIVECTOR															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	NIPRILVL															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 43-17. NIVECSR Descriptions

Name	Description	Settings
NIVECTOR Bits 31-16	**Normal Interrupt Vector —** Indicates vector index for the highest pending normal interrupt.	-1 = No normal interrupt request pending 0 = Interrupt 0 highest priority pending normal interrupt 1 = Interrupt 1 highest priority pending normal interrupt ... 63 = Interrupt 63 highest priority pending normal interrupt 64+ (not -1) = unused, will not occur
NIPRILVL Bits 15-0	**Normal Interrupt Priority Level —** Indicates the priority level of the highest priority normal interrupt. This number can be written to the NIMASK to disable the current priority normal interrupts to build a reentrant normal interrupt system.	-1 = No normal interrupt request pending 0 = Highest priority normal interrupt is level 0 1 = Highest priority normal interrupt is level 1 ... 15 = Highest priority normal interrupt is level 15 16+ (not -1) = unused, will not occur

The fast interrupt vector and status register (FIVECSR) provides the vector index for the highest priority active fast interrupt's service routine (the higher the source number of the fast interrupt, the higher the priority level). This hardware mechanism replaces the previous necessity for core support of the FF1 command. This number can be directly used as an index into a vector table to select the highest pending fast interrupt source.

This read-only register is located on the ARM7TDMI-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode.

FIVECSR Fast Interrupt Vector and Status Register **$2989_0044**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	FIVECTOR[31:16]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	FIVECTOR[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 43-18. FIVECSR Descriptions

Name	Description	Settings
FIVECTOR Bits 31-0	**Fast Interrupt Vector —** Indicates vector index for the highest pending fast interrupt.	-1 = No fast interrupt request pending 0 = Interrupt 0 highest pending fast interrupt 1 = Interrupt 1 highest pending fast interrupt ... 63 = Interrupt 63 highest pending fast interrupt 64+ (not -1) = unused, will not occur

The interrupt source register high (INTSRCH) and the interrupt source register low (INTSRCL) are each 32 bits wide. INTSRCH and INTSRCL reflect the status of all interrupt request inputs into the interrupt controller. Unused bit positions always read zero (no request pending). The state of this register out of reset is determined by the peripheral circuits generating the requests; normally, the requests would be inactive.

These read-only registers are located on the ARM7TDMI-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode.

INTSRCH Interrupt Source Register High $2989_0048

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	INTIN[63:48]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	INTIN[47:32]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: The state of this register out of reset is determined by the peripheral circuits generating the requests; normally, the requests would be inactive. This read-only register should be accessed with 32 bit reads only.

INTSRCL Interrupt Source Register Low $2989_004C

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	INTIN[31:16]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	INTIN[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: The state of this register out of reset is determined by the peripheral circuits generating the requests; normally, the requests would be inactive. This read-only register should be accessed with 32 bit reads only.

Table 43-19. INTSRCH / INTSRCL Description

Name	Description	Settings
INTIN Bits 31-0 Bits 31-0	**Interrupt Source—** Indicates the state of the corresponding hardware interrupt source.	0 = Interrupt source negated 1 = Interrupt source asserted

The interrupt force register high (INTFRCH) and the interrupt force register low (INTFRCL) are each 32 bits wide. The interrupt force registers allow for software generation of interrupts for each of the possible interrupt sources for functional or debug purposes. The system level design may reserve one or more sources for software purposes to allow software to self-schedule interrupts by forcing one or more of these “sources” in the appropriate interrupt force register(s).

These registers are located on the ARM7TDMI-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode. These registers can be only modified using 32-bit writes.

INTFRCH Interrupt Force Register High **$2989_0050**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	FORCE[63:48]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	FORCE[47:32]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTFRCL Interrupt Force Register Low **$2989_0054**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	FORCE[31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	FORCE[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-20. INTFRCH / INTFRCL Descriptions

Name	Description	Settings
FORCE Bits 31-0 Bits 31-0	**Interrupt Source Force Request—** Used to force a request for the corresponding interrupt source.	0 = Standard interrupt operation 1 = Interrupt forced asserted

The normal interrupt pending register high (NIPNDH) and the normal interrupt pending register low (NIPNDL) are 32-bit wide registers used to monitor the outputs of the enable and masking operations. These registers are actually only a set of buffers; therefore, the reset state of these registers are determined by the normal interrupt enable registers, the interrupt mask register, and the interrupt source registers. The value reflected in these registers is unaffected by the value of the NIMASK register.

These read-only registers are located on the ARM7TDMI-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode.

NIPNDH Normal Interrupt Pending Register High **$2989_0058**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	NIPEND[63:48]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	NIPEND[47:32]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NIPNDL Normal Interrupt Pending Register Low **$2989_005C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	NIPEND[31:16]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	NIPEND[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-21. NIPNDH / NIPNDL Descriptions

Name	Description	Settings
NIPEND Bits 31-0 Bits 31-0	**Normal Interrupt Pending Bit—** If a normal interrupt enable bit is set and the corresponding interrupt source is asserted, the interrupt controller will assert a normal interrupt request. The normal interrupt pending bits reflect the interrupt input lines which are asserted and are currently enabled to generate a normal interrupt.	0 = No normal interrupt request 1 = Normal interrupt request pending

The fast interrupt pending register high (FIPNDH) and the fast interrupt pending register low (FIPNDL) are 32-bit wide registers used to monitor the outputs of the enable and masking operations. These registers are actually only a set of buffers; therefore, the reset state of these registers are determined by the fast interrupt enable registers, the interrupt mask register, and the interrupt source registers.

These read-only registers are located on the ARM7TDMI-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode.

FIPNDH Fast Interrupt Pending Register High **$2989_0060**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	FIPEND[63:48]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	FIPEND[47:32]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FIPNDL Fast Interrupt Pending Register Low **$2989_0064**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	FIPEND[31:16]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	FIPEND[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-22. FIPNDH / FIPNDL Descriptions

Name	Description	Settings
FIPEND Bits 31-0 Bits 31-0	**Fast Interrupt Pending Bit—** If a fast interrupt enable bit is set and the corresponding interrupt source is asserted, the interrupt controller will assert a fast interrupt request. The fast interrupt pending bits reflect the interrupt input lines which are asserted and are currently enabled to generate a fast interrupt.	0 = No fast interrupt request 1 = Fast interrupt request pending

43.4 ARM7TDMI-S Interrupt Controller Operation

43.4.1 ARM7TDMI-S Prioritization of Exception Sources

The ARM7TDMI-S core imposes the following priority among the various exceptions:

- Reset (highest priority)
- Data Abort
- Fast Interrupt
- Normal Interrupt
- Prefetch Abort
- Undefined Instruction and SWI (lowest priority)

43.4.2 AITC Prioritization of Interrupt Sources

The AITC module prioritizes the various interrupt sources by source number where higher source numbers have higher priority. Fast interrupt always have higher priority over normal interrupts.

The interrupt requests are prioritized in the following sequence:

1. Fast interrupt requests, in order of highest source number
2. Normal interrupt requests, in order of highest priority level, then in order of highest source number with the same priority level

43.4.3 Assigning and Enabling Interrupt Sources

The interrupt controller provides for flexible assignment of any interrupt source to either of the two core interrupt request inputs. This is done by setting the appropriate bits in the INTENABLEH / INTENABLEL registers and the INTTYPEH / INTTYPEL registers. Usually, interrupt assignment is done once during system initialization and does not affect interrupt latency.

Interrupt assignment is the first of three steps required to enable an interrupt source, and this is done at chip integration. The second step is to program the source to generate interrupt requests. The final step is to enable the interrupt inputs in the core by clearing the normal interrupt disable (I) and/or the fast interrupt disable (F) bits in the program status register (CPSR).

43.4.4 Enabling Interrupts Sources

There are two methods of enabling or disabling interrupts in the AITC. The first method is directly reading the INTENABLEH / INTENABLEL registers, logically OR or BIT CLEAR these registers with a generated masks, then writing back to the INTENABLEH / INTENABLEL registers.

The second method is performing an atomic write to the source number to INTENNUM register. The AITC will decode this 6 bit register and enable one of the 64 interrupt sources. The AITC will automatically generate a “one hot” enable mask and logically OR this mask to the correct INTENABLEH or INTENABLEL register. To disable interrupts is exactly the same except the source number is written to the INTDISNUM register.

43.4.5 Controlling Bus Arbitration with AITC

The AITC has some logic to rise the priority level of the ARM core when either a fast or normal interrupt occurs. When a fast interrupt occurs and the FIAD bit is set, the AITC will assert the aitc_block_arb signal. When a normal interrupt occurs and the NIAD bit is set, the AITC will assert the aitc_block_arb signal. This signal will rise the priority level of the ARM core, so that it has priority of all other alternate masters. This signal will not stop the current alternate master transfer instead will prevent/limit future bus arbitration away from the ARM core.

The AITC includes some logic in the ABFLAG and ABFEN bits. The ABFLAG bit indicates the AITC is currently asserted the aitc_block_arb signal to the bus arbitration logic. The ABFEN bit changes the ABFLAG from a read-only status bit to a “sticky” write-1-to-clear status bit.

43.4.6 Typical Interrupt Entry Sequences

The following is a typical pipeline sequence for the ARM7TDMI-S core when a normal interrupt occurs. Assuming single cycle memories, it takes approximately 6 clocks from the acknowledgement of the normal interrupt within the ARM7TDMI-S until the first opcode of the interrupt routine is fetched.

Table 43-23. Typical Hardware Accelerated Normal Interrupt Entry Sequence

ADDR	TIME										
	-2	-1	0	1	2	3	4	5	6	7	8
	nIRQ assert		nIRQ ack.								
Last ADDR before nIRQ	Fetch	Dec	Exec	Link	Adjust						
+4 / +2		Fetch	Dec								
+8 / +4			Fetch								
0x0000_0018				Fetch	Dec	Exec	Data	Wrbk			
+4					Fetch	Dec					
+8						Fetch					
Vector Table							Vector				
n/a											
nIRQ Routine									Fetch	Dec	Exec
+4										Fetch	Dec
+8											Fetch

The following is a typical pipeline sequence for the ARM7TDMI-S core when a fast interrupt occurs, assuming that the FIQ service routine begins at 0x0000_001C and single cycle memories.

Table 43-24. Typical Fast Interrupt Entry Sequence

ADDR	TIME					
	-2	-1	0	1	2	3
	nFIQ assert		nFIQ ack.			
Last ADDR before nFIQ	Fetch	Dec	Exec	Link	Adjust	
+4 / +2		Fetch	Dec			
+8 / +4			Fetch			
0x0000_001c				Fetch	Dec	Exec
+4					Fetch	Dec
+8						Fetch

43.4.7 Writing Reentrant Normal Interrupt Routines

The AITC can be used to create a reentrant normal interrupt system. This enables preempting of lower priority level interrupts by higher priority level interrupts. This requires a small amount of software support and overhead.

1. Push the link register (LR_irq) on to the stack (SP_irq)
2. Push the saved status register (SPSR_irq) on to the stack
3. Read the current value of NIMASK and push this value on to the stack
4. Read current priority level via NIVECSR
5. Interrupts of the equal or lesser priority than the current priority level should be masked via the NIMASK register by writing value from NIVECSR
6. Clear the I bit in the ARM7TDMI-S core via a MSR / MRS command sequence (now a higher priority normal interrupt can preempt a lower priority one)
   Also change the operating mode of the core to System Mode from IRQ mode
7. Push System Mode link register (LR) on to the stack (SP_user)
8. The traditional interrupt service routine is now included
9. Pop System Mode link register (LR) from the stack (SP_user)
10. Set I bit in the ARM7TDMI-S core via a MSR / MRS command sequence (thus disabling all normal interrupts)
    Also change the operating mode of the core to IRQ Mode from System mode
11. Pop the original value of normal interrupt mask and write to the NIMASK register
12. The saved status register should be popped from the stack (SP_irq)
13. The link register should be popped from the stack into the PC
14. Return from nIRQ

NOTE:

Steps 1, 2, 13, and 14 are automatically done by most C compilers and are included for completeness.

Chapter 44
ARM7TDMI-S Watch Point Module (AWPT)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	4/1/2002	John Oakley	Added note about Data Patching and Peripheral Registers Added two port descriptions0.2
0.2	05/07/03		Updated for LTE specification release.

44.1 Introduction

The ARM7TDMI-S Watch Point Module is a 32-bit peripheral which can be used to either patch source code routines or fix data tables in the Read-Only Memory (ROM). The AWPT supports up to 64 addresses which can be patched where 4 are any location within memory and 60 are dedicated to ROM patching. Alternatively, 16 of the 64 locations can be used as a 1-word data fix. Alternatively, the 4 generic locations can be used to generate a break point event to the ARM7TDMI-S.

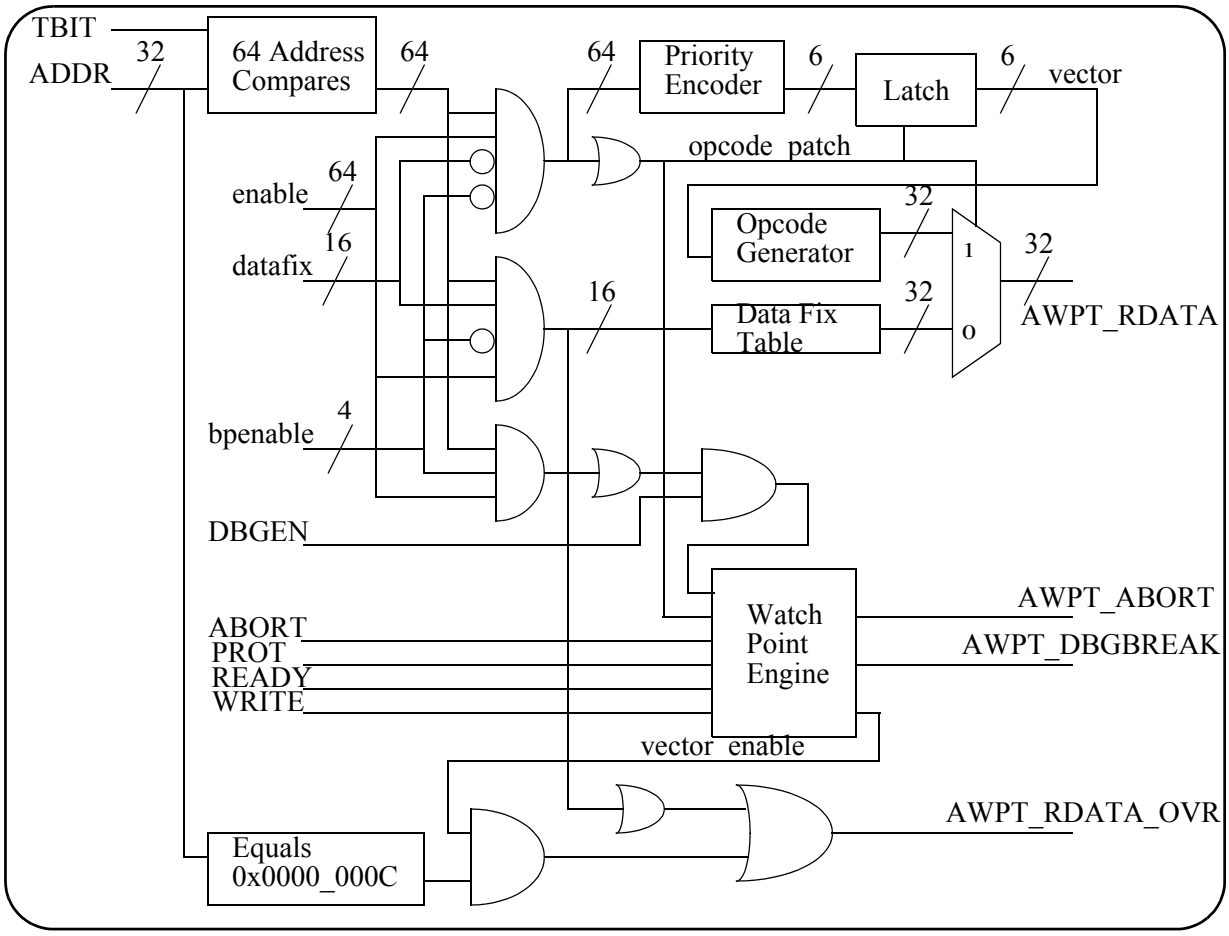


Figure 44-1. AWPT Block Diagram

- Supports up to 64 watch pointed addresses: 4 of which are generic and 60 are dedicated to ROM patching
- Supports 3 opcode patch modes: PC Relative or SP Relative or software controlled
- Hardware accelerated entry into associated patch routines
- Patch routine addresses stored to area efficient, multiply mapped RAM
- Supports one word data fixes for up to 16 memory locations
- Supports break pointing ARM7TDMI-S core for up to 4 memory locations
- Supports up to 954 opcode patch vectors in PC Relative mode using Vector Table offset feature

44.1.1 Opcode Patching

When an address match occurs and opcode patching is selected, a prefetch ABORT is inserted into the ARM7TDMI-S pipeline. If this prefetch ABORT is executed by the ARM7TDMI-S core, the ARM7TDMI-S core will fetch an instruction from location 0x0000_000C. The LSB of the address is ignored by the AWPT module for all three modes of operation. There are there different methods of opcode patching, one fully software controlled and two hardware accelerated modes. In all modes, the AWPT generates the prefetch ABORT which stops the normal program flow to allow software patching.

44.1.1.1 Software Controlled Opcode Patching Mode

When the prefetch ABORT exception occurs, the software must read the AWPTSR to determine if the AWPT generated the prefetch ABORT and which watch point event occurred. The software must then determine which watch point service routine to execute.

This mode of operations does not use any registers or resources from the ARM7TDMI-S core and is not limited in the number of vectors supported. The AWPT comparator still only has 64 addresses to compare at any one time. The disadvantage of this mode, is the necessity for additional software and the time consumed for every patch.

44.1.1.2 Program Counter (PC) Relative Opcode Patching Mode

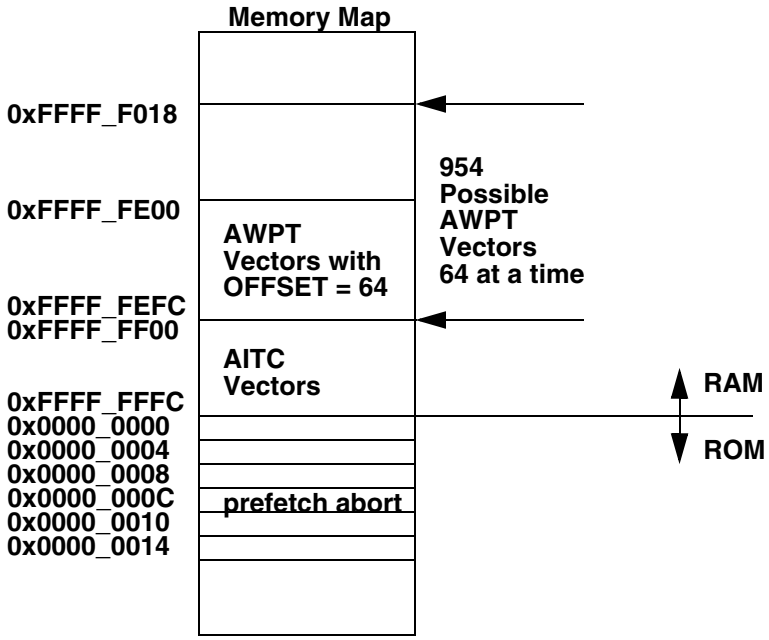


Figure 44-2. Example Memory Map For Opcode Patching With High AITC Vectors

If the Watch Point Module generated this ABORT, the read data bus is overwritten with the opcode "LDR PC, [PC,#-(276 - 4*(vector index) + 4*(vector table offset)]" when 0x0000_000C is fetched. Assuming the Vector Table Offset is 64 (reset value), this causes the ARM7TDMI-S core to load the Program Counter (PC) with a vector from a table of 64 vectors located at 0xFFFF_FE00 to 0xFFFF_FEFF; more specifically the PC is loaded with the vector located at 0xFFFF_FE00 + 4*(vector index). The Vector Table Offset allows for multiple vector tables to be stored in the high memory space compacted together. More specifically, changing the Vector Table Offset will move the vector table to 0xFFFF_FF00 - 4*(vector table offset) to 0xFFFF_FFFF - 4*(vector table offset).

This mode of operation does not use any registers or resources from the ARM7TDMI-S core but is limited to 1018 vectors and uses memory tables mapped to high memory locations. If the AITC vectors are also located in high memory 0xFFFF_FF00 to 0xFFFF_FFFF, these location should be reserved for the AITC. This would limit the number of watch point vectors to 954.

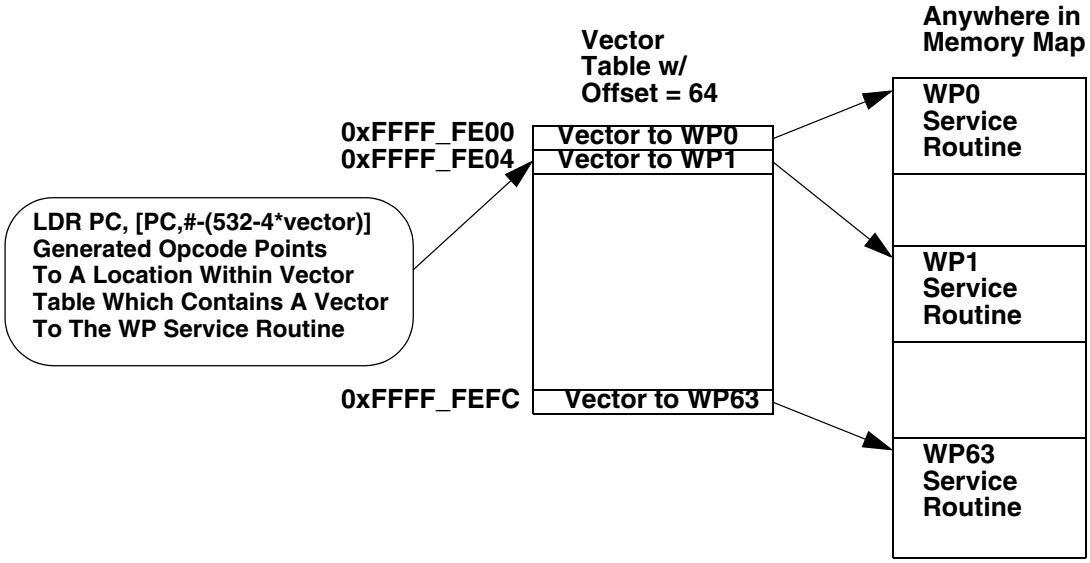


Figure 44-3. AWPT Vector Table Details in PC Relative Mode With OFFSET=64

44.1.1.3 Stack Pointer (SP) Relative Opcode Patching Mode

If the Watch Point Module generated this ABORT, the read data bus is overwritten with the opcode "LDR PC, [SP,#(4*(vector index))]" when 0x0000_000C is fetched. This causes the ARM7TDMI-S core to load the Program Counter (PC) with a vector from a table of 64 vectors located at SP+0x00 to SP+0xFF; more specifically the PC is loaded with the vector located at SP + 4*(vector index).

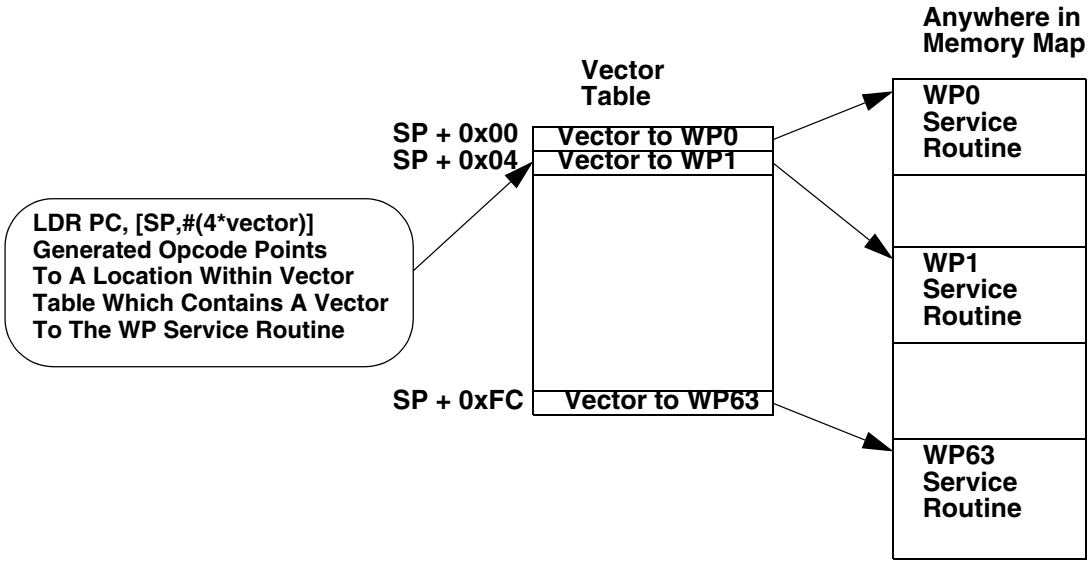


Figure 44-4. AWPT Vector Table Details in SP Relative Mode

This mode of operation uses the ABORT Stack Pointer as a pointer to the vector table, so the SP can not be used for its normal operation. The advantage is that the vector table can now be located anywhere is memory and any number of vector tables can be used. The disadvantage is that the ABORT stack pointer is not longer available for data and prefetch ABORT handling functions.

44.1.2 Data Fixing

When an address match occurs and data fixing is selected, the AWPT replaces the “bad” data with associated data stored in the register file. Entire words of data are replaced, so if byte or half-word access occurs on a “data fix” location the entire data word is replaced. The two LSBs of the address are ignored in the data fix mode of operation.

This mode of operation only affects the read data path and does not affect the actual memory writes.

NOTE:

The AWPT will not data fix any internal memory mapped registers. This is done by blocking the generation of the data fix event if the 8 MSBs of haddr are equal to 0x2.

44.1.3 ARM7TDMI-S Break Pointing

When an address match occurs, the operation is an opcode fetch from the ARM7TDMI-S core, ARM7TDMI-S break pointing is selected, and the DBGEN signal is asserted, then the AWPT tries to preform a break point event but first must check the value of BPCOUNT. If BPCOUNT is zero and a watch point triggers a break point event, the AWPT_DBGBREAK signal will be asserted to the ARM7TDMI-S core putting the core into debug mode. If BPCOUNT is non-zero and a break point event occurs, the value stored in BPCOUNT is decremented.

NOTE:

The BPCOUNT register and AWPT_DBGBREAK signal are only modified on opcode fetches from the ARM7TDMI-S core. Opcode fetches from alternate masters and data fetches from the ARM7TDMI-S core are ignored.

44.2 AWPT Signals

Table 44-1. AWPT Module Pin List

Pin Name	Direction	Description
Clock and Reset Signals		
hclk	In	Clock. This signal is the main clock input to the AWPT. It should be connected to the same source as the mcu_clk signal provided to the ARM7TDMI-S core.
hreset_b	In	Reset. This active-low input signal resets the AWPT. It should be tied to the same source as the nRESET signal to ARM7TDMI-S core.
ARM7TDMI-S Native Bus Signals		

Table 44-1. AWPT Module Pin List (Continued)

<table>
<tr><th>Pin Name</th><th>Direction</th><th>Description</th></tr>
<tr><td>dbgen</td><td>In</td><td>Debug Enable. This active-high input signal indicates the ARM7TDMI-S is enabled for debug mode. When this signal is deasserted, the AWPT will not generate a break point event to the core. This signal is only used for the break point generation logic and does not affect the opcode patching or the data fixing logic.</td></tr>
<tr><td>haddr[31:0]</td><td>In</td><td>Address Bus. This input bus specifies the address for the memory access. It is used to index the register file and in automatic vectoring of the prefetch abort.</td></tr>
<tr><td>hprot[1:0]</td><td>In</td><td>Transfer Protection Mode. This input bus encodes information about the transfer. This is used to detect whether opcode fetches or data accesses are occurring and whether the access is a privileged one.
<table>
<tr><th colspan="2">hprot</th><th rowspan="2">Mode</th><th rowspan="2">Opcode / Data</th></tr>
<tr><th>1</th><th>0</th></tr>
<tr><td>0</td><td>0</td><td>User</td><td>Opcode</td></tr>
<tr><td>0</td><td>1</td><td>User</td><td>Data</td></tr>
<tr><td>1</td><td>0</td><td>Privileged</td><td>Opcode</td></tr>
<tr><td>1</td><td>1</td><td>Privileged</td><td>Data</td></tr>
</table></td></tr>
<tr><td>hready</td><td>In</td><td>Memory Transfer Acknowledge. This active-high input signal indicates that the memory controller has completed the current access. This signal is used to stretch the RDATA masking and overwriting. It is also used for the ABORT pipeline mirror within the Watch Point Engine.</td></tr>
<tr><td>hmaster[2:0]</td><td>In</td><td>R-AHB Master. This input bus specifies which master has control of the R-AHB and initiated the current transfer. When the input equals 000, then the ARM7TDMI-S core has the bus.</td></tr>
<tr><td>hsel</td><td>In</td><td>AWPT Module Select. This active-high input signal indicates that the ARM7TDMI-S core is accessing a AWPT register. This signal is decoded from the MSBs of the haddr bus.</td></tr>
<tr><td>hsize[1:0]</td><td>In</td><td>Transfer Size. This input bus encodes the size of the transfer.
<table>
<tr><th colspan="2">hsize]</th><th rowspan="2">Transfer Width</th></tr>
<tr><th>1</th><th>0</th></tr>
<tr><td>0</td><td>0</td><td>Byte</td></tr>
<tr><td>0</td><td>1</td><td>Half-word</td></tr>
<tr><td>1</td><td>0</td><td>Word</td></tr>
<tr><td>1</td><td>1</td><td>Reserved</td></tr>
</table></td></tr>
</table>

Table 44-1. AWPT Module Pin List (Continued)

<table>
<tr><th>Pin Name</th><th>Direction</th><th>Description</th></tr>
<tr><td>htrans[1:0]</td><td>In</td><td>Transfer Type. This input bus encodes the type of the transfer using the bus.
<table>
<tr><th colspan="2">htrans</th><th rowspan="2">Cycle Type</th><th rowspan="2">Transfer Type</th></tr>
<tr><th>1</th><th>0</th></tr>
<tr><td>0</td><td>0</td><td>I</td><td>Internal Cycle</td></tr>
<tr><td>0</td><td>1</td><td>C</td><td>Coprocessor Cycle</td></tr>
<tr><td>1</td><td>0</td><td>N</td><td>Nonsequential Cycle</td></tr>
<tr><td>1</td><td>1</td><td>S</td><td>Sequential Cycle</td></tr>
</table></td></tr>
<tr><td>hwdata[31:0]</td><td>In</td><td>Write Data Bus. This input bus specifies the data for the memory access. It is used to program the register file.</td></tr>
<tr><td>hwrite</td><td>In</td><td>Write Enable. This active-high input signal specifies the direction of the transfer. When HIGH, the memory access is a write cycle.</td></tr>
<tr><td colspan="3">Watch Point Module Signals</td></tr>
<tr><td>awpt_abort</td><td>Out</td><td>AWPT Transfer Abort. This active-high output signal is the logical OR of the ABORT signal and the ABORT signal generated by the AWPT. This is used to tag the memory access as “bad” and begin the prefetch ABORT sequence.</td></tr>
<tr><td>awpt_disabled</td><td>Out</td><td>AWPT Disabled. This active-high output signal which indicates that the AWPT is completely disabled by setting the DIS bit of the AWPTCNTL register.</td></tr>
<tr><td>awpt_dbgbreak</td><td>Out</td><td>AWPT Debug Breakpoint. This active-high output signal indicates that a hardware break point has been generated for the ARM7TDMI-S core. The core will enter debug state upon assertion of this signal. This signal will not be asserted if the DBGEN signal is negated.</td></tr>
<tr><td>awpt_hready</td><td>Out</td><td>AWPT Transfer Acknowledge. This active-high output signal indicates that AWPT has completed a register access. This signal is driven out with AWPT_RDATA and AWPT_RDATA_OVR when a register access occurs.</td></tr>
<tr><td>awpt_rdata[31:0]</td><td>Out</td><td>AWPT Read Data Bus. This output bus is the read data bus from the AWPT. It either contains the output from a register read, output from the opcode generator, or output from the data fix table.</td></tr>
<tr><td>awpt_rdata_ovr</td><td>Out</td><td>AWPT Read Data Bus Overwrite. This active-high output signal that the normal read data bus should be overwritten by the AWPT. When asserted, the AHB_MUX block should multiplex off the normal read data bus from the memory controller and use the AWPT’s read data bus.</td></tr>
</table>

44.3 Watch Point Module Programming Model

44.3.1 Register Summary

The AWPT module has 86 registers.

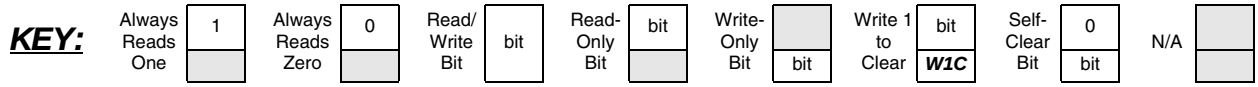


Table 44-2. Watch Point Module Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWPTD15 ($2A8A_00B4)	R	DATA15																															
	W																																
. . .																																	
AWPTD0 ($2A8A_00F0)	R	DATA0																															
	W																																
AWPTCNTL ($2A8A_00F4)	R	MODE		DIS	SCREN	0	0	OFFSET										DATAFIX															
	W																																
AWPTENH ($2A8A_00F8)	R	ENABLE[63:32]																															
	W																																
AWPTENL ($2A8A_00FC)	R	ENABLE[31:0]																															
	W																																
AWPTA0 ($2A8A_0100)	R	ADDR0																															0
	W																																
AWPTA1 ($2A8A_0104)	R	ADDR1																															0
	W																																
AWPTA2 ($2A8A_0108)	R	ADDR2																															0
	W																																
AWPTA3 ($2A8A_010C)	R	ADDR3																															0
	W																																
AWPTA4 ($2A8A_0110)	R	0	0	0	0	0	0	0	0	ADDR4																							0
	W																																
. . .																																	
AWPTA63 ($2A8A_01FC)	R	0	0	0	0	0	0	0	0	ADDR63																							0
	W																																
AWPTBRPT ($2A8A_0200)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BPCOUNT								0	0	0	0	BPENABLE			
	W																																
AWPTBASEA ($2A8A_0204)	R	BASEADDR								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																																
AWPTSR ($2A8A_0208)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SW	WP	0	0	0	0	0	0	0	0	0	0	WPVECTOR					
	W																																

All of these registers are single cycle access as the AWPT sits on the native bus of the ARM7TDMI-S.

44.3.2 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various AWPT registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

The watch point data registers (AWPTD15 through AWPTD0) store the data to use for the 16 1-word data fix events. Each register is associated with a watch point trigger. To patch data, the associated watch point must be enabled; data fixing must be selected and break pointing must be disabled for associated watch point. When the data patch event occurs, the AWPT will assert AWPT_RDATA and AWPT_RDATA_OVR with the correct data value from these registers at the appropriate time until HREADY is asserted by the memory controller.

These registers are located on the ARM7TDMI-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode. These registers should be updated with 32-bits stores only.

AWPTD15 through AWPTD0 — Watch Point Data Register 15 through Watch Point Data Register 0 — **$2A8A_00B4 through $2A8A_00F0**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	DATAX[31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	DATAX[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-3. AWPTD0 through AWPTD15 Description

Name	Description	Settings
DATAX Bits 31-0	**Watch Point Data —** Stores the data used for the associated 1-word data fix. The values stored within these registers will not affect the writes to the memory system. They only overwrite the read data bus when a data fix event occurs. If any part of the 1-word data fix is read, then the entire word is replaced. Therefore, a byte or half-word read will cause the AWPT to replace the entire word.	

The watch point control register (AWPTCNTL) contains the operating mode of the AWPT, the vector table offset number, and the data fix enable register. The three opcode patching modes of the AWPT are Software Controlled, PC Relative, and SP Relative. The data fix enable register controls whether or not the associated watch point event will preform a data fix or source code patch.

These registers are located on the ARM7TDMI-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode. These registers should be updated with 32-bits stores only.

AWPTCNTL Watch Point Control Register **$2A8A_00F4**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	MODE		DIS	SREN			OFFSET									
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	DATAFIX															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-4. AWPTCNTL Description

Name	Description	Settings
MODE Bits 31-30	**Opcode Patch Mode —** Controls watch point opcode patching vectoring mode: either PC Relative or SP Relative. In Software Controlled Mode, the AWPT only generates the prefetch ABORT for opcode patch events and does not overwrite the RDATA bus. In PC Relative Mode, the watch point service routine vector table is based off the Program Counter using the following opcode: LDR PC, [PC,#-(276 - 4*(vector) + 4*(offset))] In SP Relative Mode, the watch point service routine vector table is based off the Stack Pointer using the following opcode: LDR PC, [SP,#(4*(vector))]	**MODE[1:0]** / **Function** 0 0 — Opcode Patching Disabled 0 1 — Software Controlled Mode 1 0 — PC Relative Mode 1 1 — SP Relative Mode
DIS Bit 29	**AWPT Disable--** This bit, when set, disables all AWPT operations. This bit is used to enable secure operations.	0 = Does not affect any AWPT functions 1 = Disable all AWPT functions: break point generation, data fixing, and opcode patching
SREN Bit 28	**Secure ROM Enable Overwrite--** This write once bit, when set, enables the secure ROM even if the security_enable port is inactive. **NOTE:** SCREN bit should only be set by an opcode that is fetched from unprotected space.	0 = Port security_enable determines if the secure ROM is enabled 1 = Secure ROM is enabled
Bit 27-26	Reserved	N/A

Name	Description	Settings
OFFSET Bits 25-16	**Vector Table Offset —** Controls which vector table to use in PC Relative Mode. Each table can represent another s/w context and contains a different set of watch point routine vectors. This offset is a pointer to the watch point service routine associated with Watch Point 0.	In PC Relative mode, the AWPT supports up to 1018 vectors but only 64 contiguous vectors at a time. This is further limited to 954 vectors, if the AITC has its vectors in high memory.
DATAFIX Bits 15-0	**Data Fix Enable —** Controls whether the Any of these locations can be used as a trigger a source code patch routine or as a 1-word data fix.	0 = Watch Point event triggers a opcode patch 1 = Watch Point event triggers a data fix

The watch point enable register high (AWPTENH) and watch point enable register low (AWPTENL) control whether or not the associated watch point address can trigger a watch point event. A watch point trigger can then be used for opcode patching, data fixing, or break pointing.

These registers are located on the ARM7TDMI-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode. These registers should be updated with 32-bits stores only.

AWPTENH Watch Point Enable Register High **$2A8A_00F8**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	ENABLE[63:48]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	ENABLE[47:32]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

AWPTENL Watch Point Enable Register Low **$2A8A_00FC**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	ENABLE[31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	ENABLE[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-5. AWPTENH/AWPTENL Descriptions

Name	Description	Settings
ENABLE Bits 31-0 Bits 31-0	**Enable Watch Point —** This bit enables the corresponding watch point address to trigger a watch point event. Bit 63 is associated with ADDR63,...	0 = Watch Point disabled 1 = Watch Point enabled, AWPT will trigger a opcode patch, data fix, or break point event upon matching of the associated address

The watch point address registers (AWPTA0 through AWPTA63) store the memory addresses where opcode patching begins, data fixing occurs, or break pointing occurs. The watch point address registers AWPTA0 through AWPTA3 are each 31 bits wide and can point to any memory location. The watch point address register AWPTA4 through AWPTA63 are each 23 bits wide and dedicated to one 16 Mbyte memory space. The 8 MSBs of these 60 watch points is located in the AWPTBASEA register. 1-word data fixing can only be used on the watch points 0 through 15, the 4 generic watch points and the first 12 of the dedicated watch points.

These registers are located on the ARM7TDMI-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode. These registers should be updated with 32-bits stores only.

AWPTA0 Watch Point Address Register 0 **$2A8A_0100**
AWPTA1 Watch Point Address Register 1 **$2A8A_0104**
AWPTA2 Watch Point Address Register 2 **$2A8A_0108**
AWPTA3 Watch Point Address Register 3 **$2A8A_010C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	ADDRX[31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	ADDRX[15:1]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-6. AWPTA0 through AWPTA3 Description

Name	Description	Settings
ADDRX Bits 31-1	**Watch Point Address —** Indicates the memory address to be watched. Any of these locations can be used as a trigger a opcode patch routine, a 1-word data fix, or a break point event.	
Bit 0	Reserved	N/A

AWPTA4 through AWPTA63	Watch Point Address Register 4 through Watch Point Address Register 63	**$2A8A_0110 through $2A8A_01FC**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
									ADDRX[23:16]							
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	ADDRX[15:1]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-7. AWPTA4 through AWPTA63 Description

Name	Description	Settings
Bit 31-24	Reserved	N/A
ADDRX Bits 23-1	**Watch Point Address —** Indicates the memory address to be watched. Any of these locations can be used as a trigger a opcode patch routine. Only ADDR4 through ADDR15 can be used as a trigger for a 1-word data fix. The full 32-bit address are generated by concatenating the BASEADDR value to the MSBs of the associated watch point address.	
Bit 0	Reserved	N/A

The break point enable register (AWPTBRPT) controls whether or not the associated watch point event will preform a break point event to the ARM7TDMI-S core. There is also a mechanism for counting watch point events before a break point event is generated. When a watch point event occurs and the break point counter is equal to zero, a break point will be generated for the ARM7TDMI-S and the core will enter debug state and wait for the external EmbeddedICE controller to exit from debug state.

This register should be all zeros in a non debug environment. This register is held in the RESET state when DBGEN is negated which provides a hardware interlock to prevent undesired break point events.

These registers are located on the ARM7TDMI-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode. These registers should be updated with 32-bits stores only.

AWPTBRPT Watch Point Breakpoint Enable Register **$2A8A_0200**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	BPCOUNT												BPENABLE			
TYPE	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rw	rw	rw	rw	rwm	rwm	rwm	rwm
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-8. AWPTBRPT Description

Name	Description	Settings
Bit 31-16	Reserved	N/A
BPCOUNT Bits 15-8	**Break Point Event Counter —** Controls the number of watch point triggers that occur before generating a break point to the ARM7TDMI-S. Note: This register will auto decrement for each watch point trigger, i.e. on any opcode fetch from the ARM7TDMI-S core to any location marked as a break point event. When this register is equal to zero, the next watch point trigger will generate a break point to the ARM7TDMI-S.	0 = The next Watch Point event will generate a break point event to the ARM7TDMI-S core 1+ = The next Watch Point event will decrement this register
Bit 7-4	Reserved	N/A
BPENABLE Bits 3-0	**Break Point Enable —** Controls whether the Any of these locations can be used as a trigger a break point event to the ARM7TDMI-S core.	0 = Watch Point event triggers a opcode patch or a data fix 1 = Watch Point event triggers a break point event

Note: This register is held in RESET state when DBGEN is negated which provides a hardware interlock to prevent undesired break point events. When DBGEN is asserted to the ARM7TDMI-S core, it is assumed that the core is allowed to enter debug mode and that an external controller is present to manipulate the core.

Note: This register is only modified on opcode fetches from the ARM7TDMI-S core. Opcode fetches from alternate masters and data fetches from the ARM7TDMI-S core are ignored.

The watch point base address register (AWPTBASEA) stores the base location of the dedicated 16 Mbyte memory space. These 8 bits are concatenated onto the 60 dedicated watch points. These concatenated registers are then used by the address comparators to monitor for watch points.

These registers are located on the ARM7TDMI-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode. These registers should be updated with 32-bits stores only.

AWPTBASEA Watch Point Base Address Register **$2A8A_0204**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	BASEADDR															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-9. AWPTBASEA Description

Name	Description	Settings
BASEADDR Bits 31-24	**ROM Base Address —** Indicates the base address of the dedicated 16 Mbyte memory space. These 8 bits are concatenated onto the 60 dedicated watch point address registers.	
Bits 23-0	Reserved	N/A

The watch point status register (AWPTSR) indicates the current state of the AWPT and the vector index of the most recent watch point event.

These read-only registers are located on the ARM7TDMI-S native bus, accessible in 1 cycle, and can only be accessed to in privileged mode.

AWPTSR Watch Point Status Register **$2A8A_0208**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
															SW	WP
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	w1c	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
											WPVECTOR					
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-10. AWPTSR Description

Name	Description	Settings
Bits 31-18	Reserved	N/A
SW Bit 17	**Simultaneous Watch Points Indicator —** Indicates that multiple watch point events occurred at same time. The AWPT will enforce a prioritization scheme as defined and will set this status flag. Writing a 1 to this bit will clear this bit.	0 = no watch point event collisions 1 = multiple watch point events occurred simultaneously
WP Bit 16	**Watch Point Occurred Indicator —** Indicates that the last prefetch ABORT was generated by the AWPT. This bit is used before the software controlled Opcode Patching Mode of operation.	0 = AWPT did not generate last prefetch ABORT, therefore was a normal prefetch ABORT 1 = AWPT generated last prefetch ABORT indicating a watch point
Bits 15-6	Reserved	N/A
WPVECTOR Bits 5-0	**Watch Point Vector —** Indicates vector index for the most recent opcode patch event. This register will only indicate the opcode patch events and ignores data fix and break point events.	0 = Watch Point 0 occurred 1 = Watch Point 1 occurred ... 63 = Watch Point 63 occurred

44.4 ARM7TDMI-S Watch Point Module Operation

44.4.1 Typical AWPT Opcode Patching

The following is a typical pipeline sequence for the ARM7TDMI-S core when an opcode patch occurs. It takes approximately 8 clocks from the watch point address until the first opcode of the opcode patch routine is fetched.

Table 44-11. Typical AWPT Opcode Patching Sequence

ADDR	TIME										
	0	1	2	3	4	5	6	7	8	9	10
ABORT	1	0	0	0	0	0	0	0	0	0	0
WP ADDR	Fetch	Dec	Exec	Link	Adjust						
+4 / +2		Fetch	Dec								
+8 / +4			Fetch								
0x0000_000c				Fetch	Dec	Exec	Data	Wrbk			
+4					Fetch	Dec					
+8						Fetch					
Vector Table							Vector				
n/a											
WP Routine									Fetch	Dec	Exec
+4										Fetch	Dec
+8											Fetch

The AWPT opcode patch routine must be written intelligently and know how return to normal software flow after the opcode patch is complete. The software should exit from ABORT mode back into the previous operating mode, including thumb state.

44.4.1.1 Typical Entry to Opcode Patching Routine

The typical first line of the opcode patch routine is:

SUBS PC, PC, #4

This command will exit ABORT mode, restore the saved status register to the current status register, and start executing the next location of code. This next location of memory can be either an ARM instruction or a THUMB instruction, as the SUBS command modifies the PC; this causes the ARM7TDMI-S core to flush and refill the instruction pipeline in the current operating mode.

44.4.1.2 Typical Exit from Opcode Patching Routine

The opcode patching routine must know where to reenter the normal software flow. Also, if the ‘SUBS PC, PC, #4’ command was not used at the begin of the routine, then the operating mode of the processor should be restored before returning back to the normal software flow.

44.4.2 AWPT Event Priority

The AWPT module has the following priority of events:

1. Break point event
2. Data fix event
3. Opcode patch event

If multiple events occurs at the same time, the AWPT will assert the SW bit in the control register (AWPTCNTL).

If multiple events of the same priority level occur at the same time, then the AWPT will choose the watch point based on the one with the highest source number. For example, the AWPT is setup to data fix the same location with watch points 4 and 7, then watch point 7 will be used.

The AWPT_RDATA bus multiplexer has the following priority:

1. Register read
2. Data fix
3. Opcode generation for 0x0000_000C

44.4.3 Typical Software Context Switching and AWPT

To enable N opcode patches on a context switch, the software should write N+3 contiguous words to the AWPT. These writes correspond to the last N address vectors (AWPTA63 through AWPTA(64-N)), the two enable registers (AWPTENH and AWPTENL), and the control register (AWPTCNTL).

To enable N opcode patches and M data fix on a context switch, the software should write two blocks, N+3 contiguous words (for opcode patching) and 2*M contiguous words (for data fixing), to the AWPT.

44.4.4 Alternate Masters and AWPT

The AWPT module does not know which master initiated the current transfer for opcode patching or data fixing. Therefore, the AWPT responses to an alternate master exactly the same as the ARM7TDMI-S core for opcode patching and data fixing. This includes ABORT generation on appropriate opcode fetches and data fixing normal accesses. The break point generation will not occur if an alternate master initiated the transfer.

NOTE:

In Neptune, it is understood that all normal alternate masters will generate only data accesses and as such will not generate opcode fetches. This means that the AWPT will not generate the prefetch ABORT on alternate master accesses but will preform data fixing and break point generation.

44.4.4.1 Low Power Mode and AWPT

The AWPT modules does not know about low power modes. Any time the ARM7TDMI-S core complex would be clocked (any alternate master or ARM7TDMI-S access), the AWPT would also be clocked.

44.4.5 Normal Memory ABORTs and AWPT

The AWPT monitors the ABORTs generated from other modules. Within the AWPT is a mirror of the prefetch abort pipeline of the ARM7TDMI-S core. This mirror is updated every opcode fetch, ARM or THUMB mode.

If a prefetch abort occurs in either of the two previous opcode fetches before a AWPT opcode patching event, the AWPT will not generate the ABORT signal to the ARM7TDMI-S core. When the ARM7TDMI-S core processes a non-AWPT prefetch abort, the AWPT will not mask the RDATA bus, the normal prefetch abort opcode.

If a prefetch ABORT occurs at the same time as a AWPT opcode patching event or afterwards, the AWPT will generate an ABORT to the ARM7TDMI-S core and will mask the RDATA bus when the prefetch abort opcode is read. Therefore, the AWPT will “claim” to have generated the prefetch abort even if some other abort source also generated a prefetch abort.

Chapter 45
MCU Memories (MCUMEM)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	03/28/02	Lloyd Khuc	Update Memory Map for MCU RAM & ROM.
0.2	06/1/02	Lloyd Khuc	Update Memory Map for MCU RAM (LTE) and removed Table44-2.
0.3	6/17/02	Steve Benzel	Updated Block Diagrams and Module Pin List.
0.4	6/18/02	Lloyd Khuc	Updated the Section 44.4 & 44.5.
0.5	05/07/03		Updated for LTE specification release.

45.1 Introduction

The Neptune has 2 large memory blocks, a 2Mbit SRAM and a 14Mbit diffusion ROM all taking up approximately half the die area. The SRAM arrays include laser fuse invoked redundancy to lower the yield impact of fab defectivity.

The Large RAM and ROM interface to the ARM AHB bus through a memory controller, MCTL. It provides select and clocking functions to the memories as well as memory BIST.

Figure 45-1 shows the memory complex which includes MCTL (memory controller), 512KB SRAM & 1792KB ROM.

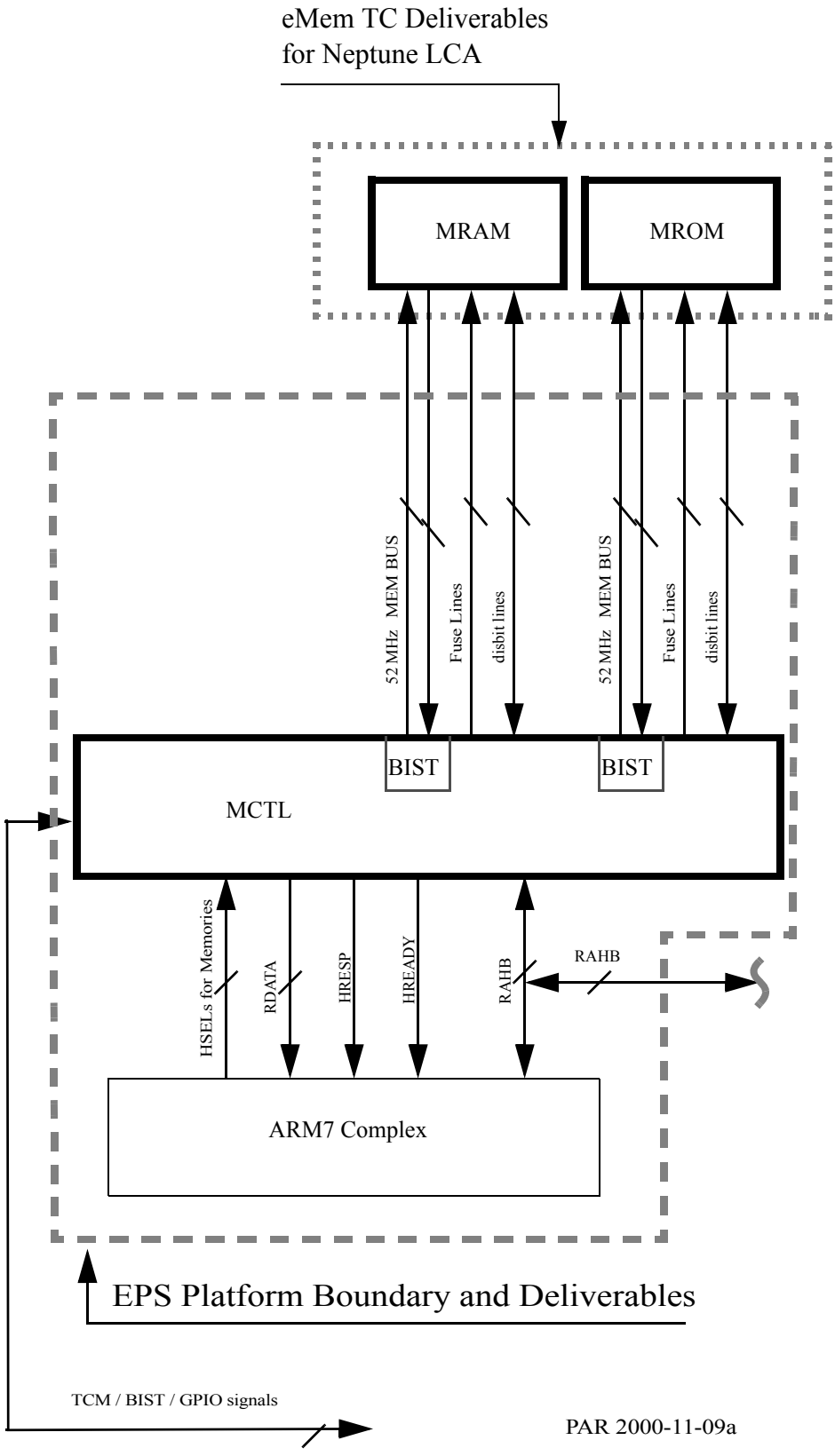


Figure 45-1. Neptune Memory Complex Block Diagram

45.2 Memory Controller (MCTL)

45.2.1 AHB/Memory Interface

Memory controller (a slave) essentially serves as the interface between the masters (ARM7, GEM, DMAC & TCM) and the sub slaves (256K 8bits RAM and 1792K 8bits ROM). It consists of (i) Control signal circuitry which generates memory select signals (such as “asel” & “dsel”) for all the memories, (ii) Control circuitry which sets the default values of all the memory mapped registers during the system reset stage, (iii) circuitry to generate reset and fuse sense signals to all the memories during the system reset, (iii) Control circuitry to turn on the appropriate I/O of memories, which are connected in “Big-Endian” mode, when the accessed data size is less than a full word, (iv) BIST engine which is designed to test either 512K 8bits RAM or 1792K 8bits ROM at clock frequencies of that of the microcontroller, ARM7 and (v) Memory Mapped Registers whose information is used purely during test & characterization stage.

Memory Mapped Registers are used to store the following: (i) Information relating to the fuses. i.e. whether they are blown or not, (ii) Information to adjust the current flowing in the fuse detect circuitry depending on the mode of operation (i.e. test mode or normal mode), (iii) Information to adjust the write and read control timings for the sub-slave, (iv) Information to change the sense amplifier reference (precharge) inputs from both being high to being low, (v) Information from BIST related to the “failure” if any, (vi) information to either “turn on” or “turn off” the self timed word line drivers, (vii) information to initiate the “column disconnect” information to flow from the memories to the tester during DMA mode

45.2.2 Block Diagram

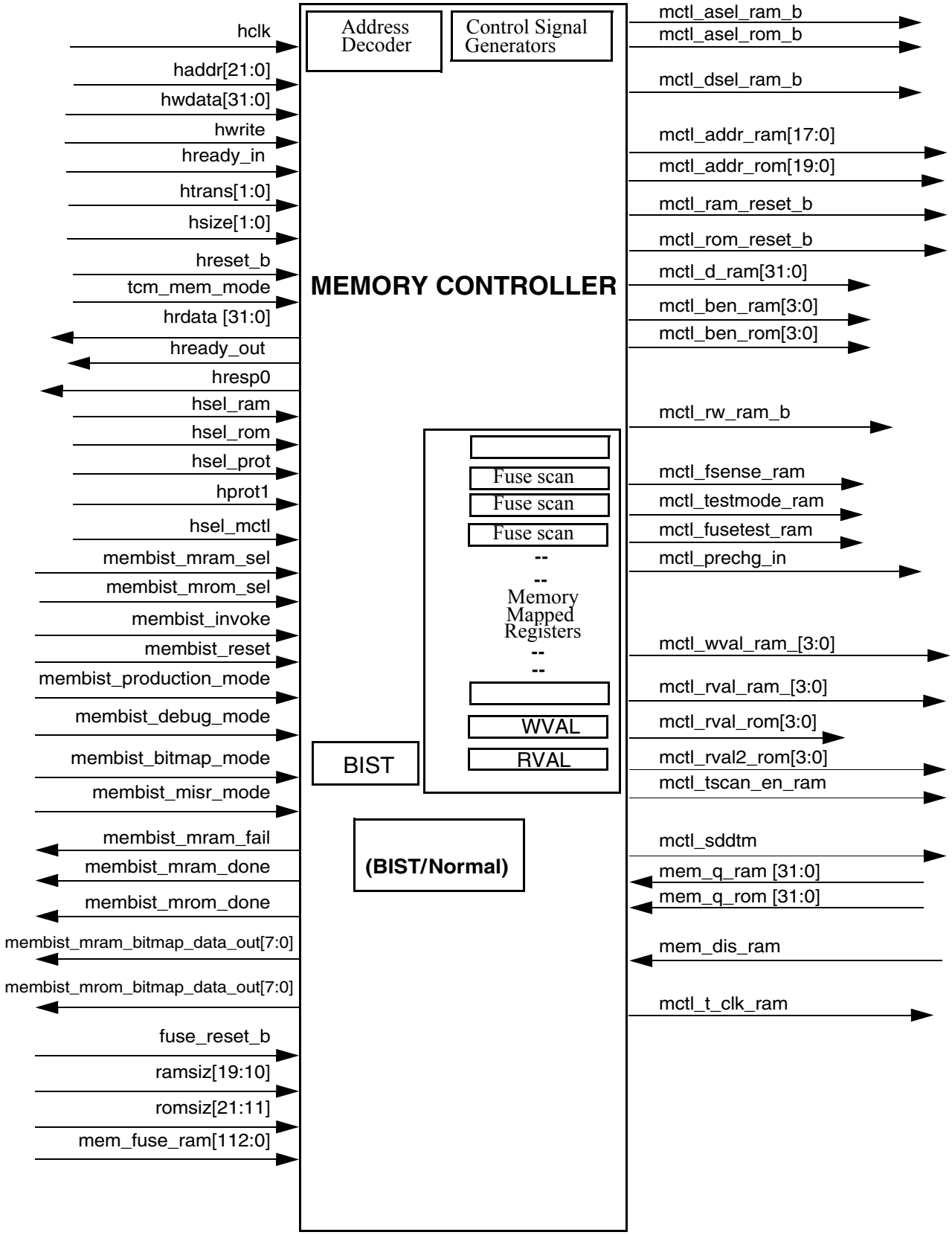


Figure 45-2. Memory Controller Block Diagram

45.2.3 MCUMEM Module Pin List

Table 45-1 is a list of the MCUMEM module pins.

Table 45-1. MCUMEM Module Pin List

Pin Name	Direction	Description	Source
hclk	Input		System clock control module
hreset_b	Input		System
hready_in	Input		Bus Master (i.e. Microcontroller, DMAC, GEM & TCM)
htrans[1:0]	Input	When the signal value is set to 10 (“N cycle”) the memory controller initiates an operation. Memory controller doesn’t involve in any activity for any other state of this signal	Bus Master (i.e. Microcontroller, DMAC, GEM & TCM)
haddr[21:0]	Input		Bus Master (i.e. Microcontroller, DMAC, GEM & TCM)
hwrite	Input Active high		Bus Master (i.e. Microcontroller, DMAC, GEM & TCM)
hsize[1:0]	Input	Three possible states (i) Full word (“10”), (ii) Halfword (“01”) and (iii) Byte (“00”)	Bus Master (i.e. Microcontroller, DMAC, GEM & TCM)
hwdata[31:0]	Input		Bus Master (i.e. Microcontroller, DMAC, GEM & TCM)
tcm_mem_mode	Input		
hresp0	Output	MCTL Error Response	Memory Controller
hprot1	Input	HPROT[1]	
hsel_ram hsel_rom hsel_mctl	Input	A combinational logic signal generated by a central decoding unit, using the MSB of haddr[31:0] to indicate that the slave being accessed by the bus master is the 512K/256K RAM, 1792K ROM & MCTL.	Central Decoding Unit
hsel_prot	Input	Indicates supervisor protected memory space	Central Decoding Unit
mem_dis_ram	Input	MCU RAM DisBit	512k/256k X 8 ram
mem_fuse_ram[112:0]	Input	A signal providing information to indicate whether intended banks of fuses are blown to replace the defective columns/rows with redundant columns/rows.	512k/256K X 8 ram
membist_invoke	Input		GPIO

Table 45-1. MCUMEM Module Pin List

Pin Name	Direction	Description	Source
membist_reset	Input	Memory BIST engine reset signal	GPIO
membist_mram_fail	Input	Fail flag from MRAM BIST engine	MRAM BIST Engine
membist_mram_done	Input	Done flag from MRAM BIST engine	MRAM BIST Engine
membist_mrom_done	Input	Done flag from MROM BIST engine	MROM BIST Engine
membist_mram_sel	Input	MRAM BIST engine enable	TCM
membist_mrom_sel	Input	MROM BIST engine enable	TCM
membist_mram_bitmap_data_out	Output	Bitmap data from MRAM BIST engine	MRAM BIST Engine
membist_mrom_bitmap_data_out	Output	Bitmap data from MROM BIST engine	MROM BIST Engine
membist_production_mode	Input	MBIST production test mode. Refer to the Test Guide for more information.	TCM
membist_debug_mode	Input	MBIST debug test mode. Refer to the Test Guide for more information.	TCM
membist_bitmap_mode	Input	MBIST bitmap test mode. Refer to the Test Guide for more information.	TCM
membist_misr_mode	Input	MBIST misr test mode. Refer to the Test Guide for more information.	TCM
mctl_ram_reset_b	Output	Early system reset from CCM	Microcontroller
mctl_rom_reset_b	Output	Early system reset from CCM	Microcontroller
mctl_asel_ram_b mctl_asel_rom_b	Output	A combinational logic (between a qualified “htrans” and hsel_sub slave)) feeding to all the sub slaves (i.e. Large SRAM and ROM). This signal is used by the sub-slave to generate the required “aclk”.	Memory Controller
mctl_addr_ram[17:0]	Output	RAM word address from either the RAM BIST or haddr[19:2]	Memory Controller
mctl_addr_rom[19:0]	Output	ROM word address from either the ROM BIST or haddr[21:2]	Memory Controller
mctl_rw_ram_b	Output	MCU RAM read not write	Memory Controller

Table 45-1. MCUMEM Module Pin List

<table>
<tr><th>Pin Name</th><th>Direction</th><th>Description</th><th>Source</th></tr>
<tr><td>mctl_ben_ram[3:0]
mctl_ben_rom[3:0]</td><td>Output</td><td>This signal directs the memory to appropriate portion of the data bus when the instruction length is less than the full word (“32 bits”). Only the following 7 states of possible 16 states are used.

<table><tr><th>Mode</th><th>Bit String</th><th>wdata / rdata</th></tr><tr><td>Byte</td><td>0001</td><td>[7:0]</td></tr><tr><td>Byte</td><td>0010</td><td>[15:8]</td></tr><tr><td>Byte</td><td>0100</td><td>[23:16]</td></tr><tr><td>Byte</td><td>1000</td><td>[31:24]</td></tr><tr><td>Half Word</td><td>0011</td><td>[15:0]</td></tr><tr><td>Half Word</td><td>1100</td><td>[31:16]</td></tr><tr><td>Word</td><td>1111</td><td>[31:0]</td></tr></table>
mctl_ben is generated using the signals “ap_mem_siz” and “ap_mem_addr[1:0]”</td><td>Memory Controller</td></tr>
<tr><td>hrdata[31:0]</td><td>Output</td><td>This is primarily the buffered version of the data supplied by the large RAM, ROM.</td><td>Memory Controller</td></tr>
<tr><td>mctl_dsel_ram_b</td><td>Output</td><td>Registered versions of mctl_asel_sub_slave of the previous clock cycles during a write operation. Defaults to “0” for a read cycle.</td><td>Memory Controller</td></tr>
<tr><td>mctl_d_ram[31:0]</td><td>Output</td><td>MCU RAM data in.</td><td>Memory Controller</td></tr>
<tr><td>mem_q_ram[31:0]</td><td>Input</td><td>Data supplied by 512k/256k_ram</td><td>512k/256k_ram</td></tr>
<tr><td>mem_q_rom[31:0]</td><td>Input</td><td>Data supplied by 1792k rom</td><td>1792k_rom</td></tr>
<tr><td>hready_out</td><td>Output</td><td></td><td>Memory Controller</td></tr>
<tr><td>mctl_wval_ram [3:0]</td><td>Output</td><td>Controls the write signal timing internal to the RAM. The default value “1000” is set by memory controller during the reset.</td><td>Memory Controller</td></tr>
<tr><td>mctl_rval_ram [3:0]</td><td>Output</td><td>Controls the read signal timing internal to the RAM. The default value “1000” is set by memory controller during the reset.</td><td>Memory Controller</td></tr>
<tr><td>mctl_rval_rom [3:0]</td><td>Output</td><td>Controls the read signal timing internal to the ROM. The default value “1000” is set by memory controller during the reset.</td><td>Memory Controller</td></tr>
</table>

Table 45-1. MCUMEM Module Pin List

Pin Name	Direction	Description	Source
mctl_rval2_rom [3:0]	Output	Controls the read signal timing, in addition to the above, internal to the ROM. The default value is set by memory controller during the reset.	Memory Controller
mctl_fusetest_ram	Output	Used to control the bias current in the fuse sense circuitry to sense whether a fuse is blown or not. When low, low bias currents are applied. e.g. Test mode.	Memory Controller
fuse_reset_b	Output		Memory Controller
mctl_fsense_ram	Output	During the system reset stage, a short timed pulse, which the fuse sense circuitry uses to sense whether a fuse is blown or not. During the reset stage, this signal is generated using a combinational logic of early_reset_b and ~hreset_b. During test mode, this signal could be toggled to “0” or “1” by a memory mapped register.	Memory Controller
mctl_testmode_ram	Output	To indicate which test mode sub-slave is in. normal mode (0) or test mode (“1”). The default state is set to “normal mode (“0”)” by the memory controller when nreset is asserted. Self timed world line driver circuitry is either enabled or disabled during normal mode and test mode respectively	Memory Controller
mctl_prechg_in	Output	To indicate whether the global data lines and data lines should be precharged to high (when the signal is asserted) or low (when the signal is negated).	Memory Controller
mctl_t_clk_ram	Output	A set of clock pulses (tester clock) being sent to memories to scan out disabled column information to the tester	Memory Controller
mctl_tscan_en_ram	Output	To enable the above clock. A default value of ‘0” is set to this register by MCTL during the system reset. A value of “1” needs to written to by TCM to read out the column disconnect information during the consecutive 512 read cycles of this memory mapped register.	Memory Controller
mctl_sddtm	Output	MCU RAM Soft Data Defect test mode	Memory BIST Control
ramsiz [19:10]	Input	MCU RAM size selection; Refer to the Integration Guide for more information	Tie-offs at the Platform level
romsiz [21:11]	Input	MCU ROM size selection; Refer to the Integration Guide for more information	Tie-offs at the Platform leve

45.2.4 Wave Diagram

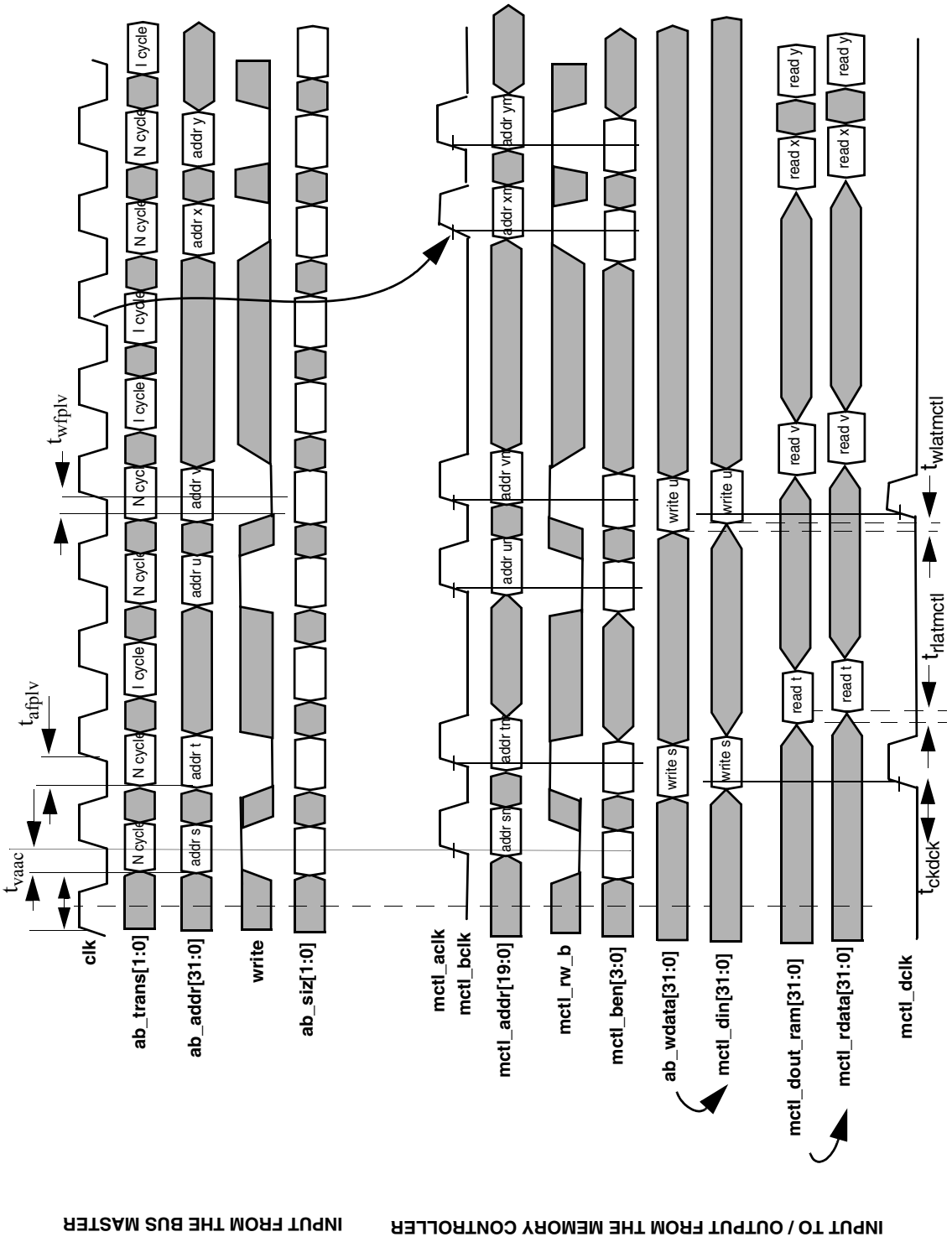


Figure 45-3. Read Write Waveform

45.3 MCTL Memory mapped registers

45.3.1 Register Descriptions

45.3.1.1 MCTL fuse sense 512K/256K RAM (MFSLRAM)

This single bit memory mapped register could be written and read by the TCM. The state of this memory mapped register is set to a default value of “0” by MCTL during the assertion of hreset_b and it could be toggled to either state by TCM after the reset. The state of the register is used to asynchronously feed the signal mctl_fsense_ram after the reset.

Location	Offset		31	30	29	28.24	23.17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MFSLRAM	0x0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EN
		W																						
reset																							0	

EN-
Both write and read feature of this bit is enabled for TCM.
State “0”: Current is not passing through the copper fuses located in the 512K RAM.
State “1”: Current is passing through the copper fuses located in the 512K RAM.

45.3.1.2 Reserved Register

Reading this location will return all 0.

Location	Offset		31	30	29	28.24	23.17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	0x4	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																						

45.3.1.3 MCTL fuse test 512K/256K RAM (MFTLRAM)

This single bit memory mapped register could be written and read by the TCM. The state of this memory mapped register is set to a default value of “0” by MCTL during the assertion of hreset_b and it could be toggled to either state by TCM after the reset. The state of the register is used to asynchronously feed the signal mctl_ftest_ram after the reset.

Location	Offset		31	30	29	28.24	23.17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MFTLRAM	0x8	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EN
		W																						
reset																							0	

EN-
Both write and read feature of this bit is enabled for TCM.
State “0”: A low current is passing through the copper fuses located in the 512K RAM if the state of MFSLRAM is “0”.
State “1”: An high current is passing through the copper fuses located in the 512K RAM if the state of MFSLRAM is “1”.

45.3.1.4 Reserved Register

Reading this location will return all 0.

Location	Offset		31	30	29	28.24	23.17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	0xC	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																						

45.3.1.5 MCTL test mode (MCTLTM)

This single bit memory mapped register could be written and read by any bus master. The state of this memory mapped register is set to a default value of “0” by MCTL during the assertion of hreset_b. The state of the register is asynchronously fed to the memory using the signal line mctl_testmode. This bit should be set to “1” during burn-in and certain memory tests that measure bit speed.

Location	Offset		31	30	29	28.24	23.17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCTLTM	0x10	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EN
		W																						
reset																							0	

EN-
Both write and read feature of this bit is enabled for TCM.
State “0”: Self timed circuitry for the word line drivers is “ON”
State “1”: Self timed circuitry for the word line drivers is “OFF”

45.3.1.6 MCTL Precharge MCTPHG

This single bit memory mapped register could be written and read by the TCM.The state of this memory mapped register is set to a default value of “1” by MCTL during the assertion of hreset_b. The state of the register is asynchronously fed to the memory using the signal mctl_prechg_in.

Location	Offset		31	30	29	28.24	23.17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCTPHG	0x14	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EN
		W																						
reset																							0	

EN-

Both write and read feature of this bit is enabled for TCM.
State “0”: Bitlines (BL & BLB), Data Lines (DL, DLB) and Global Data Lines (GDL, GDLB) are all precharged to “0”.
State “1”: Bitlines (BL & BLB), Data Lines (DL, DLB) and Global Data Lines (GDL, GDLB) are all precharged to “1”.

45.3.1.7 MCTL Write Val setting 512K RAM (WVALLRM)

This four bit memory mapped register could be written and read by the TCM. The state of this memory mapped register is set to a default value of “1000” by MCTL during the assertion of hreset_b. The state of the register is asynchronously fed to the memory using the signal mctl_wval_ram.

Location	Offset		31	30	29	28.24	23.17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WVALLRM	0x18	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WRITE VAL[3:0]			
		W																						
reset																				1	0	0	0	

WRITE VAL [3:0]-
All the sixteen different states could be written and read by TCM.

45.3.1.8 Reserved Register

Reading this location will return all 0.

Location	Offset		31	30	29	28.24	23.17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	0x1C	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																						

45.3.1.9 MCTL Read Val setting 512K/256K RAM (RVALLRM)

This four bit memory mapped register could be written and read by the TCM. The state of this memory mapped register is set to a default value of “1000” by MCTL during the assertion of hreset_b. The state of the register is asynchronously fed to the memory using the signal mctl_rval_ram.

Location	Offset		31	30	29	28.24	23.17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RVALLRM	0x20	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	READ VAL[3:0]			
		W																						
reset																				1	0	0	0	

READ VAL [3:0]-
All the sixteen different states could be written and read by TCM.

45.3.1.10 Reserved Register

Reading this location will return all 0.

Location	Offset		31	30	29	28.24	23.17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	0x24	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																						

45.3.1.11 MCTL Read Val setting1 1792K ROM (RVALROM)

This four bit memory mapped register could be written and read by the TCM. The state of this memory mapped register is set to a default value of “0000” by MCTL during the assertion of hreset_b. The state of the register is asynchronously fed to the memory using the signal mctl_rval_rom.

Location	Offset		31	30	29	28.24	23.17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RVALROM	0x28	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	READ VAL[3:0]			
		W																						
reset																				0	0	0	0	

READ VAL [3:0]-
All the sixteen different states could be written and read by TCM.

45.3.1.12 MCTL Read Val setting2 1792K RAM (RVAL2ROM)

This four bit memory mapped register could be written and read by the TCM. The state of this memory mapped register is set to a default value of “1000” by MCTL during the assertion of hreset_b. The state of the register is asynchronously fed to the memory using the signal mctl_rval2_rom.

Location	Offset		31	30	29	28.24	23.17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RVAL2ROM	0x2C	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	READ VAL[3:0]			
		W																						
reset																				1	0	0	0	

READ VAL [3:0]-
All the sixteen different states could be written and read by TCM.

45.3.1.13 Reserved Register

Reading this location will return all 0.

Location	Offset		31	30	29	28.24	23.17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	0x30	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																						

45.3.1.14 Reserved Register

Reading this location will return all 0.

Location	Offset		31	30	29	28.24	23.17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	0x34	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																						

45.3.1.15 Reserved Register

Reading this location will return all 0.

Location	Offset		31	30	29	28.24	23.17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	0x38	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																						

45.3.1.16 Reserved Register

Reading this location will return all 0.

Location	Offset		31	30	29	28.24	23.17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	0x3C	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																						

45.3.1.17 MCTL Fuse sense2 512K/256K RAM (MFSLRAM1)

This 32 bit virtual memory mapped register is written by the 512K/256K RAM when the memory comes out of reset (i.e. negation of hreset_b). The state of the register is set to either “1” or “0”depending on whether the fuse is blown or not. The contents of this register could be read by TCM.

Location	Offset		31	30	29	28.24	23.17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MFSLRAM1	0x40	R	FUSE_VAL[31:0]																					
		W																						

FUSE_VAL[31:0]-
The state of the register is set to either “1” or “0”depending on whether the fuse is blown or not.

45.3.1.18 MCTL Fuse sense2 512K/256K RAM (MFSLRAM2)

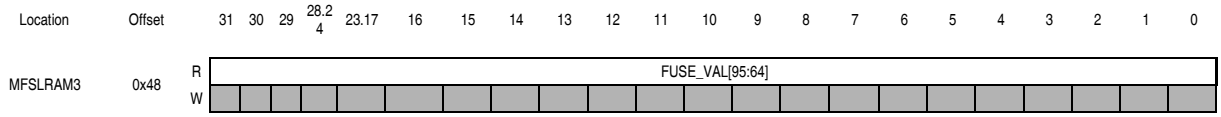
This 32 bit virtual memory mapped register is written by the 512K/256K RAM when the memory comes out of reset (i.e. negation of hreset_b). The state of the register is set to either “1” or “0”depending on whether the fuse is blown or not. The contents of this register could be read by TCM.

Location	Offset		31	30	29	28.24	23.17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MFSLRAM2	0x44	R	FUSE_VAL[63:32]																					
		W																						

FUSE_VAL[63:32]-
The state of the register is set to either “1” or “0”depending on whether the fuse is blown or not.

45.3.1.19 MCTL Fuse sense3 512K/256K RAM (MFSLRAM3)

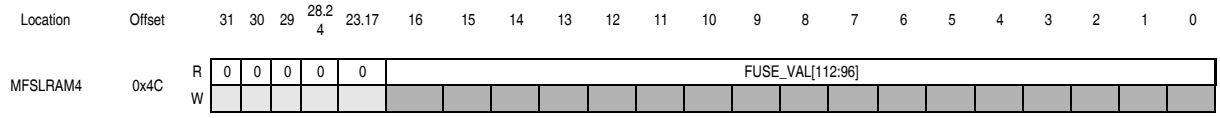
This 32 bit virtual memory mapped register is written by the 512K/256K RAM when the memory comes out of reset (i.e. negation of hreset_b). The state of the register is set to either “1” or “0”depending on whether the fuse is blown or not. The contents of this register could be read by TCM.



FUSE_VAL[95:64]-
The state of the register is set to either “1” or “0”depending on whether the fuse is blown or not.

45.3.1.20 MCTL Fuse sense4 512K/256K RAM (MFSLRAM4)

This 17 virtual bit memory mapped register is written by the 512K/256K RAM when the memory comes out of reset (i.e. negation of hreset_b). The state of the register is set to either “1” or “0”depending on whether the fuse is blown or not. The contents of this register could be read by TCM.



FUSE_VAL[112:96]-
The state of the register is set to either “1” or “0”depending on whether the fuse is blown or not.

45.3.1.21 MCTL tscan en 512K/256K RAM (MTSCLRAM)

This single bit memory mapped register is set to a default value of “0” by the memory mapped controller when the system comes out of reset (i.e. negation of hreset_b). The state of the register could be toggled to “1” by TCM, at that point onwards TCM is expected to generate 512 consecutive read request cycles (of maximum frequency of 20 MHz) corresponding to this memory mapped register. The data being read would actually come from the registers located within the 512K/256K memory. After the read operation is complete, the state of this memory mapped register could be set back to “0”

Location	Offset		31	30	29	28.24	23.17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MTSCLRAM	0x50	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EN
		W																						
reset																						0		

EN-
The state of memory mapped register could be toggled to either “0” or “1” by the TCM. When the state of the register is “1”, any read operation of this memory mapped register would actually a read of one of the 512 registers located within the memory.

45.3.1.22 Memory Map

Location	Offset		31	30	29	28.24	23.17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MFSLRAM	0x0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EN
		W																						
RESERVED	0x4	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																						
MFTLRAM	0x8	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EN
		W																						
RESERVED	0xC	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																						
MCTLTM	0x10	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EN
		W																						
MCTPHG	0x14	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EN
		W																						
WVALLRM	0x18	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WRITE VAL[3:0]			
		W																						
RESERVED	0x1C	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																						
RVALLRM	0x20	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	READ VAL[3:0]			
		W																						
RESERVED	0x24	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																						
RVALROM	0x28	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	READ VAL[3:0]			
		W																						
RVAL2ROM	0x2C	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	READ VAL[3:0]			
		W																						
RESERVED	0x30	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																						
RESERVED	0x34	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																						
RESERVED	0x38	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																						
RESERVED	0x3C	R						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																						

MFSLRAM1	0x40	R	FUSE_VAL[31:0]																				
		W																					
MFSLRAM2	0x44	R	FUSE_VAL[63:32]																				
		W																					
MFSLRAM3	0x48	R	FUSE_VAL[95:64]																				
		W																					
MFSLRAM4	0x4C	R	0	0	0	0	0	FUSE_VAL[112:96]															
		W																					
MTSCLRAM	0x50	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EN
		W																					

45.4 MCU RAM (MRAM)

The Neptune LTE has a 64K x 32 SRAM array. The RAM is connected to the AHB through the Memory Controller, MCTL. It is a 0 wait state memory running at 52MHz. Features include byte enables, support for Abort, adjustable bit sense timing, burn-in mode, and redundancy with all fuse values readable.

45.4.1 Block Diagram

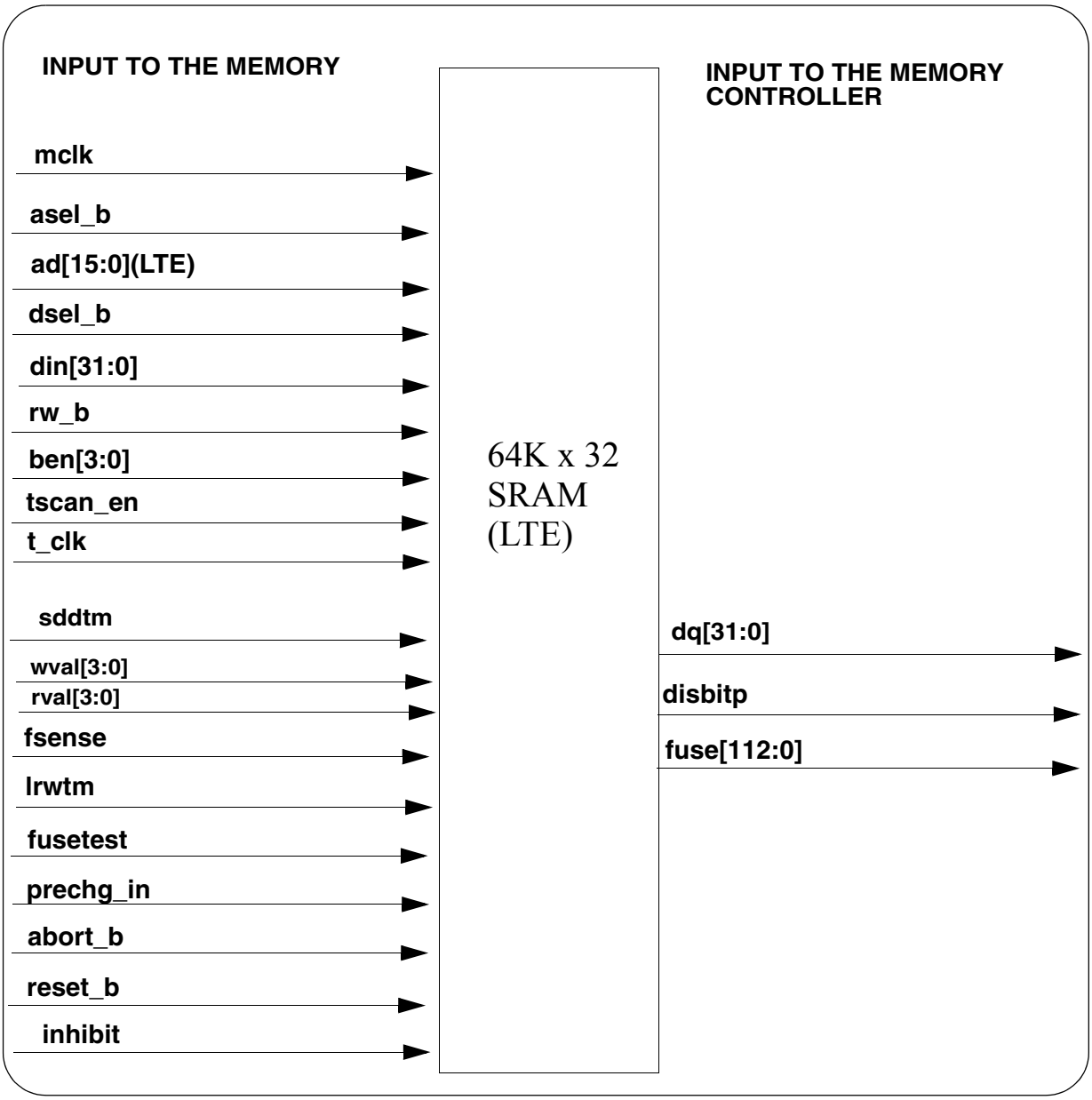


Figure 45-4. Neptune Large RAM Block signals

45.4.2 Pin Description

Table 45-2. Neptune SRAM Pin Description

Symbol	Type	Description
mclk	Input	main clock. 52MHz max. For neptune, input is the early clock from the CCM. Captures values on the rising edge.
ad[15:0] (LTE)	Input	Address Port: Captured on the rising edge of mclk.
asel_b	Input	The clock gating signal for the address/attributes
dsel_b	Input	The clock gating signal for the Data Input Port.
din[31:0]	Input	Data Input Port - Captured on the rising edge of mclk
rw_b	Input	Read/Write input - captured on the rising edge of mclk
ben[3:0]	Input	Byte Enable Control - Captured on the rising edge of mclk.
tscan_en	Input	Asynchronous input that enables the scan clock to scan out the values of the scan chain.
t_clk	Input	Clock input for the scan chain
disbitp	Output	Output of scan chain of column disconnect registers
rval[3:0]	Input	DC Input for programmable change of duration of sense time on the bitlines (2ns programming of 16 steps)
wval[3:0]	Input	DC Input for programmable duration of bit writing (2ns programming of 16 steps)
fsense	Input	During the system reset stage, a short timed pulse, which the fuse sense circuitry uses to sense whether a fuse is blown or not. During the reset stage, this signal is generated by the MCTL using a combinational logic of early_reset_b and ~hreset_b. During test mode, this signal could be toggled to “0” or “1” by a memory mapped register.
fuse[112:0]	Output	Outputs fuse values. These are valid ~3us after resetb_in goes high.
fusetest	Input	Asynchronous input that changes resistance requirement for a blown fuse.
sddtm	Input	Soft Defect Detect input asserted by BIST in certain BIST tests. rw_b must be high along with sddtm in order for it to work properly.
prechg_in	Input	To indicate whether the BL/BLB should be precharged to high (when the signal is asserted) or low (when the signal is negated).
lrwtm	Input	Long read/write test mode pin. A default value of “0” is set by memory controller during the reset operation. This signal disables the self timed write and read circuitry in the RAM. Also used in burn-in mode for proper bit stressing.
dq[31:0]	Output	Data Output Port.
abort_b	Input	Abort - Halts execution of a read or write cycle: Active low. Captured on the falling edge of mclk

Table 45-2. Neptune SRAM Pin Description

Symbol	Type	Description
reset_b	Input	Asynchronous reset.
inhibit	Input	Asynchronous, active high input to disable memory activity during scan tests. Puts Data Output bus in a logic 1 state.
V_{DD}	Supply	Core Power Supply.
V_{SS}	Supply	Ground.

45.4.3 Wave Diagram

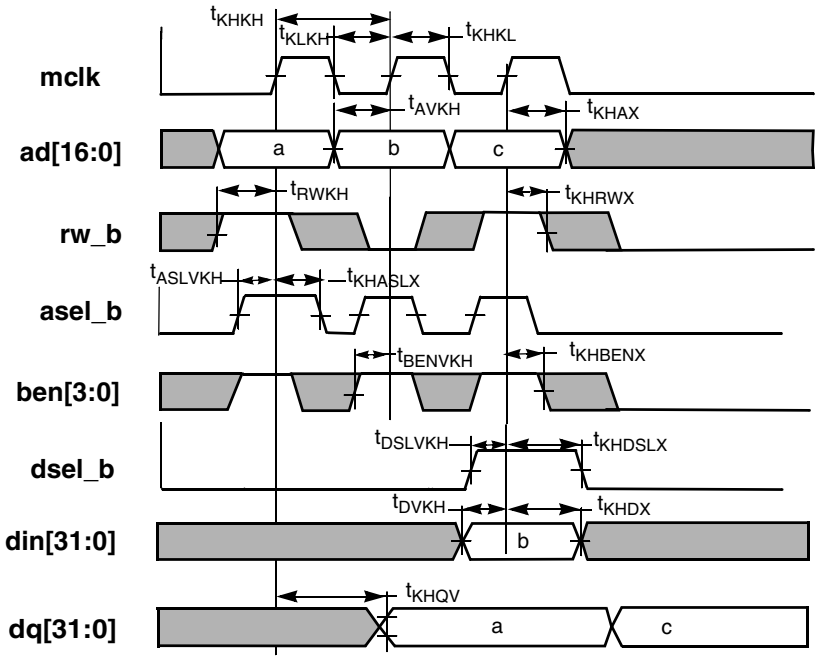


Figure 45-5. Read/Write Waveforms

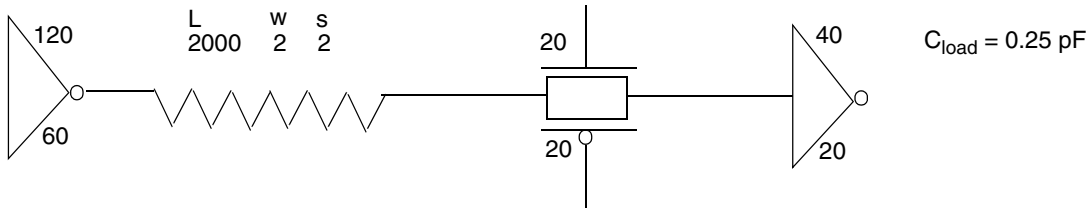
45.4.4 AC Operating Conditions and Chacteristics

(TA = 0 to 70C, Unless Otherwise Noted)

Table 2: Read/Write Cycle Timing

Parameter	Symbol	Min	Max	Unit	Notes
Cycle Time	t_{KHKH}	15	—	ns	
Clock Access Time	t_{KHQV}	—	15.5	ns	See the schematic below
Clock Low Pulse Width	t_{KLKH}	4	—	ns	
Clock High Pulse Width	t_{KHKL}	4	—	ns	
Clock High to Data Output Invalid	t_{KHQX}	5	—	ns	
Setup Times: asel_b dsel_b	t_{ASLVKH} t_{DSLVKH}	1		ns	1
Hold Times: asel_b dsel_b	t_{KHASLX} t_{KHDSLX}	1		ns	1
Setup Times:ad[16:0] rw_b ben[3:0] din[31:0]	t_{AVKH} t_{RWKH} t_{BENKH} t_{DVKH}	0	—	ns	1
Hold Times:ad[16:0] rw_b ben[3:0] din[31:0]	t_{KHAX} t_{KHRWX} t_{KHBENX} t_{KHDX}	2	—	ns	1

1) This is a synchronous device. All synchronous inputs must meet the specified setup and hold times with stable logic levels for ***ALL*** rising edges of clock (K) while the device is selected.



45.5 MCU ROM (MROM)

The Neptune has a 448K x 32 ROM array, diffusion programmed, which is connected to the AHB through the Memory Controller, MCTL. It is a 0 wait state memory running at 52MHz. Features include byte enables and Error Correction Logic for high yield.

45.5.1 Block Diagram

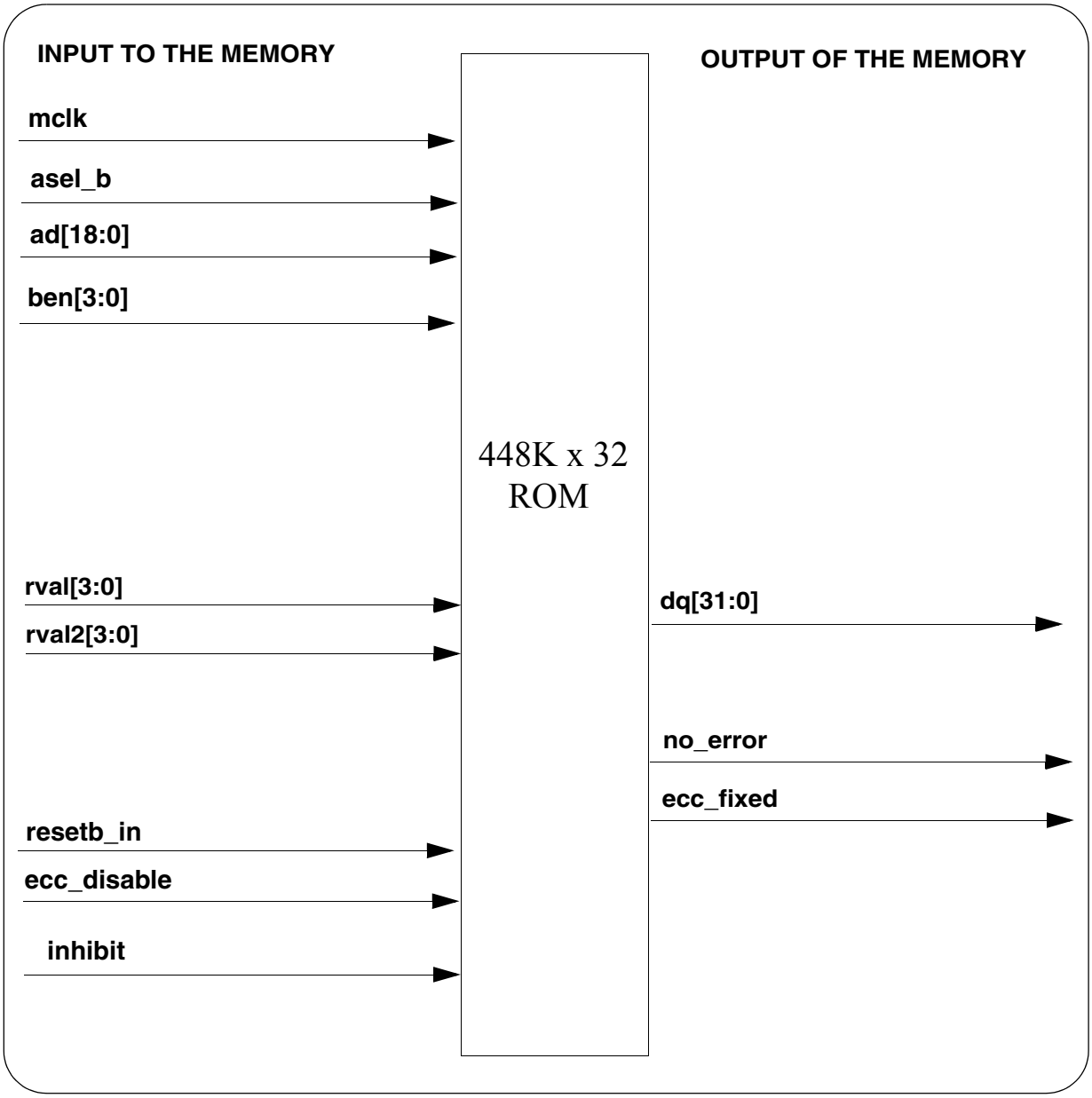


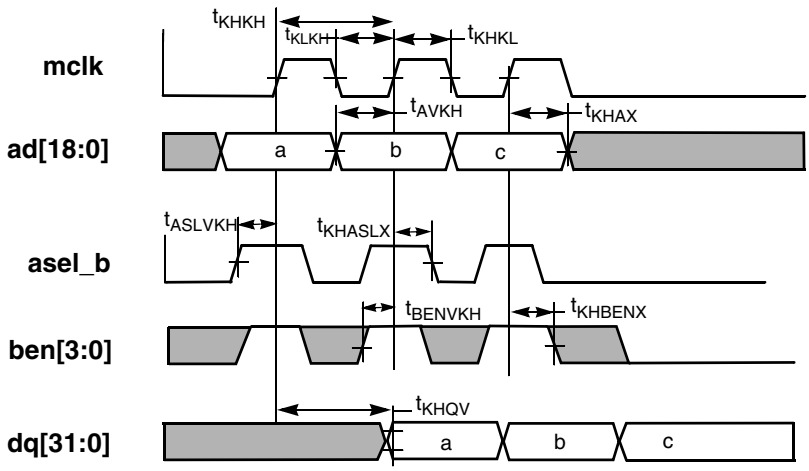
Figure 45-6. Neptune Large ROM Block signals

45.5.2 Pin Description

Table 45-3. Neptune ROM Pin Description

Symbol	Type	Description
mclk	Input	main clock. 52MHz max. For neptune, input is the early clock from the CCM.
ad[18:0]	Input	Address Port: Captured on the rising edge of mclk.
ben[3:0]	Input	Byte Enable Control - Captured on the rising edge of mclk.
dq[31:0]	Output	Data Output Port - Always active
asel_b	Input	The clock gating signal for the address/attributes
resetb_in	Input	Asynchronous reset.
no_error	Output	Outputs a high when the data being read does not have to be corrected by the ECC.
ecc_fixed	Output	Outputs a high when the data being read has been corrected by the ECC
ecc_disable	Input	Disables ECC circuitry for testing and bitmapping.
rval[3:0]	Input	DC Input for programmable timing delay between address latched to sense amp on. Default value 0000
rval2[3:0]	Input	DC Input for programmable timing delay of sense amp on to data being latched from the sense amp. Default value 1111
inhibit	Input	Asynchronous, active high input to disable memory activity during scan tests.
V_{DD}	Supply	Core Power Supply.
V_{SS}	Supply	Ground.

45.5.3 Wave Diagram



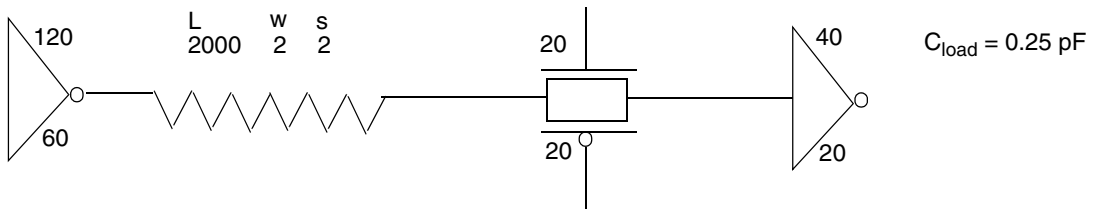
45.5.4 AC Operating Conditions and Chacteristics

(TA = 0 to 70C, Unless Otherwise Noted)

Table 3: Read Cycle Timing

Parameter	Symbol	Min	Max	Unit	Notes
Cycle Time	t_{KHKH}	11	—	ns	
Clock Access Time	t_{KHQV}	—	15	ns	
Clock Low Pulse Width	t_{KLKH}	4	—	ns	
Clock High Pulse Width	t_{KHKL}	4	—	ns	
Clock High to Data Output Invalid	t_{KHQX}	3	—	ns	
Setup Times: asel_b	t_{ASLVKH}	1		ns	1
Hold Times: asel_b	t_{KHASLX}	1		ns	1
Setup Times:ad[18:0] mctl_ben[3:0]	t_{AVKH} t_{BENKH}	0	—	ns	1
Hold Times:ad[18:0] mctl_ben[3:0]	t_{KHAX} t_{KHBENX}	2	—	ns	1

1) This is a synchronous device. All synchronous inputs must meet the specified setup and hold times with stable logic levels for ***ALL*** rising edges of clock (K) while the device is selected.



Chapter 46
IP-Bus Mux (IPMUX)

Revision History Table

Revision	Date	Author	Changes
1.0	03/06/02	Min Liao	Initial Release.
1.1	03/13/02	Min Liao	Update the AIPI module enable decode from Requirement doc.
1.2	04/01/02	Min Liao	Change the module HAC signal from “hacc” to “hac”
1.3	04/16/02	Min Liao	1. Replace ips_module_en[i] with modulename_module_en 2. Added the actual address for IP peripherals.
1.4	05/07/03		Updated for LTE specification release.

46.1 Introduction

The IP-bus Mux module (IPMUX) is located inside the Sea of Gates (SOG). The main function of IPMUX is to mux only one set of read and data attribute signals to ARM7 platform. It connects to 24 MCU peripheral modules (for MDI, it needs 4 extra module enables for its 4 MCU-access shared memory banks) on the IP-bus:

1. Clock Control Module (CCM),
2. Test Control Module (TCM),
3. Analog to Digital Logic Module (A2DIGL),
4. General Purpose I/O Module (GPIO),
5. Alternate Master Arbitor Module (AMARB)
6. Real-Time Clock Module (RTC).
7. Deep Sleep Module (DSM)
8. Subsriber Interface Module (SIM)
9. GPRS Encryption Module (GEM)
10. Real Time Reference (RTR)
11. L1 Timer (L1T)
12. Universal Serial Bus (USB)
13. Display Memory Access Controller (DMAC)
14. One wire Interface (OWIRE)
15. Hash Acceleration Module (HAC)
16. Universal Asynchronous Receiver/Transmitter (UART1)
17 Universal Asynchronous Receiver/Transmitter (UART2)
18. MCU/DSP Interface (MDI)
19. Keypad Port (KPP)
20. MCU External Interrupt Module (INT)
21. Enhanced General Purpose Timer (EGPT)
22. IC Identification Module (IIM)
23. Watchdog Timer (WDOG)
24. Multiple Queue Serial Peripheral Interface (MQSPI)

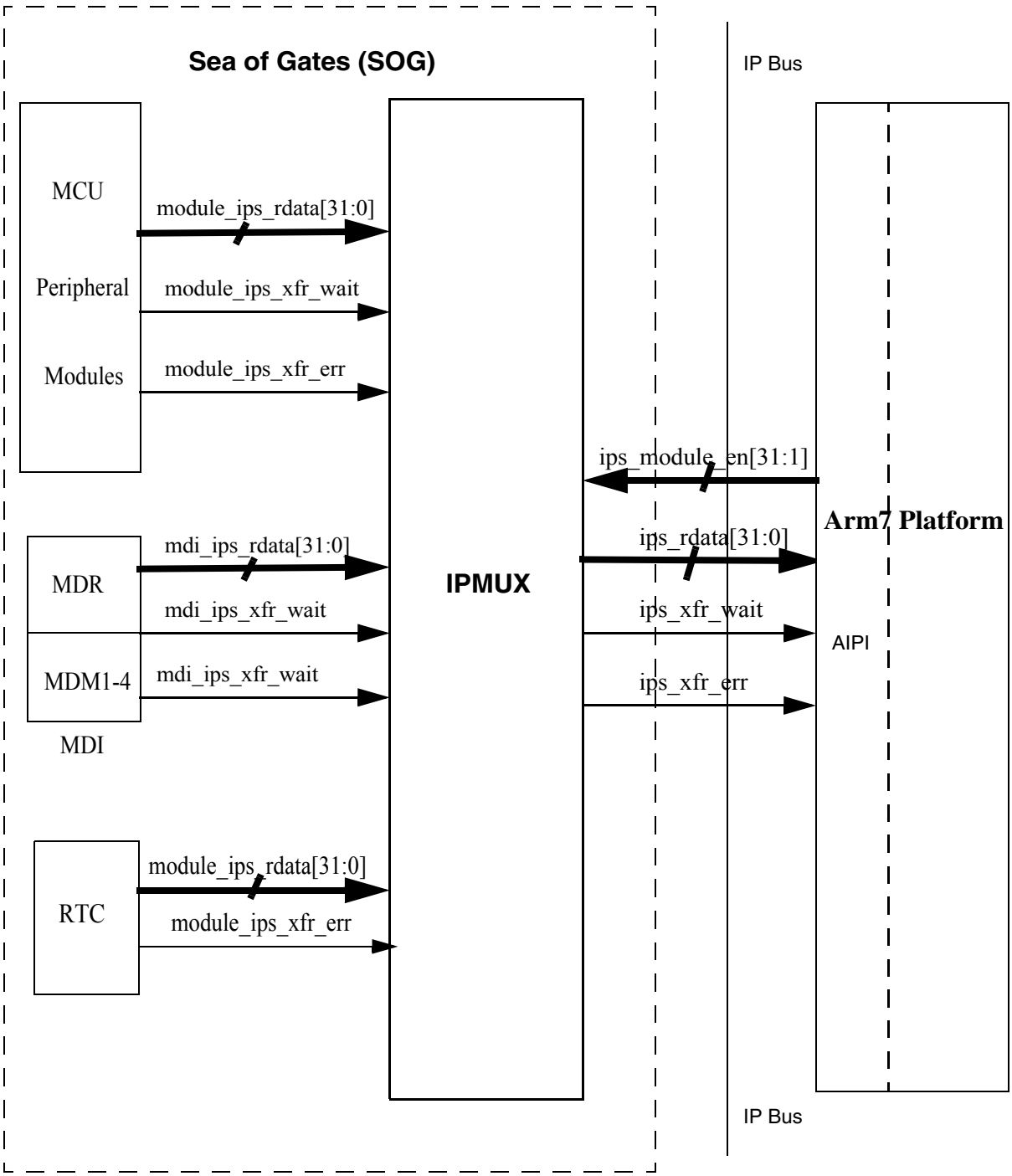


Figure 46-1. Block diagram

46.2 IPMux Module Pin List

Table 46-1 is a list of the IPMUX module pins.

Table 46-1. IPMUX Module Pin List

Pin Name	Direction	Description
ccm_ips_rdata[31:0]	Input	32-bit read data bus from Clock Control Module (CCM)
ccm_ips_xfr_wait	Input	Clock Control Module (CCM) will not complete transfer in current cycle
ccm_ips_xfr_err	Input	Data Error during transfer from Clock Control Module (CCM)
tcm_ips_rdata[31:0]	Input	32-bit read data bus from Test Control Module (TCM)
tcm_ips_xfr_wait	Input	Test Control Module (TCM) will not complete transfer in current cycle
tcm_ips_xfr_err	Input	Data Error during transfer from Test Control Module (TCM)
a2di_ips_rdata[31:0]	Input	32-bit read data bus from Analog to Digital Logic Module (A2DIGL)
a2di_ips_xfr_wait	Input	Analog to Digital Logic Module (A2DIGL) will not complete transfer in current cycle
a2di_ips_xfr_err	Input	Data Error during transfer from Analog to Digital Logic Module (A2DIGL)
gpio_ips_rdata[31:0]	Input	32-bit read data bus from General Purpose I/O Module (GPIO)
gpio_ips_xfr_wait	Input	General Purpose I/O Module (GPIO) will not complete transfer in current cycle
gpio_ips_xfr_err	Input	Data Error during transfer from General Purpose I/O Module (GPIO)
amarb_ips_rdata[31:0]	Input	32-bit read data bus from Alternate Master Arbitor Module (AMARB)
amarb_ips_xfr_wait	Input	Alternate Master Arbitor Module (AMARB) will not complete transfer in current cycle
amarb_ips_xfr_err	Input	Data Error during transfer from Alternate Master Arbitor Module (AMARB)
rtc_ips_rdata[31:0]	Input	32-bit read data bus from Real-Time Clock Module (RTC)
rtc_ips_xfr_err	Input	Data Error during transfer from Real-Time Clock Module (RTC)
dsm_ips_rdata[31:0]	Input	32-bit read data bus from Deep Sleep Module (DSM)
dsm_ips_xfr_wait	Input	Deep Sleep Module (DSM)will not complete transfer in current cycle

Table 46-1. IPMUX Module Pin List

Pin Name	Direction	Description
dsm_ips_xfr_err	Input	Data Error during transfer from Deep Sleep Module (DSM)
sim_ips_rdata[31:0]	Input	32-bit read data bus from Subscriber Interface Modue (SIM)
sim_ips_xfr_wait	Input	Subscriber Interface Modue (SIM) will not complete transfer in current cycle
sim_ips_xfr_err	Input	Data Error during transfer from Subscriber Interface Modue (SIM)
gem_ips_rdata[31:0]	Input	32-bit read data bus from GPRS Encryption Module (GEM)
gem_ips_xfr_wait	Input	GPRS Encryption Module (GEM) will not complete transfer in current cycle
gem_ips_xfr_err	Input	Data Error during transfer from GPRS Encryption Module (GEM)
rtr_ips_rdata[31:0]	Input	32-bit read data bus from Real Time Rererence (RTR)
rtr_ips_xfr_wait	Input	Real Time Rererence (RTR) will not complete transfer in current cycle
rtr_ips_xfr_err	Input	Data Error during transfer from Real Time Rererence (RTR)
l1t_ips_rdata[31:0]	Input	32-bit read data bus from L1 Timer (L1T)
l1t_ips_xfr_wait	Input	L1 Timer (L1T) will not complete transfer in current cycle
l1t_ips_xfr_err	Input	Data Error during transfer from L1 Timer (L1T)
usb_ips_rdata[31:0]	Input	32-bit read data bus from Universal Serial Bus (USB)
usb_ips_xfr_wait	Input	Universal Serial Bus (USB) will not complete transfer in current cycle
usb_ips_xfr_err	Input	Data Error during transfer from Universal Serial Bus (USB)
aeim_ips_rdata[31:0]	Input	32-bit read data bus from ARM External Interface Module (AEIM)
aeim_ips_xfr_wait	Input	ARM External Interface Module (AEIM) will not complete transfer in current cycle
aeim_ips_xfr_err	Input	Data Error during transfer from ARM External Interface Module (AEIM)
dmac_ips_rdata[31:0]	Input	32-bit read data bus from Display Memory Access Controller (DMAC)

Table 46-1. IPMUX Module Pin List

Pin Name	Direction	Description
dmac_ips_xfr_wait	Input	Display Memory Access Controller (DMAC) will not complete transfer in current cycle
dmac_ips_xfr_err	Input	Data Error during transfer from Display Memory Access Controller (DMAC)
owire_ips_rdata[31:0]	Input	32-bit read data bus from One Wire Interface (OWIRE)
owire_ips_xfr_wait	Input	One Wire Interface (OWIRE) will not complete transfer in current cycle
owire_ips_xfr_err	Input	Data Error during transfer from One Wire Interface (OWIRE)
hac_ips_rdata[31:0]	Input	32-bit read data bus from Hash Acceleration Module (HAC)
hac_ips_xfr_wait	Input	Hash Acceleration Module (HAC) will not complete transfer in current cycle
hac_ips_xfr_err	Input	Data Error during transfer from Hash Acceleration Module (HAC)
uart1_ips_rdata[31:0]	Input	32-bit read data bus from Universal Asynchronous Receiver/Transmitter (UART1)
uart1_ips_xfr_wait	Input	Universal Asynchronous Receiver/Transmitter (UART1) will not complete transfer in current cycle
uart1_ips_xfr_err	Input	Data Error during transfer from Universal Asynchronous Receiver/Transmitter (UART1)
uart2_ips_rdata[31:0]	Input	32-bit read data bus from Universal Asynchronous Receiver/Transmitter (UART2)
uart2_ips_xfr_wait	Input	Universal Asynchronous Receiver/Transmitter (UART2) will not complete transfer in current cycle
uart2_ips_xfr_err	Input	Data Error during transfer from Universal Asynchronous Receiver/Transmitter (UART2)
mdi_ips_rdata[31:0]	Input	32-bit read data bus from MCU/DSP Interface (MDI)
mdi_ips_xfr_wait	Input	MCU/DSP Interface (MDI) will not complete transfer in current cycle
mdi_ips_xfr_err	Input	Data Error during transfer from MCU/DSP Interface (MDI)
kpp_ips_rdata[31:0]	Input	32-bit read data bus from Keypad Port (KPP)

Table 46-1. IPMUX Module Pin List

Pin Name	Direction	Description
kpp_ips_xfr_wait	Input	Keypad Port (KPP) will not complete transfer in current cycle
kpp_ips_xfr_err	Input	Data Error during transfer from Keypad Port (KPP)
int_ips_rdata[31:0]	Input	32-bit read data bus from MCU External Interrupt Module (INT)
int_ips_xfr_wait	Input	MCU External Interrupt Module (INT) will not complete transfer in current cycle
int_ips_xfr_err	Input	Data Error during transfer from MCU External Interrupt Module (INT)
egpt_ips_rdata[31:0]	Input	32-bit read data bus from Enhanced General Purpose Timer (EGPT)
egpt_ips_xfr_wait	Input	Enhanced General Purpose Timer (EGPT) will not complete transfer in current cycle
egpt_ips_xfr_err	Input	Data Error during transfer from Enhanced General Purpose Timer (EGPT)
iim_ips_rdata[31:0]	Input	32-bit read data bus from IC Identification Module (IIM)
iim_ips_xfr_wait	Input	IC Identification Module (IIM) will not complete transfer in current cycle
iim_ips_xfr_err	Input	Data Error during transfer from IC Identification Module (IIM)
wdog_ips_rdata[31:0]	Input	32-bit read data bus from Watchdog Timer (WDOG)
wdog_ips_xfr_wait	Input	Watchdog Timer (WDOG) will not complete transfer in current cycle
wdog_ips_xfr_err	Input	Data Error during transfer from Watchdog Timer (WDOG)
mqspi_ips_rdata[31:0]	Input	32-bit read data bus from Multiple Queue Seriel Perapheral Interface (MQSPI)
mqspi_ips_xfr_wait	Input	Multiple Queue Seriel Perapheral Interface (MQSPI) will not complete transfer in current cycle
mqspi_ips_xfr_err	Input	Data Error during transfer from Multiple Queue Seriel Perapheral Interface (MQSPI)
ips_rdata[31:0]	Output	IP-Bus 32-bit read data bus
ips_xfr_wait	Output	Module will not complete transfer in current cycle
ips_xfr_err	Output	IP-Bus Data Error during transfer

Table 46-1. IPMUX Module Pin List

Pin Name	Direction	Description
amarb_module_en	Input	IP bus module enable for AMARB
ccm_module_en	Input	IP bus module enable for CCM
tcm_module_en	Input	IP bus module enable for TCM
gpio_module_en	Input	IP bus module enable for GPIO
a2di_module_en	Input	IP bus module enable for A2DIGL
rtc_module_en	Input	IP bus module enable for RTC
dsm_module_en	Input	IP bus module enable for DSM
sim_module_en	Input	IP bus module enable for SIM
gem_module_en	Input	IP bus module enable for GEM
rtr_module_en	Input	IP bus module enable for RTR
l1t_module_en	Input	IP bus module enable for L1T
usb_module_en	Input	IP bus module enable for USB
dmac_module_en	Input	IP bus module enable for DMAC
owire_module_en	Input	IP bus module enable for OWIRE
hac_module_en	Input	IP bus module enable for HAC
uart1_module_en	Input	IP bus module enable for UART1
uart2_module_en	Input	IP bus module enable for UART2
mdi_module_en	Input	IP bus module enable for MDI
mdm0_module_en	Input	IP bus module enable for MDI/MDM0
mdm1_module_en	Input	IP bus module enable for MDI/MDM1
mdm2_module_en	Input	IP bus module enable for MDI/MDM2
mdm3_module_en	Input	IP bus module enable for MDI/MDM3

Table 46-1. IPMUX Module Pin List

Pin Name	Direction	Description
kpp_module_en	Input	IP bus module enable for KPP
int_module_en	Input	IP bus module enable for INT
egpt_module_en	Input	IP bus module enable for EGPT
iim_module_en	Input	IP bus module enable for IIM
wdog_module_en	Input	IP bus module enable for WDOG
mqspi_module_en	Input	IP bus module enable for MQSPI

46.3 IP-Bus Mux Operation

The IPMUX is a mux which inputs the read data bus and data attribute signals from each of the IP-bus modules. The IP-bus module enables serve as the select lines to the mux. The IPMUX outputs a single set of read data bus(ips_rdata[31:0]) and data attribute signals (ips_xfr_wait and ips_xfr_err) to the IP-bus.

If the IP-bus module enable selects an address space which does not match the address space of any of the IP-bus modules, then the IPMUX will assert the data error signal (ips_xfr_err) and drive the wait signal (ips_xfr_wait) low. The following table45-2 shows the mapping of the IP-bus modules on Neptune LTE to the corresponding bit of ips_module_en[31:1].

If an IP-bus module does not generate a wait signal (ips_xfr_wait), then the IPMUX will drive the wait signal (ips_xfr_wait) low for that particular module. RTC doesn't generate the wait signal.

The MDI is connected as a peripheral to AIPI (ARM IP Interface), which buffers the peripherals from the MCU memory bus. A module on AIPI is allocated 4K addresses (byte wide) in the MCU memory map. The "Messaging And Control Unit"(MDR) is mapped within one 4K byte bank in the MCU address space and the "MCU-access Shared Memory Controller" is mapped within 4 other 4K byte banks (MDM0, MDM1, MDM2, MDM3) in the MCU address space. For each of these 4 memory banks, one module enable is needed to select the corresponding memory locations.

And in Neptune LTE, we need two UART (UART1, UART2) from system perspective.

T o implement the design more clearly, each module is enabled by modulename_module_en, and then it gets connected with corresponding ips_module_en[i] on the chip level.

The actual mapping address for the IP bus peripherals are as follows:

a

Table 46-2.

Peripheral Address	ips_module_en	modulename_module_en
0x2484_1000	1	gpio_module_en
0x2484_2000	2	
0x2484_3000	3	rtc_module_en
0x2484_4000	4	tcm_module_en
0x2484_5000	5	ccm_module_en
0x2484_6000	6	a2di_module_en
0x2484_7000	7	amarb_module_en
0x2484_8000	8	egpt_module_en

Table 46-2.

Peripheral Address	ips_module_en	modulename_module_en
0x2484_9000	9	wdog_module_en
0x2484_A000	10	rtr_module_en
0x2484_B000	11	dsm_module_en
0x2484_C000	12	int_module_en
0x2484_D000	13	uart1_module_en
0x2484_E000	14	kpp_module_en
0x2484_F000	15	sim_module_en
0x2485_0000	16	iim_module_en
0x2485_1000	17	mqspi_module_en
0x2485_2000	18	usb_module_en
0x2485_3000	19	l1t_module_en
0x2485_4000	20	dmac_module_en
0x2485_5000	21	
0x2485_6000	22	owire_module_en
0x2485_7000	23	uart2_module_en
0x2485_8000	24	hac_module_en
0x2485_9000	25	gem_module_en
0x2485_A000	26	
0x2485_B000	27	mdi_module_en
0x2485_C000	28	mdm0_module_en
0x2485_D000	29	mdm1_module_en
0x2485_E000	30	mdm2_module_en
0x2485_F000	31	mdm3_module_en

Chapter 47
IC Identification Module (IIM)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	03/19/02	Adil Kidwai	Made the module IP- Bus compliant Updated IIM module pin list Added more description of UID bits Changed description of fuse polarityAdded IIM module PIN diagram. Added security module enable signals
0.2	05/01/02	Adil Kidwai	Modified Rev reg bit values PR[4:0] for Neptune LTS.
0.3	05/23/02	Adil Kidwai	Modified the port list and UID0 description.Modified Security module enable signals.
0.4	24/10/02	Adil Kidwai	Modified “iim_scc_en” bit location Added description for 115-112 bits of UID register.Updated DDTS “DSPhl15606”
0.5	1 November 2002	Mark Babcock	Consolidate Neptune LTS, LTE, and ULS settings for common chapter. (Update REV description table.)
0.6	05/07/03		Updated for LTE specification release.

47.1 Introduction

The IIM consists of two functions: the Unique Identifier (UID) and the Hardware Revision Register(REV).

The Unique Identifier cell is a laser programmed register that is programmed to a unique value for each die produced. The Unique Identifier is a 128 bit wide value that is accessible from the IP bus interface to the MCU.

The Hardware revision register is simply a 16-bit read-only register that is fixed in value. The value is changed in layout whenever a silicon change is performed. This provides the software with a knowledge of what hardware features are available in the current revision. The value in the REV register must be programmable in metal only to allow for metal mask changes to be detected in software. The register is readable only by the MCU.

The upper three bits of the 128 fuse bits are brought out as ports.They are used to individually enable the modules of the security architecture. These bits remain readable by the software so the software knows the configuration.

NOTE:

There are no new bits added to IIM.The only change is to make UID[127:124]accessible through ports iim_hacc_en,iim_msu_en and iim_scc_en.These signals are high when their assosiated fuses are blown.

When reading these bits in UID register,0 means that the fuse is intact and the associated security module is disabled .1 means the fuse is blown and the associated security module is enabled.

47.1.1 IIM Module Pin List

Table 47-1 is a list of the IIM module pins.

Table 47-1. IIM Module Pin List

S.No	Pin Name	Direction	Description
1.	hrev_reg[15:0]	input	Hardware revision register
2.	ips_addr[11:1]	input	Module register address
3.	ips_module_en	input	Module select
4.	ips_rwb	input	Module read/write control,read = 1 write = 0
5.	ccm_reset_b	input	Reset signal
6.	ips_xfr_wait	output	Wait signal
7.	ips_xfr_err	output	Invalid access error signal
8.	iim_hacc_en	output	Hash accelerator block enable
9.	iim_scc_en	output	Secure RAM and Security Monitor Module enable
10.	iim_msu_en	output	Memory Separation Unit enable
11.	ips_rdata[15:0]	output	Peripherals data-bus from module

47.2 IIM Module Block Diagram

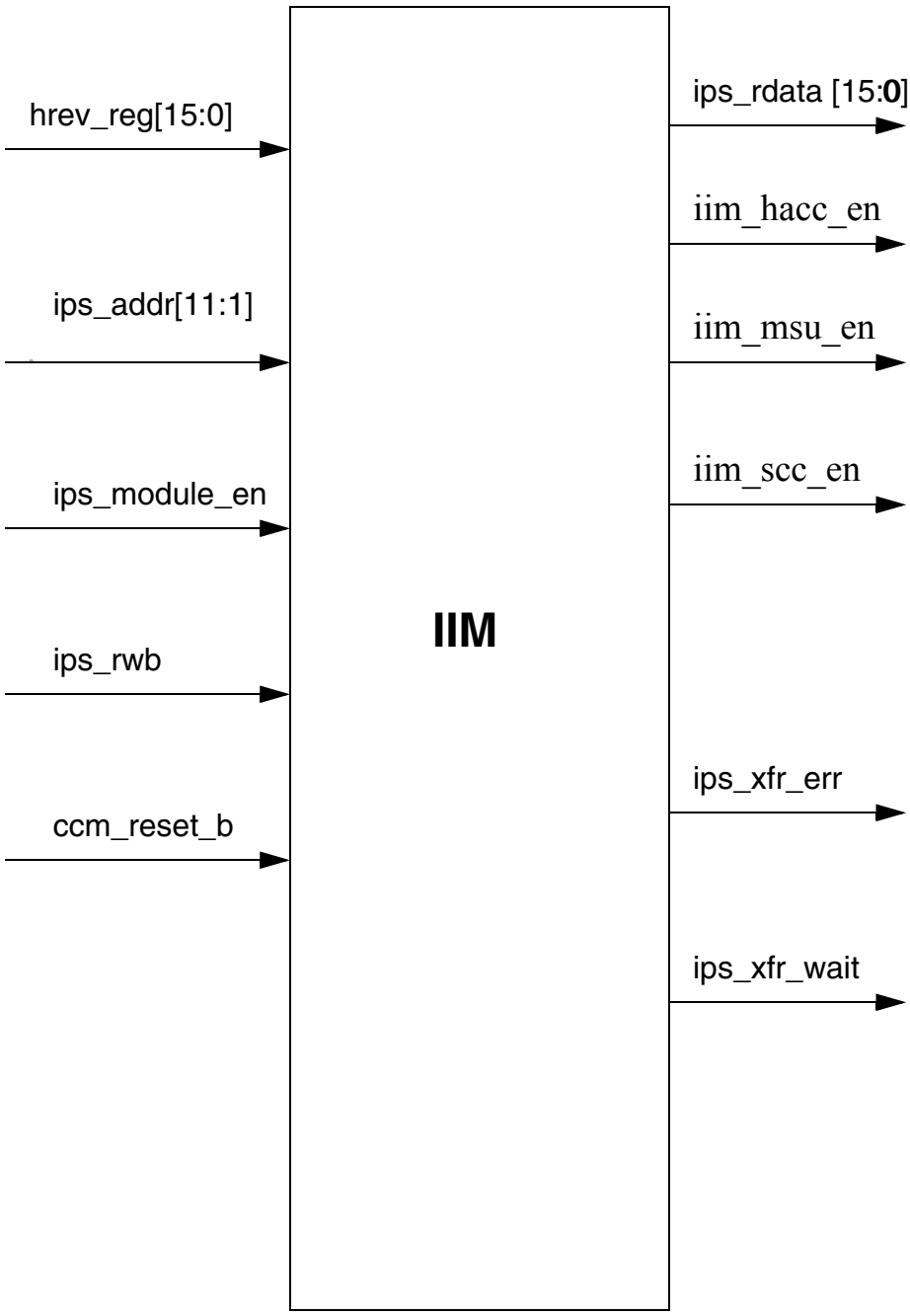


Figure 47-1. IIM Block Diagram

47.3 Register Summary

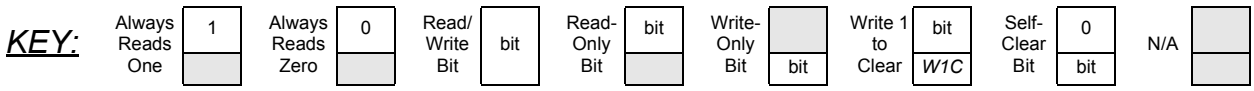


Table 47-2. UID Register Map Summary (Base Addr = $2485_0000)

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID0 ($2485_0000)	R	UID[127:112]															
	W																
UID1 ($2485_0002)	R	UID[111:96]															
	W																
UID2 ($2485_0004)	R	UID[95:80]															
	W																
UID3 ($2485_0006)	R	UID[79:64]															
	W																
UID4 ($2485_0008)	R	UID[63:48]															
	W																
UID5 ($2485_000A)	R	UID[47:32]															
	W																
UID6 ($2485_000C)	R	UID[31:16]															
	W																
UID7 ($2485_000E)	R	UID[15:0]															
	W																
REV ($2485_0010)	R	REV[15:0]															
	W																

47.3.1 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various UID registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

UID0 Unique Identifier Register 0 $2485_0000

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	UID[127-112]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Table 47-3. UID0 Description

Name	Description	Settings
UID [127-112] Bits 15-0	**Unique Identifier--** The Unique Identifier Register (UID) contains a 128-bit read-only field. Each part is programmed with a unique security identification code during the manufacturing process. Programming is done with laser fuses. Bits 127-124 are used as security module enables.These signals are readable so that the software can determine the chip configuration. **Note:** These bits are read 1 when the associated module is enabled (ie the associated fuse is blown). Bits 115-112 are used to indicate the usage of the chip to the software.	UID[127]-iim_hacc_en(1 = enabled,fuse blown) UID[126]-iim_msu_en(1 = enabled,fuse blown) UID[125]-iim_srom_en(1 = enabled, fuse blown) -- Reserved. Not used in Neptune LTE. UID[124]-iim_scc_en(1 = enabled,fuse blown)

UID[115:112]	Configuration
0001	Development Part
0010	Production Part
1000	Non-Secure Part

UID1 Unique Identifier Register 1 $2485_0002

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	UID[111-96]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Table 47-4. UID1 Description

Name	Description	Settings
UID [111-96] Bits 15-0	**Unique Identifier--** The Unique Identifier Register (UID) contains a 128-bit read-only field. Each part is programmed with a unique security identification code during the manufacturing process. Programming is done with laser fuses. **Note:These bits are read 1 when the associated fuse is blown and vice versa.**	N/A

UID2 Unique Identifier Register 2 $2485_0004

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	UID[95-80]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Table 47-5. UID2 Description

Name	Description	Settings
UID [95-80] Bits 15-0	**Unique Identifier--** The Unique Identifier Register (UID) contains a 128-bit read-only field. Each part is programmed with a unique security identification code during the manufacturing process. Programming is done with laser fuses. **Note:These bits are read 1 when the associated fuse is blown and vice versa.**	N/A

UID3 Unique Identifier Register 3 $2485_0006

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	UID[79-64]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Table 47-6. UID3 Description

Name	Description	Settings
UID [79-64] Bits 15-0	**Unique Identifier--** The Unique Identifier Register (UID) contains a 128-bit read-only field. Each part is programmed with a unique security identification code during the manufacturing process. Programming is done with laser fuses. **Note:These bits are read 1 when the associated fuse is blown and vice versa.**	N/A

UID4 Unique Identifier Register 4 $2485_0008

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	UID[63-48]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Table 47-7. UID4 Description

Name	Description	Settings
UID [63-48] Bits 15-0	**Unique Identifier--** The Unique Identifier Register (UID) contains a 128-bit read-only field. Each part is programmed with a unique security identification code during the manufacturing process. Programming is done with laser fuses. **Note:These bits are read 1 when the associated fuse is blown and vice versa.**	N/A

UID5 Unique Identifier Register 5 $2485_000A

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	UID[47-32]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Table 47-8. UID5 Description

Name	Description	Settings
UID [47-32] Bits 15-0	**Unique Identifier--** The Unique Identifier Register (UID) contains a 128-bit read-only field. Each part is programmed with a unique security identification code during the manufacturing process. Programming is done with laser fuses. **Note:These bits are read 1 when the associated fuse is blown and vice versa.**	N/A

UID6 Unique Identifier Register 6 **$2485_000C**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	UID[31-16]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Table 47-9. UID6 Description

Name	Description	Settings
UID [31-16] Bits 15-0	**Unique Identifier--** The Unique Identifier Register (UID) contains a 128-bit read-only field. Each part is programmed with a unique security identification code during the manufacturing process. Programming is done with laser fuses. **Note:These bits are read 1 when the associated fuse is blown and vice versa.**	N/A

UID7 Unique Identifier Register 7 $2485_000E

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	UID[15-0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Table 47-10. UID7 Description

Name	Description	Settings
UID [15-0] Bits 15-0	**Unique Identifier--** The Unique Identifier Register (UID) contains a 128-bit read-only field. Each part is programmed with a unique security identification code during the manufacturing process. Programming is done with laser fuses. **Note:These bits are read 1 when the associated fuse is blown and vice versa.**	N/A

REV Hardware Revision Register $2485_0010

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	X----------PR[4] ----------X--------- PVT[2:0]-------X---------- SR[7:0]----------X															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Table 47-11. Hardware Rev Register Description

Name	Description	Settings
PR[4:0] Bits 15-11	**Product Revision** - Specifies the Neptune product type (ROM, RAM, Production RAM, etc.).	**PR[4:0]** / **Neptune Product** 00000 - Reserved 10001 - Neptune LTS ROM 10010 - Neptune LTE ROM 10011 - Neptune ULS ROM
PVT[2:0] Bits 10-8	**Product Vendor / Technology -** Specifies the Neptune vendor and silicon technology (SPS HIP6W, SPS HIP7, SPS HIP8 etc.).	**PVT[2:0]** / **Neptune Vendor / Technology** 000 001 010 - SPS Hip7 011 - SPS Hip8
SR[7:0] Bits 7-0	**SIlicon Revision -** Increments with each silicon change.	0000_0000 - Initial Revision (Pass 1) 0000_0001 - Pass 2

Chapter 48
ARM External Interface Module (AEIM)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	08/19/02	Shannon Osgood	Updated per DDTS# DSPhl15160.
0.2	09/04/02	Mark Babcock, Shannon Osgood	Updated per DDTS# DSPhl15216. Updated note under Table 48-3.
0.3	12/05/02	Mark Babcock, Shannon Osgood	Updated per DDTS# DSPhl15813.
0.4	05/07/03		Updated for LTE specification release.

48.1 Overview

The ARM External Interface Module (AEIM) handles the interface to devices external to an ARM architecture based chip, including generation of chip selects for external peripherals and memory. It contains the following features:

- Six Chip Selects for external devices, each covering an unrestricted address range
- Programmable Wait-State generator for each Chip Select
- Selectable Protection for each Chip Select
- Programmable Data Port Size for each Chip Select
- Bus Watchdog counter for all bus cycles
- Programmable general output capability for unused Chip Select outputs
- Show cycles to allow internal bus cycles to be externally monitored
- Synchronous Burst Mode support for industry standard flash devices
- 20 dedicated General Purpose Outputs
- Address suppression during burst mode operations

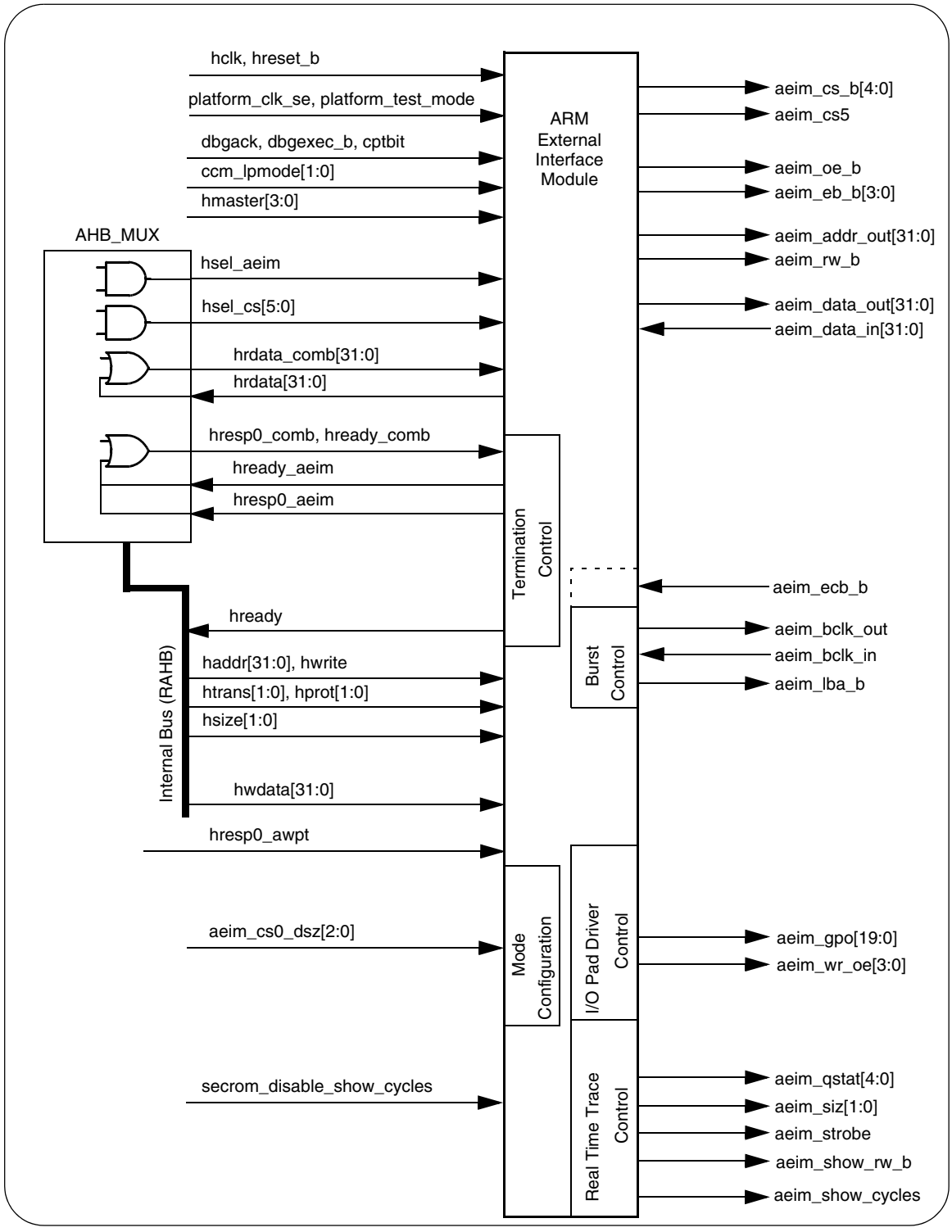


Figure 48-1. AEIM Block Diagram

48.2 AEIM Module Pin List

Table 48-1 is a list of the AEIM module pins.

Table 48-1. AEIM Module Pin List

Pin Name	Direction	Description
RAHB Interface Signals		
hrdata[31:0]	output	Read Data bus (reads from AEIM registers and external memory)
hrdata_comb[31:0]	input	AHB_MUX combined Read Data from all slaves (used for show cycles)
hwdata[31:0]	input	Write Data bus (writes to AEIM registers and external memory)
hready	output	Indicates the bus transaction's data phase is complete (for entire system)
hready_aeim	output	Indicate the AEIM's bus transaction data phase is complete
hready_comb	input	AHB_MUX combined hreadys from all slaves (used to delay hready for system during show cycles if external bus is busy).
hresp0_aeim	output	Asserted to acknowledge a bus transfer error
hresp0_comb	input	AHB_MUX combined hresp0s from all slaves (final RAHB hresp0 signal)
hresp0_awpt	input	HRESP output from Watchpoint Module
haddr[31:0]	input	RAHB Address
hprot[1:0]	input	Indicates bus transfer priviledge level
hsize[1:0]	input	Indicates data transfer size
htrans[1:0]	input	Indicates RAHB transfer type
hsel_aeim	input	Indicates AEIM register access decoded
hsel_cs[5:0]	input	Indicates AEIM chip select access decoded
hwrite	input	Identifies the transfer as a read or write access
hmaster[3:0]	input	AHB master number - used on show cycles to generate aeim_qstat[4:0]
System Signals		
hclk	input	System Clock; Same clock domain as RAHB
hreset_b	input	RAHB Reset; asynchronous, active low
aeim_cs0_dsz[2:0]	input	Input signals that determine the reset (external boot) size of the memory bus for CS0
platform_clk_se	input	ARM7 Platform clock gating cell scan enable
platform_test_mode	input	ARM7 Platform scan test mode control
secrom_disable_ show_cycles	input	From Secure ROM module, to disable show cycles.
External I/O Signals		
aeim_data_out[31:0]	output	External Data bus outputs to pad drivers
aeim_addr_out[31:0]	output	External Address bus outputs to pad drivers
aeim_gpo[19:0]	output	General purpose outputs
aeim_cs_b[4:0]	output	Active Low External Chip Selects
aeim_cs5	output	Active High External Chip Select
aeim_eb_b[3:0]	output	Active low external Enable Byte signals. aeim_eb_b[3] controls D[31:24], aeim_eb_b[0] controls D[7:0].
aeim_wr_oe[3:0]	output	External Data tri-state enables for each byte of the external data bus. aeim_wr_oe[3] controls D[31:24], aeim_wr_oe[0] controls D[7:0].

Table 48-1. AEIM Module Pin List (Continued)

Pin Name	Direction	Description
aeim_oe_b	output	Active Low output enable for external data bus
aeim_bclk_out	output	Clock used for external synchronous memories
aeim_lba_b	output	Active low Load Burst Address indicates to external synchronous memories that Address is valid for latching
aeim_rw_b	output	Indicates if external access is a read (high) or write (low) cycle
aeim_data_in[31:0]	input	External Data bus inputs (driven by the pad input buffers)
aeim_ecb_b	input	Input signal that identifies when to end an external access. It is active low in burst mode, while it's polarity is determined by CSPA bit for asynchronous access when in external termination mode (IWAIT[5:0]=111111).
aeim_bclk_in	input	Clock used for external synchronous memories. This is the aeim_bclk_out signal routed through the I/O pad and fed back into the AEIM to reduce setup time requirements for external memories.
Real Time Trace and Debug Signals		
aeim_strobe	output	This signal can be used as a Logic Analyzer clock to capture the AEIM external I/O when valid. The rising edge of the aeim_strobe signal represents valid external I/O.
aeim_show_rw_b	output	This signal indicates whether an access is a read or a write.
aeim_qstat[4:0]	output	These signals represent the state of the ARM Processor and RAHB bus.
aeim_siz[1:0]	output	These signals are a latched version of the ARM p_tsiz[1:0] signals.
aeim_show_cycles	output	AEIM Show Cycles Indicator: 1=> Enabled, 0=>Disabled (to Secure RAM module).
ccm_lpmode[1:0]	input	Indicates low power mode of operation
dbgack	input	MCU debug mode status for qstat generation during show cycles
dbgexec_b	input	ARM instruction execution indicator
cptbit	input	ARM / Thumb state indicator from ARM7 core

48.3 Typical AEIM System Connections

The following figures show example connections of the AEIM with burst and asynchronous memories. Figure 48-2 shows an example of a typical set of AEIM interfaces to external memory and peripherals. Figure 48-3 shows the AEIM interface to two supported external burst flash devices.

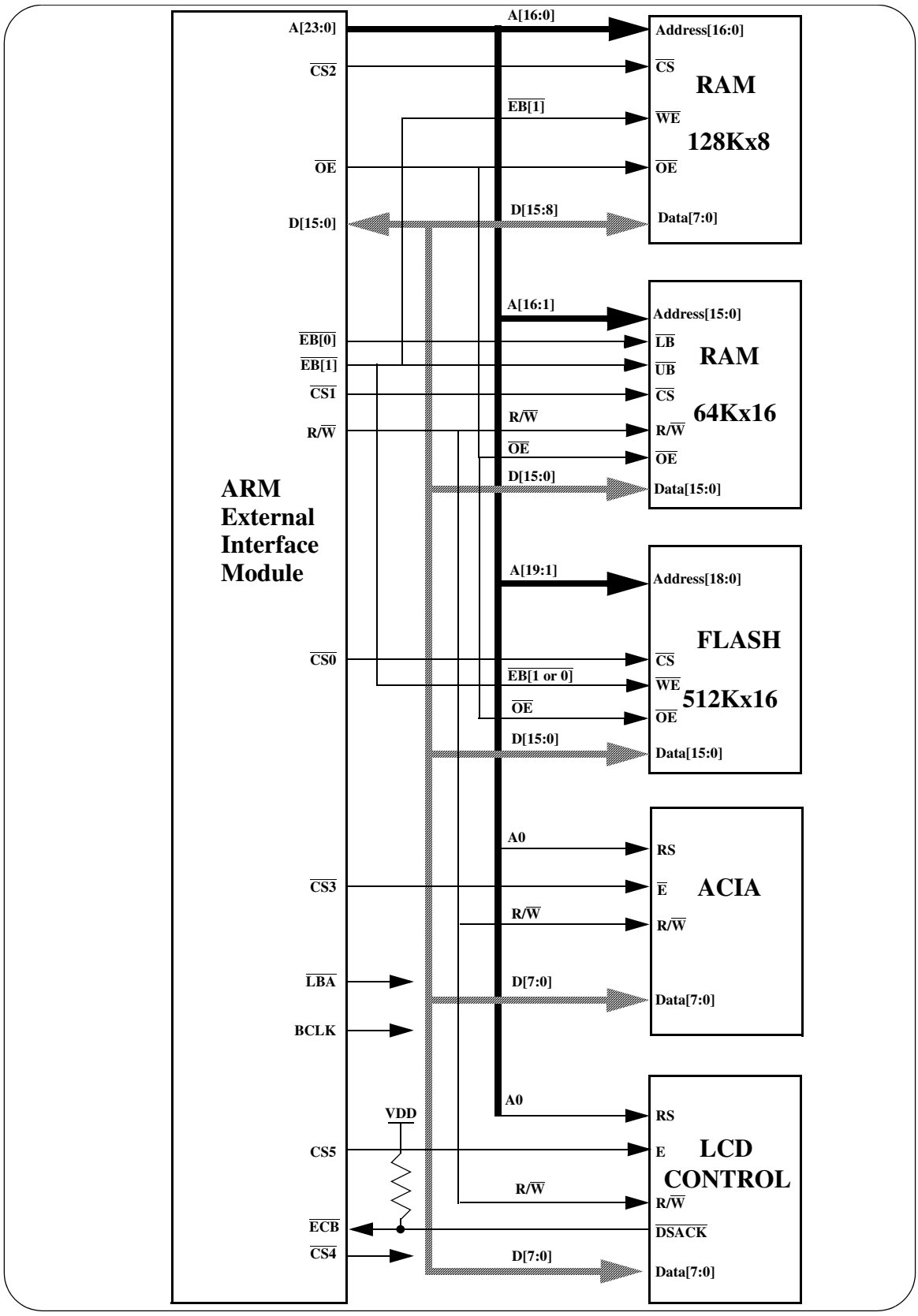


Figure 48-2. Example of AEIM Interface to Memory and Peripherals

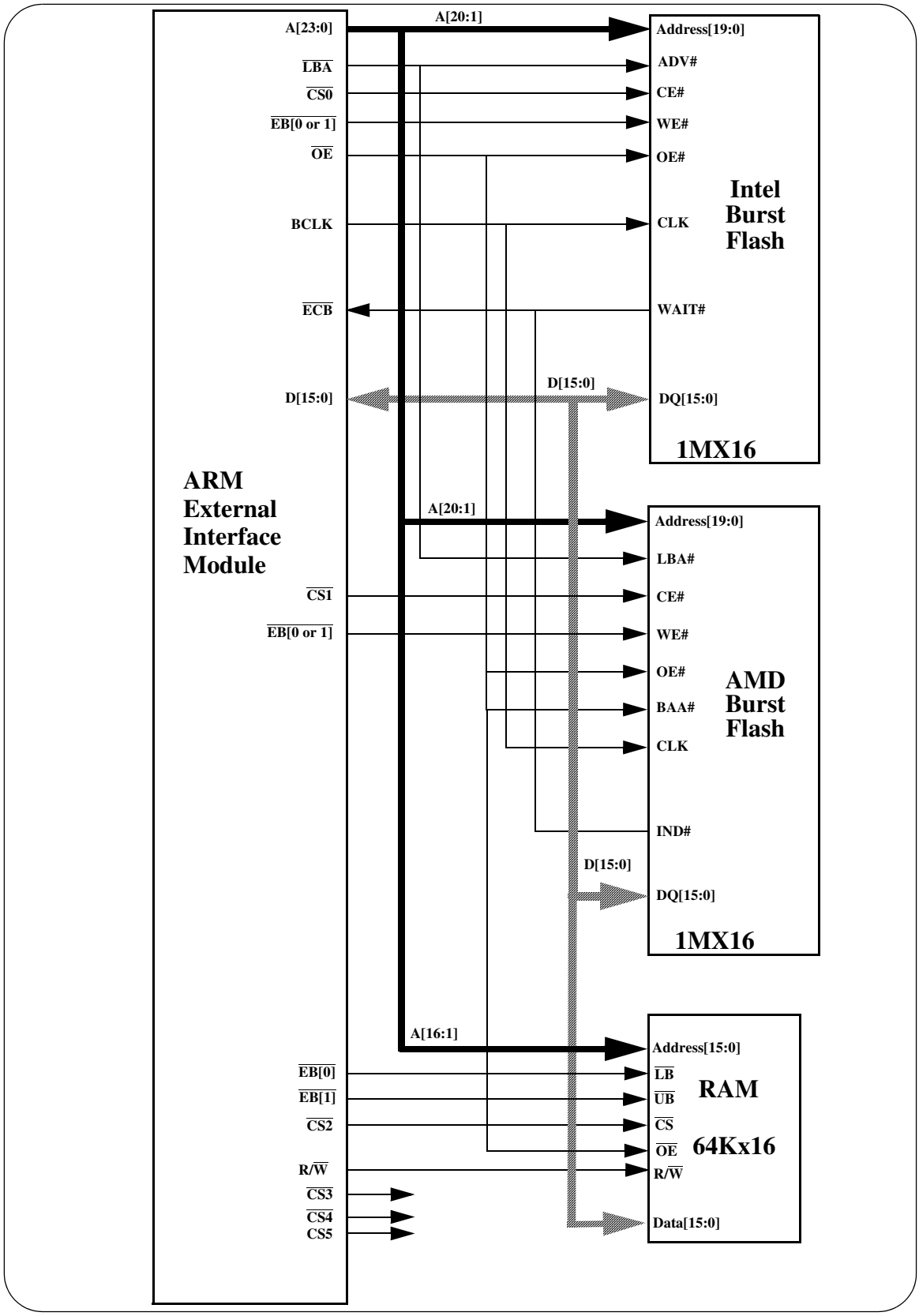


Figure 48-3. Example for AEIM Interface to Burst Memory

48.4 AEIM Functionality

The following sections describe the functionality of the AEIM module.

48.4.1 Chip Select Outputs

The AEIM provides six outputs intended to be used as Chip selects for external devices. The chip select functionality is described in the following sub sections. Chip select 0 (CS0) has special capabilities and is discussed separately.

48.4.1.1 $\overline{\text{CS0}}$ - Chip Select 0

This active-low output signal is asserted based on the hsel_cs[0] input port. The address range assigned to CS0 is controlled by the AHB_MUX on the platform. Refer to Section 48.4.5 on page 48-10 for a description of the boot behavior of the AEIM and CS0.

CS0 is controlled through the CS0_PRIM_CFG, CS0_SEC_CFG, CS0_WS_CTL, and CS0_BCLK_CTL registers. These registers provide software the following control of the chip select outputs:

- Ability to enable/disable the chip select (CSEN control bit)
- Ability to control the relationship of CS0 to the beginning and end of the external transfer (CSWID[3:0] control bits).
- Ability to control the time between back-to-back accesses (EDEAD[1:0] control bits)
- Ability to configure Chip Selects for synchronous operation (SYNC control bit)
- Ability to restrict the minimum number of cycles CS0 can be de-asserted (CSIDLE[1:0] control bits)

Refer to Table 48-10 on page 48-27 and Table 48-11 on page 48-29 for a detailed description of the controllability of the Chip select outputs via the register set.

CS0 has one special capability not given in any other chip select - it can be used to boot from external (off-chip) memory. More information on this subject is available (See **External Boot ROM Control** on page -10.)

Since CS0 is used for external boot, it differs from the other chip selects in the following ways:

- It is enabled at reset.
- It cannot be controlled as a general purpose output while disabled.When disabled it will remain at logic ‘1’.
- Reset state of the size and alignment of the external memory (DPSIZ[2:0] control bits) is controlled by the aeim_cs0_dsz[2:0] input ports. After reset, the chip select data size (DPSIZ[2:0]) can be configured by software.

48.4.1.2 $\overline{\text{CS1}}$ - $\overline{\text{CS4}}$, CS5- Chip Select 1 - Chip Select 5

These output signals are asserted based on the hsel_cs[5:1] input ports. CS1 through CS4 are active low. CS5 is active high. The address ranges assigned to these chip selects are controlled by the AHB_MUX on the platform. A reset event will force the chip select’s respective control registers to their default state. This renders the chip selects disabled, with the outputs negated.

CS1-CS5 are controlled through the CSx_PRIM_CFG, CSx_SEC_CFG, CSx_WS_CTL, and CSx_BCLK_CTL registers. Each chip select has its own set of registers. These registers provide software the following control of the chip select outputs:

- Ability to enable/disable the chip select (CSEN control bit)
- Ability to control the relationship of Chip Selects to the beginning and end of the external transfer (CSWID[3:0] control bits).
- Ability to control the time between back-to-back accesses (EDEAD[1:0] control bits)
- Ability to configure Chip Selects for synchronous operation (SYNC control bit)
- Software control of the Chip Select data size (DPSIZ[2:0] control bits)
- Ability to restrict the minimum number of cycles a chip select can be de-asserted (CSIDLE[1:0] control bits)

48.4.1.3 Programmable Output Generation

Unused Chip Select outputs may be configured to provide a programmable output signal. This functionality is not provided for the $\overline{CS0}$ output signal. When CSEN0 bit is cleared, $\overline{CS0}$ is always inactive (logic '1'). To operate as a programmable output pin, the corresponding CSENi control bit must be cleared. Since CS5 is an active-high chip-select (by inversion of a standard chip select), the polarity of the CS5 output is the inversion of the CSPA bit's state.

48.4.2 Programmable Transfer Duration

The AEIM provides the ability to program the duration (number of clock cycles) for an external transfer independently for each chip select address range. Using the control registers, software programs the number of "wait states" used for a transfer to the selected chip select. A "wait state" is simply an added delay of one cycle to an external transfer. If a transfer has zero wait states, then it takes one cycle to complete. A two wait-state transfer takes three cycles to complete, and so on...

The programmability of the wait states is provided by the CS Wait State Control (CSx_WS_CTL) registers. This register also provides additional programmability:

- Ability to add up to 62 wait states independently for each chip select (IWAIT[5:0] control bits)
- Ability to add additional wait states to write accesses independently for each chip select (WWAIT[1:0] control bits)
- Variable wait states (0 to 512) independently available for each chip select using the aeim_ecb_b input. (CSPA, IWAIT[5:0] control bits)
- Ability to add wait states for synchronous access (initial and burst) independently for each chip select (IWAIT[5:0], BWAIT[3:0] control bits)

Refer to Table 48-12 on page 48-34 for a detailed description of the controllability of the Chip select wait states using the CSx_WS_CTL register.

48.4.3 Protected Addressing

The AEIM provides the ability to protect each chip select range from various types of accesses. Each chip select can be protected from user-mode accesses, write accesses and sub-atomic writes. When the chip select is set for 'Supervisor Only' access, the AEIM generates hresp=ERROR if a user-mode access is attempted. Likewise, if the chip select is configured for 'read only' access, the AEIM generates hresp=ERROR to the RAHB if a write is attempted. Finally, if the chip select is set for 'Atomic Write Only' access, the AEIM generates hresp=ERROR if a write to less than the full port width of the selected device is attempted.

These protected modes are not mutually exclusive. In other words, it is possible to configure a chip select as Supervisor-only and Read-only accessible.

48.4.4 Configurable Bus Sizing

The AEIM supports byte, halfword, and word operands allowing access to 8-bit ports and 16-bit ports. Misaligned transfers are forced into alignment. This is accomplished by zeroing out the appropriate lower address bits. In other words, all half word accesses have A0 = 0 and all word accesses have A1=0 and A0 = 0.

The port size is programmable through the AEIM register set. In addition, an 8-bit port can reside on either D[15:8] or D[7:0] of the data bus. The reset state of the CS0 port width and alignment is controlled by the aeim_cs0_dsz[2:0] pins. After reset, all chip selects' size and port alignments are configurable by software.

A word access to an 8-bit port requires four external bus transfers to complete the access. A word access to a 16-bit port requires two external bus transfers to complete the access. A halfword access to an 8-bit port requires two external bus transfers to complete the access. In the case of a multi-transfer access, the lower two address bits, A[1:0], are incremented appropriately.

Table 48-2. "Interface Requirements for Read and Write Cycles", lists the combination of hsize[1:0], haddr[1:0] signals and DPSZ[2:0] bits that are used for each possible transfer size, alignment, and port width. The bytes labeled with a dash are not required; they are ignored on read transfers, and driven with undefined data on write transfers.

Table 48-2. Interface Requirements for Read and Write Cycles

Transfer	Signal Encoding from the RAHB				Port Width	External Address		Active Interface Bus Sections (RAHB to External)			
Size	hsize[1]	hsize[0]	haddr[1]	haddr[0]	DPSIZ[2:0]	A1	A0	RAHB D31-D24	RAHB D23-D16	RAHB D15-D8	RAHB D7-D0
Byte	0	0	0	0	000	0	0	D[15:8]	---	---	---
					001	0	0	D[7:0]	---	---	---
					010	0	0	D[15:8]	---	---	---
			0	1	000	0	1	---	D[15:8]	---	---
					001	0	1	---	D[7:0]	---	---
					010	0	1	---	D[7:0]	---	---
			1	0	000	1	0	---	---	D[15:8]	---
					001	1	0	---	---	D[7:0]	---
					010	1	0	---	---	D[15:8]	---
			1	1	000	1	1	---	---	---	D[15:8]
					001	1	1	---	---	---	D[7:0]
					010	1	1	---	---	---	D[7:0]

Table 48-2. Interface Requirements for Read and Write Cycles (Continued)

Transfer	Signal Encoding from the RAHB				Port Width	External Address		Active Interface Bus Sections (RAHB to External)			
Size	hsize[1]	hsize[0]	haddr[1]	haddr[0]	DPSIZ[2:0]	A1	A0	RAHB D31-D24	RAHB D23-D16	RAHB D15-D8	RAHB D7-D0
Half word	0	1	0	X	000	0	0	D[15:8]	---	---	---
							1	---	D[15:8]	---	---
					001	0	0	D[7:0]	---	---	---
							1	---	D[7:0]	---	---
					010	0	0	D[15:8]	D[7:0]	---	---
			1	X	000	1	0	---	---	D[15:8]	---
							1	---	---	---	D[15:8]
					001	1	0	---	---	D[7:0]	---
							1	---	---	---	D[7:0]
					010	1	0	---	---	D[15:8]	D[7:0]
Word	1	0	X	X	000	0	0	D[15:8]	---	---	---
							1	---	D[15:8]	---	---
						1	0	---	---	D[15:8]	---
							1	---	---	---	D[15:8]
					001	0	0	D[7:0]	---	---	---
							1	---	D[7:0]	---	---
						1	0	---	---	D[7:0]	---
							1	---	---	---	D[7:0]
					010	0	0	D[15:8]	D[7:0]	---	---
						1	0	---	---	D[15:8]	D[7:0]

a. hsize[1:0] = 11 is invalid for 32-bit processor implementations (i.e. ARM7, ARM9)

b. DPSIZ[2:0] = 1xx are reserved configurations

48.4.5 External Boot ROM Control

The platform's boot_int and aeim_cs0_dsz[2:0] input signals are used to determine the location and external bus alignment of the boot ROM device during hardware reset. If an external boot ROM is used instead of the internal ROM, the $\overline{CS0}$ output is used to select the external ROM.

This is accomplished by swapping the memory map decodings for the internal ROM with the CS0 chip select. This functionality is handled by the address decoder within the AHB_MUX. For more information on the memory map refer to the AHB_MUX and memory map specifications.

Table 48-3. CS0_DPSIZ[2:0] Encoding

CS0_DPSIZ[2:0]	$\overline{CS0}$ Reset Port Size, Location
000	8-bit Port, resides on D[15:8] pins
001	8-bit Port, resides on D[7:0] pins
010	16-bit Port, resides on D[15:0] pins
011	32-bit Port
1XX	Reserved

Neptune LTE does not support EBW selection using the EBW pin. Reset boot configuration of 16-bit external memory is implemented. CS0 bus width configuration can still be changed in the CS0_PRIM_CFG by software.

48.4.6 Enable Byte Outputs

The AEIM provides four signals that allow external control of byte lanes - the enable byte (aeim_eb_b[3:0]) signals. The aeim_eb_b[3:0] signals may be configured to act as byte lane write strobes or true byte lane enables.

48.4.6.1 Enable Byte Control

The AEIM provides several methods of controlling the timing relationship between the enable byte signals and the other AEIM I/O (Address, Data, CS, etc....). The register set provides for the configuration of the enable bytes.

The register set provides several means to adjust the timing of the enable bytes independently for each of the six chip selects through the CS Primary and Secondary Config registers. The following adjustments to enable byte timing are provided:

- Assert enable bytes during write only, or on both read and write accesses (EBM control bit)
- Programmable assertion delay from beginning of cycle (EBASS[2:0] control bits)
- Programmable negation delay from end of cycle (EBNEG[2:0] control bits)

Refer to Table 48-10 on page 48-27 and Table 48-11 on page 48-29 for detailed description of the controllability of the enable byte signals provided by the CS Primary and Secondary Config registers.

48.4.7 Output Enable Outputs

The AEIM provides the aeim_oe_b signal to allow external control of when a device is allowed to drive the external data bus. The register set provides the ability to delay the output enable by a programmable number of half-clock cycles from the start of a read cycle. Refer to Table 48-11 on page 48-29 for detailed description of the controllability of the output enable signal provided by the register set.

48.4.8 Burst (Synchronous) Mode

The AEIM provides support for accessing external synchronous devices. Several control bits are provided to enable enhanced performance modes for synchronous operation, and to enable modes specifically designed to support a particular vendor.

The register set provides the following for synchronous operation:

- Ability to enable synchronous operation independently for each chip select (BEN control bit)
- Ability to program the number of wait-states for initial and burst accesses (IWAIT[5:0], BWAIT[3:0] control bits)
- Support for AMD Family of Burst Flash memories (BCLK_CTL register)
- Ability to independently control the BCLK waveform during initial and burst phases of accesses for each chip select (CSx_BCLK_CTL register)
- Ability to control when read data is captured inside the AEIM (BFEED control bit)
- Ability to control the number of transitions that occur on the external BCLK signal (IPULSE[2:0], IDIV[1:0], and BDIV[1:0] control bits)
- Ability to control when the BCLK begins to transition during the access for initial and burst accesses (IBDLY[2:0], BBDLY[2:0] and BBDIR control bits)

48.4.8.1 Burst (Synchronous) Mode Operation

With burst mode enabled by setting the BEN control bit, the AEIM will burst read data from as many sequential address locations as possible, being limited only by the length of the burst flash internal page buffer, or the non-sequential nature of the RAHB master's code or data stream.

For the first access in a burst sequence, the AEIM asserts Load Burst Address ($\overline{\text{LBA}}$) along with the Address and Chip Select. The AEIM then generates BCLK according to the configuration of the Chip Select Configuration registers. The first rising edge of the burst clock (BCLK) signals the burst device to capture the Address value on the external Address bus and use it as the initial address of a block of internal data memory. The AEIM then negates $\overline{\text{LBA}}$ and proceeds to toggle BCLK a predefined number of cycles to latch the first unit of data into the AEIM from the burst device. Subsequent read data units can then be burst from the external device in fewer clock cycles, realizing an overall increase in bus bandwidth.

Burst accesses are terminated by the AEIM whenever it detects that the next RAHB access is not sequential in nature or by the external burst device if it needs additional cycles to retrieve the next requested memory location ($\overline{\text{ECB}}$ asserted.)

Burst accesses do not terminate during internal idle cycles on the RAHB. This is a change in functionality from the original Patriot CDR3 EIM design that provides better performance. The AEIM delays the negation of the chip select output and the assertion of the next BCLK edge until the next active cycle. If the core platform enters a low-power state, the active Chip Select is negated to terminate a pending burst.

Writes to a chip select configured for burst mode are forced to be initial accesses, even if they occur in succession or are wider than the external memory data bus. This is because no burst memories presently defined support burst writes.

48.4.8.2 Pulse Mode Operation

Pulse mode operation supports AMD burst flash Am29BL162C without requiring a dedicated pin for $\overline{\text{BAA}}$. This mode of operation allows the user to connect the $\overline{\text{OE}}$ signal from the AEIM to both the OE# and BAA# inputs on the burst mode flash device.

Pulse mode is utilized by configuring the CSx_BCLK_CTL register in a specific fashion described below. When pulse mode is used, BCLK is pulsed initially (during $\overline{\text{LBA}}$ assertion) so that the address can be latched with the $\overline{\text{LBA}}$ signal and then once more immediately following $\overline{\text{LBA}}$ negation. It is then held low during the completion of the initial access time and restarted only when OE is asserted. For every subsequent burst access, BCLK is pulsed while OE is asserted. For operation when the burst access requires wait states,

(BWAIT[4:0] is non-zero), BCLK must only pulse once during the burst access regardless of how many cycles the burst access actually takes to complete. This allows the OE signal to remain low through the entire burst access and still function as BAA. This is achieved by programming the BDIV[1:0] field to divide the BCLK during the burst phase of the access. This will allow successful pulse mode operation with up to 3 wait states for the burst access phase (BCLK can be divided by 4 at most).

Pulse mode operation should not be used for non-AMD burst flash because the other flash vendor's interfaces count BCLK cycles during the initial access in order to output data at the appropriate time. When using Pulse mode with AMD flash Am29BL162C be sure and program IWAIT >= 1 in order for the flash to function properly. This is necessary, independent of system frequency.

Pulse mode can be achieved by configuring the CSx_BCLK_CTL register to the following values:

1. IBDLY[2:0] = 010
2. BBDIR = 1
3. BBDLY[2:0] = 000
4. IDIV[1:0] = 0
5. BDIV[1:0] = 0
6. IPULSE[2:0] = 010 (AMD only)

These settings are for configuring the BCLK operation only! The wait-states, chip select, and output enable controls would have to be set up according to the requirements of the memory being accessed. The AMD BAA input should still be connected to the AEIM OE output.

NOTE:

The BCM bit in the AEIM Configuration Register has priority over the other control register bit settings. If BCM = 1, the BCLK will continue to run regardless of the access being performed. This will cause access to any burst memory device to fail!

48.4.8.3 Quasi-Pipelined Mode

With high system frequencies normal burst mode operation may not allow single clock sequential accesses. In order to allow some flexibility in trying to achieve single clock sequential accesses, the user has the option to enter into Quasi-Pipelined Mode. In this mode, the burst clock and $\overline{\text{LBA}}$ assertions are skewed in half clock cycle increments so they are still valid with the rising edge of BCLK. Instead of data needing to be valid at the rising edge of BCLK, more relaxed timing allows data to be valid on the following falling edge of BCLK or later. The read data will be captured by the AEIM on the rising edge of BCLK_IN. This will compensate for routing and buffer delays that would ordinarily require additional burst wait states.

The only disadvantage of this mode is that it will cost the user additional wait states for the initial access to the flash memory.

The BCLK configuration registers allow for detailed control of the BCLK waveform. Quasi-Pipelined Mode operation can be achieved by configuring the CSx_BCLK_CTL register to the following values:

1. IBDLY[2:0] >= 001
2. BBDIR = 1
3. BBDLY[2:0] >= 001
4. IDIV[1:0] = 0
5. BDIV[1:0] = 0
6. BFEED = 1
7. IPULSE[2:0] = 010 (AMD), or IPULSE[2:0] = 111 (Intel Siretta)

These settings are for configuring the BCLK operation only! The wait-states, chip select, and output enable controls would have to be set up according to the requirements of the memory being accessed.

NOTE:

The BCM bit in the AEIM Configuration Register has priority over the other control register bit settings. If BCM = 1, the BCLK continues to run in its natural, non-delayed, state and Quasi-Pipelined Mode will not function properly.

48.4.9 Bus Watchdog Operation

The AEIM contains a bus watchdog timer which monitors the length of all requested accesses on the RAHB. If an access does not properly terminate (i.e. the bus monitor timer does not receive an hready within 513 clocks of the access being initiated), the bus monitor timer will time out, and will force the access to be terminated by negating the Chip Select output and any control signals which were asserting during the access, and will drive the hresp=ERROR signal back to the bus master. The bus monitor timer restarts after the termination of each access. If an internal MCU peripheral does not terminate its access, or if the bus master accesses an unmapped memory location, the bus monitor times-out to prevent the bus from locking up.

48.4.10 Error Conditions

The following conditions will cause hresp=ERROR to be asserted by the Bus Watchdog:

- Access to a disabled chip-select, i.e. an access to a mapped chip-select address space where the CSEN bit in the corresponding CS configuration register is clear.
- Write access to a read-only chip-select address space, i.e. the RO bit in the corresponding CS configuration register is set.
- Attempting to write data with size that is less than the port width when the corresponding Atomic Write Only (AWO) bit is set.
- User access to a supervisor-protected chip-select address space, i.e. the SO bit in the corresponding CS configuration register is set.
- Bus watchdog time-out when an access does not terminate within 513 clocks of being initiated. See Section 48.4.9, “Bus Watchdog Operation,” for a description of the bus watchdog operation. This is the case for accessing non-implemented addresses in the address map.
- User-mode access to the AEIM registers.
- Byte or half-word access to the AEIM registers.

48.4.11 Show Cycles

The RAHB can perform data transfers to or from internal modules without using the external bus. But for debugging purposes, it is desirable to have the internal address and data bus appear on the external bus. These external bus cycles, called show cycles, are enabled by the SHEN[1:0] bits in the AEIM_CONFIG register.

When show cycles are enabled, the AEIM does the following:

- Drives the internal address bus onto the external address bus pins. The Address lines will reflect the address of the RAHB even during burst accesses.
- The internal data bus hdata[31:0] is driven onto the external data bus pins aeim_data_out[31:0]. Half word accesses will be aligned according to Address[1]. Byte accesses will be aligned according to Address[1:0].
- Various system signals, including information from the RAHB is latched and driven externally by the AEIM as the aeim_qstat[4:0] outputs.
- The aeim_strobe output of the AEIM becomes active and available for use as a logic analyzer clock.

The show cycles operation is supported through the AEIM_CONFIG register. The following features are provided:

- Ability to delay external accesses that follow internal accesses, by up to 3 cycles (when EDEAD > 0), in order to ensure that internal accesses are visible externally (SHEN[1:0] control bits).
- Ability to configure the Burst Clock (aeim_bclk_out) output to be output as the internal MCU Clock (BCM control bit).
- Ability to select whether the upper or lower 16-bits of the internal memory bus are visible on the lower 16 data bus pins, in case only a 16-bit data bus is pinned out (HDB control bit).

Refer to Table 48-9 on page 48-24 for detailed description of the show cycles function provided by the AEIM_CONFIG register.

48.5 Real Time Trace Support

The AEIM provides seven Real Time Trace pins for debugging and giving information about the data size, pipeline status, operating mode, and master of the current bus cycle. When the AEIM show cycles mode is enabled, the following output pins are useful in dis-assembly and determining what is happening inside the chip:

aeim_addr[23:0] - Displays the address of the internal access

aeim_data[31:0] - Displays the data read or written internally

aeim_siz[1:0] - Indicates the size of the internal transfer

aeim_show_rw_b - Indicates whether the internal transfer is a read (1) or a write (0)

aeim_qstat[4:0] - Indicates transfer attributes, AHB master, ARM7 status, etc.

aeim_strobe - Indicates when address, attributes, and data should be captured.

aeim_bclk - Can be configured to output the ARM7 platform hCLK.

The aeim_siz[1:0] pins indicate the data size using the encoding shown in Table 48-4. "aeim_siz[1:0] Encoding". The aeim_siz[1:0] pins are latched by the AEIM and held valid for the duration of the cycle.

Table 48-4. aeim_siz[1:0] Encoding

aeim_siz[1]	aeim_siz[0]	Transfer Size
0	0	Byte (8 bits)
0	1	Halfword (16 bit)
1	0	Word (32 bits)
1	1	Reserved

The aeim_qstat[4:0] pins provide additional detailed information on the state of the ARM processor and RAHB bus. The AEIM uses the hprot[1:0], CPTBIT, DBGnEXEC, hmaster[2:0], ccm_lpmode[1:0] and few other signals to encode the aeim_qstat signals for external trace support.

The following table shows the encodings of the aeim_qstat signals for “normal” ARM7 bus accesses. Note that in these cases the aeim_qstat[3:0] bits are equivalent to the hprot[1], hprot[0], CPTBIT, and DBGnEXEC signals respectively. These mappings are valid when aeim_qstat[4]=0.

Table 48-5. aeim_qstat[4:0] Encodings - Normal ARM7 Accesses

aeim_qstat[4]	aeim_qstat[3] = hprot[1] ≈!CPnTRANS	aeim_qstat[2] = hprot[0] ≈ CPnOPC	aeim_qstat[1] =CPTBIT	aeim_qstat[0] =DBGnEXEC	Internal Processor Status
0	0	0	0	0	ARM mode User Opcode Access, Instruction executed
0	0	0	0	1	ARM mode User Opcode Access, Instruction NOT executed
0	0	0	1	0	Thumb mode User Opcode Access, Instruction executed
0	0	0	1	1	Thumb mode User Opcode Access, Instruction NOT executed
0	0	1	0	0	ARM mode User Data Access, Instruction executed
0	0	1	0	1	ARM mode User Data Access, Instruction NOT executed
0	0	1	1	0	Thumb mode User Data Access, Instruction executed
0	0	1	1	1	Thumb mode User Data Access, Instruction NOT executed
0	1	0	0	0	ARM mode Supervisor Opcode Access, Instruction executed
0	1	0	0	1	ARM mode Supervisor Opcode Access, Instruction NOT executed
0	1	0	1	0	Thumb mode Supervisor Opcode Access, Instruction executed
0	1	0	1	1	Thumb mode Supervisor Opcode Access, Instruction NOT executed
0	1	1	0	0	ARM mode Supervisor Data Access, Instruction executed

Table 48-5. aeim_qstat[4:0] Encodings - Normal ARM7 Accesses (Continued)

aeim_qstat[4]	aeim_qstat[3] = hprot[1] ≈!CPnTRANS	aeim_qstat[2] = hprot[0] ≈ CPnOPC	aeim_qstat[1] =CPTBIT	aeim_qstat[0] =DBGnEXEC	Internal Processor Status
0	1	1	0	1	ARM mode Supervisor Data Access, Instruction NOT executed
0	1	1	1	0	Thumb mode Supervisor Data Access, Instruction executed
0	1	1	1	1	Thumb mode Supervisor Data Access, Instruction NOT executed

Further information is encoded on the aeim_qstat[4:0] signals. The following table is relevant for cases where the ARM7 processor is in debug mode, stopped, not in control of the AHB, or has an access aborted.

Table 48-6. aeim_qstat[4:0] Encodings - Special Cases

aeim_qstat [4]	aeim_qstat [3]	aeim_qstat [2]	aeim_qstat [1]	aeim_qstat [0]	Internal Processor/Chip Status
1	0	0	0	0	Watchpoint ABORTED Instruction fetch, Instruction executed
1	0	0	0	1	Watchpoint ABORTED Instruction fetch, Instruction NOT executed
1	0	0	1	0	Illegal/Timeout ABORTED Instruction fetch, Instruction executed
1	0	0	1	1	Illegal/Timeout ABORTED Instruction fetch, Instruction NOT executed
1	0	1	0	0	RESERVED
1	0	1	0	1	Alt. master (typically CCM) idle cycle.
1	0	1	1	0	Illegal/Timeout ABORTED Data fetch, Instruction executed
1	0	1	1	1	Illegal/Timeout ABORTED Data fetch, Instruction NOT executed
1	1	0	0	0	ARM7 in STOP mode; AHB inactive
1	1	0	0	1	ARM7 in WAIT mode; AHB inactive
1	1	0	1	0	ARM7 in DOZE mode; AHB inactive
1	1	0	1	1	ARM7 in DEBUG mode; AHB inactive
1	1	1	0	0	Alt. master 1 (GEM) in control of AHB
1	1	1	0	1	Alt. master 2 (DMAC) in control of AHB
1	1	1	1	0	Alt. master 3 (HAC) in control of AHB
1	1	1	1	1	AHB idle cycle (NOT a wait state)

The AEIM provides a strobe signal (aeim_strobe) that can be used to sample the AEIM outputs using a logic analyzer, or post-processor hardware. The rising edge of the strobe signal occurs when all AEIM address attribute outputs and aeim_data are known to be valid. Thus, the rising edge of the strobe can be used to capture all AEIM real-time trace outputs.

The AEIM also provides the aeim_show_rw_b output. This indicates whether the visible cycle is a READ (1) or a WRITE (0) access.

In order to save power and reduce noise, when show cycles are disabled, none of the debug signals toggle.

48.6 AEIM Programming Model

All AEIM registers may only be accessed as words (32 bits) in supervisor mode. Accesses in any other mode are ignored and hresp=ERROR is generated.

Complete decoding of the register address is not performed, so mirroring will occur within the register bank's memory space. In order to preserve software upward compatibility, the registers should only be accessed at their lowest possible address.

48.6.1 Register Summary

Figure 0-8.

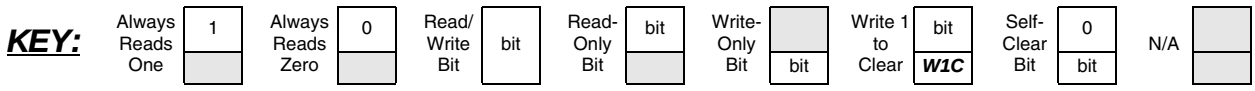


Table 48-7. AEIM Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AEIM_CONFIG ($2888_0000)	R	0	0	0	0	0	0	0	0	GPO[19:0]							
	W																
	R	GPO[19:0]												BCM	HDB	SHEN[1:0]	
	W																
RESERVED ($2888_0004) ($2888_0008) ($2888_000C)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
CS0_PRIM_CFG ($2888_0010)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	LBAEN	AWO	0	DPSIZ[2:0]			EBM	0	SO	RO	0	0	BEN	CSEN
	W																
CS0_SEC_CFG ($2888_0014)	R	0	0	0	0	0	0	0	0	CSWID [3:0]				0	0	CSIDLE [1:0]	
	W																
	R	OEASS [3:0]				0	0	EDEAD [1:0]		0	EBASS [2:0]			0	EBNEG [2:0]		
	W																
CS0_WS_CTL ($2888_0018)	R	0	0	0	0	0	0	WWAIT [1:0]		0	0	IWAIT [5:0]					
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	BWAIT [3:0]			
	W																
CS0_BCLK_CTL ($2888_001C)	R	0	0	BFEED	PBC	0	IBDLY [2:0]			0	0	0	0	BBDIR	BBDLY [2:0]		
	W																
	R	IDIV [1:0]		BDIV [1:0]		0	IPULSE [2:0]			0	0	0	0	0	0	0	0
	W																
CSx_PRIM_CFG CS1:($2888_0020) CS2:($2888_0030) CS3:($2888_0040) CS4:($2888_0050) CS5:($2888_0060)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	LBAEN	AWO	0	DPSIZ [2:0]			EBM	0	SO	RO	CSPA	0	BEN	CSEN
	W																
CSx_SEC_CFG CS1:($2888_0024) CS2:($2888_0034) CS3:($2888_0044) CS4:($2888_0054) CS5:($2888_0064)	R	0	0	0	0	0	0	0	0	CSWID [3:0]				0	0	CSIDLE [1:0]	
	W																
	R	OEASS [3:0]				0	0	EDEAD [1:0]		0	EBASS [2:0]			0	EBNEG [2:0]		
	W																

Table 48-7. AEIM Register Summary

CSx_WS_CTL CS1:($2888_0028)	R	0	0	0	0	0	0	WWAIT [1:0]		0	0	IWAIT]5:0]					
	W																
CS2:($2888_0038) CS3:($2888_0048) CS4:($2888_0058) CS5:($2888_0068)	R	0	0	0	0	0	0	0	0	0	0	0	0	BWAIT [3:0]			
	W																
CSx_BCLK_CTL CS1:($2888_002C)	R	0	0	BFEED	PBC	0	IBDLY [2:0]			0	0	0	0	BBDIR	BBDLY [2:0]		
	W																
CS2:($2888_003C) CS3:($2888_004C) CS4:($2888_005C) CS5:($2888_006C)	R	IDIV [1:0]		BDIV [1:0]		0	IPULSE [2:0]			0	0	0	0	0	0	0	0
	W																

48.6.2 Quick Reference To CS Control Bit Fields

Table 48-8. CS Control Bit Fields

Bit Field Name	Description
Primary Configuration Register	
LBAEN	LBA Enable - 1 bit, enables LBA assertion in associated chip select
AWO	Atomic Write Only - 1 bit, disables writes smaller than the port width, i.e. byte write to half-word port
DPSIZ	Data Port Size - 3 bits, port width and alignment
EBM	Enable Byte Mode - 1 bit, asserts EB on writes only or on both reads and writes
SO	Supervisor Only - 1 bit, restricts access to supervisor only
RO	Read Only - 1 bit, restricts access to read only
CSPA	Chip Select Pin Assert - 1 bit, sets state of CS pin while CS is disabled
BEN	Burst Enable - 1 bit, enables burst mode
CSEN	Chip Select Enable - 1 bit, enables CS
Secondary Configuration Register	
CSWID	Chip Select Width control - 4 bits, assertion/negation delay narrowing CS, units of clks
CSIDLE	Chip Select Idle time - 2 bits, in ***additional*** clks (1 is for free)
OEASS	OE Assertion delay - 4 bits, delay in clk phases
EDEAD	Extra Dead Cycles - 2 bits, dead cycles when transitioning to ***another*** CS
EBASS	Enable Byte Assertion delay - 3 bits, units of clk phases
EBNEG	Enable Byte Negation time - 3 bits, units of clk phases
Wait State Control Register	
WWAIT	Write Wait States - 2 bits, in ***extra*** clocks required for a write access
IWAIT	Initial Access Wait Count - 6 bits, wait state control
BWAIT	Burst Access Wait Count - 4 bits, wait state control
Burst Clock Control Register	
BFEED	BCLK Feedback Data Strobe - 1 bit, selects BCLK for capturing data into AEIM
PBC	Parameter Block Control - 1 bit, forces async accesses to Intel parameter block
IBDLY	Initial Phase Burst Clock Delay - 3 bits, delay in clk phases
BBDIR	Burst Phase Burst Clock Delay Direction - 1 bit, direction of BBDLY from beginning of burst
BBDLY	Burst Phase Burst Clock Delay - 3 bits, delay in clk phases

Table 48-8. CS Control Bit Fields

Bit Field Name	Description
IDIV	Initial Phase Burst Clock Divisor - 2 bits (divisors of 1,2,3,4)
BDIV	Burst Phase Burst Clock Divisor - 2 bits (divisors of 1,2,3,4)
IPULSE	Initial Phase Pulse Count - 3 bits, units of BCLK

48.6.3 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various AEIM registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Resets to logic 0.
  - **1**: Resets tologic 1.
  - **?**: Reset state is unknown.
  - **u**: Unaffected by reset.

AEIM_CONFIG AEIM Configuration Register **$2888_0000**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
									GPO [19:0]							
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	GPO [19:0]												BCM	HDB	SHEN 1	SHEN 0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 48-9. AEIM Configuration Register Description

Name	Description	Settings
Bits 31 -24	**AEIM Configuration Register Reserved—** These bits are reserved for future expansion. They are read as zero and should be written with zero for upward compatibility.	N/A
GPO[19:0] Bits 23-4	**General Purpose Outputs—** These bits are output only software controlled registers. They are forced to zero at reset, and can be read or written by the CPU. Reads will return either the reset value, or the previously written value.	0 - The general purpose register output is driven low. 1 - The general purpose register output is driven high. For Neptune specific usage of these GPO bits, please refer to “IO Pad Drive Programming” section, in chapter “Chip Configuration and Memory Maps” (Chapter 6) in Neptune Specification.
BCM Bit 3	**Burst Clock Mode—** This bit is used to select the burst clock mode of operation. When set, the BCLK will act as the MCU_CLK. This bit is cleared by hardware reset. **Note:** Setting this bit will cause other burst modes to fail!	0 - The burst clock only runs when accessing a chip select range with the BEN bit set. When the burst clock is not running it will remain in a logic 0 state. When the burst clock is running it will be configured by the CS Burst Clock Control Register. 1 - The burst clock runs all the time. (Independent of CS accesses).

Table 48-9. AEIM Configuration Register Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>HDB
Bit 2</td><td>High Data Bus— This bit is used to determine which byte lanes of the internal data bus are driven onto the lower 16 external data bus bits when show cycles are enabled. This bit is ignored if SHEN is cleared.
This bit is cleared by hardware reset.</td><td>0 - Lower internal data bus bits D15-D0 are driven externally.

1 - Upper internal data bus bits D31-D16 are driven externally.</td></tr>
<tr><td>SHEN[1:0]
Bits 1-0</td><td>Show Cycle Enable — These two bits are used to determine what the AEIM does with the external bus during internal transfers (e.g. an access to the internal ROM, RAM or internal peripherals). When show cycles are enabled, the internal address and data bus are driven externally.
When Show cycles is disabled, the external address lines will not toggle during burst accesses. Additionally, the PSTAT[3:0], SIZ[1:0], SHOW_RW_B, and STROBE outputs of the AEIM are disabled.
These bits are cleared by hardware reset.</td><td>
<table>
<tr><th>SHEN1-SHEN0</th><th>Show Cycles Function</th></tr>
<tr><td>00</td><td>Show cycles are disabled. The external address bus is driven with the last valid external address.</td></tr>
<tr><td>01</td><td>Show cycles enabled. Internal termination to the RAHB during idle cycles (caused by EDEAD being set) follows normal operation. This means that internal transfers that occur during EDEAD idle cycles will not be visible externally.</td></tr>
<tr><td>10</td><td>Show cycles enabled. Internal termination to the MCU during idle cycles (caused by EDEAD being set) is delayed by the appropriate number of clocks. This ensures that all internal transfers can be externally monitored, at the expense of performance.</td></tr>
<tr><td>11</td><td>Reserved</td></tr>
</table>
</td></tr>
</table>

CS0_PRIM_CFG — CS0 Primary Configuration Register — $2888_0010

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
			LBAEN	AWO		DPSIZ [2:0]			EBM		SO	RO			BEN	CSEN
TYPE	r	r	rw	rw	r	rw	rw	rw	rw	r	rw	rw	r	r	rw	rw
RESET	0	0	1	0	0	aeim_cs0_dsz[2:0]			1	0	0	0	0	0	0	1

CS1_PRIM_CFG — $2888_0020
CS2_PRIM_CFG — $2888_0030
CS3_PRIM_CFG — $2888_0040
CS4_PRIM_CFG — $2888_0050
CS5_PRIM_CFG — $2888_0060

CS1 - CS5 Primary Configuration Register

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
			LBAEN	AWO		DPSIZ [2:0]			EBM		SO	RO	CSPA		BEN	CSEN
TYPE	r	r	rw	rw	r	rw	rw	rw	rw	r	rw	rw	rw	r	rw	rw
RESET	0	0	1	0	0	0	1	0	1	0	0	0	1	0	0	0

Table 48-10. CS Primary Configuration Register Descriptions

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>Bits 31-14</td><td>CSx Primary Configuration Register Reserved— These bits are reserved for future expansion. They are read as zero and should be written with zero for upward compatibility.</td><td>N/A</td></tr>
<tr><td>LBAEN
Bit 13</td><td>LBA Enable— This bit enables LBA pin assertions for the associated chip select. Some intel burst flash memories require LBA to assert for asynchronous accesses (Siretta). To allow boot from these types of burst flash memories, LBA must assert for all accesses until the chip select can be configured appropriately for the type of memory being used.
This bit is ignored on synchronous accesses i.e. read accesses with burst mode enabled (BEN=1).

This bit is set by hardware reset.</td><td>0 - LBA will not assert for asynchronous accesses to this chip select.

1 - LBA will assert for all asynchronous accesses to this chip select.</td></tr>
<tr><td>AWO
Bit 12</td><td>Atomic Write Only— This bit disables writes that are smaller than the port size, i.e.- byte write to 16-bit port, byte or half-word write to 32-bit port. This feature is used to protect devices that do not support byte writes. While this bit is clear, undersized writes are allowed to 16 and 32-bit ports, but the byte(s) that are not addressed will be written with unknown data.

This bit is cleared by hardware reset.</td><td>0 - writes smaller than the port size are allowed. Programmer beware for half-word-only devices, unknown data will be written to the unaddressed byte.

1 - writes smaller than the port size are ignored and hresp=ERROR asserted.</td></tr>
<tr><td>Bit 11</td><td>CSx Primary Configuration Register Reserved— This bit is reserved for future expansion. It is read as zero and should be written with zero for upward compatibility.</td><td>N/A</td></tr>
<tr><td>DPSIZ[2:0]
Bits 10-8</td><td>Data Port Size— This field defines the width and alignment of the external device data port. At hardware reset, the value of these bits are ’010’ for all chip selects except CS0. For CS0, the reset state of these bits is directly controlled by the aeim_cs0_dsz[2:0] ports on the AEIM module boundary. This allows various boot configurations if allowed by the system integration. For more information on boot configurations refer to Section 48.4.5.

These bits are set to ‘010’ (except CS0) by hardware reset.</td><td><table><tr><th>DPSIZ</th><th>Port Size</th></tr><tr><td>000</td><td>8-bit Port, resides on D[15:8]</td></tr><tr><td>001</td><td>8-bit Port, resides on D[7:0]</td></tr><tr><td>010</td><td>16-bit Port</td></tr><tr><td>011</td><td>32-bit Port</td></tr><tr><td>1xx</td><td>Reserved</td></tr></table></td></tr>
<tr><td>EBM
Bit 7</td><td>Enable Byte Mode— This bit is used to indicate which access types should assert the Enable Byte outputs EB[3:0].

This bit is set by hardware reset.</td><td>0 - Read and Write accesses cause EB[3:0] to be asserted, thus configuring them as byte enables.

1 - Only Write accesses cause EB[3:0] to be asserted, thus configuring them as byte write enables. The EB[3:0] outputs should be configured as byte write enables for accesses to dual or quad x8 memories.</td></tr>
</table>

Table 48-10. CS Primary Configuration Register Descriptions (Continued)

Name	Description	Settings
Bit 6	**CSx Primary Configuration Register Reserved—** This bit is reserved for future expansion. It is read as zero and should be written with zero for upward compatibility.	N/A
SO Bit 5	**Supervisor Only—** This bit restricts access to this chip select memory space to be supervisor mode only. This bit is cleared by hardware reset.	0 - User and Supervisor mode accesses are allowed in the range of this chip select. 1 - User mode accesses are prohibited. A user mode access to this chip select is ignored and hresp=ERROR asserted.
RO Bit 4	**Read Only—** This bit restricts access to this chip select memory space to be read-only. This bit is cleared by hardware reset.	0 - Writes are allowed in the range of this chip select. 1 - Writes are prohibited. A write to this chip select is ignored and hresp=ERROR asserted.
CSPA Bit 3	**Chip Select Pin Assert—** This bit controls the Chip Select pin while it is operating as a programmable output pin (i.e. the CSEN bit is clear). This bit is not available for CS0. **ECB Polarity Control**— If the Chip Select is enabled (CSEN=1) with external termination mode (IWAIT[5:0]=111111) for an asynchronous access (BEN=0) then, this bit controls the polarity of the external termination input “aeim_ecb_b”. For burst mode (BEN=1), aeim_ecb_b is treated as an active low input irrespective of CSPA. This bit is set by hardware reset.	If CSEN=0: 0 - drives Chip Select output to logic-low (CS5 is logic-high) 1 - drives Chip Select output to logic-high (CS5 is logic-low) If CSEN=1, BEN=0 and IWAIT[5:0] = 111111: 0 - aeim_ecb_b input is treated as active LOW input. 1 - aeim_ecb_b input is treated as active HIGH input. **Otherwise:** CSPA bit is ignored.
Bit 2	**CSx Primary Configuration Register Reserved—** This bit is reserved for future expansion. It is read as zero and should be written with zero for upward compatibility.	N/A
BEN Bit 1	**Burst Enable—** This bit is used to enable burst mode. While enabled, the AEIM is capable of interfacing to burstable flash devices through the use of additional burst control signals: Burst Clock (BCLK), LBA (Load Burst Address) and ECB (End Current Burst). The sequencing of these additional I/O’s is controlled by other AEIM configuration register bit settings. This bit is cleared by hardware reset.	0 - Burst mode is disabled. 1 - Burst mode is enabled.
CSEN Bit 0	**Chip Select Enable—** This bit controls the operation of the chip select pin, and the cycle termination status of access to the CS address range. This bit is cleared by hardware reset for CS1 through CS5. For CS0 this bit is set by hardware reset.	0 - Chip Select function is disabled. Any access to the memory space of this Chip Select is ignored and hresp=ERROR asserted. 1 - Chip Select is enabled, and is asserted when a legal access falls within the CS’ saddress range.

CS0_SEC_CFG		**$2888_0014**
CS1_SEC_CFG		**$2888_0024**
CS2_SEC_CFG	CS0 - CS5 Secondary Configuration Register	**$2888_0034**
CS3_SEC_CFG		**$2888_0044**
CS4_SEC_CFG		**$2888_0054**
CS5_SEC_CFG		**$2888_0064**

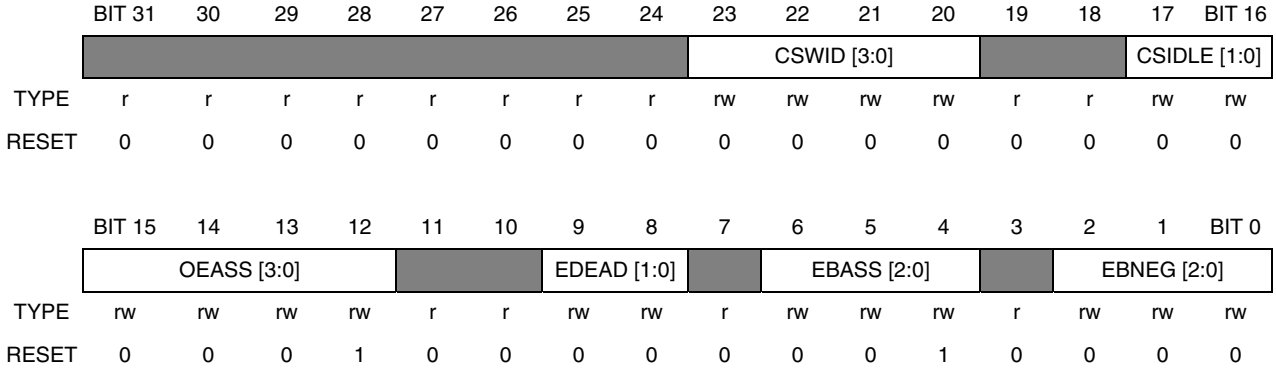


Table 48-11. CS Secondary Configuration Register Descriptions

Name	Description	Settings
Bits 31-24	**CSx Secondary Configuration Register Reserved—** These bits are reserved for future expansion. They are read as zero and should be written with zero for upward compatibility.	N/A

Table 48-11. CS Secondary Configuration Register Descriptions (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>CSWID[3:0]
Bits 23-20</td><td>Chip Select Width— These bits are used for devices that require additional address setup time and additional address/data hold times. They determine when Chip Select is asserted and if an idle cycle is inserted between back-to-back external transfers. If IWAIT=000000, these bits are ignored. CSWID does not effect chip select timing, or add idle cycles between transfers if BEN = 1
When CSWID=0000, chip select is asserted normally, i.e. as early as possible. No idle cycle is inserted between back-to-back external transfers. When CSWID>0000, chip select is asserted late during both read and write cycles and may be negated early (refer to the table at right for specifics). In addition, an idle cycle is inserted between back-to-back external transfers.
These bits function independently of the IWAIT bits (unless IWAIT=000000), so it is possible to program a combination of IWAIT and CSWID that would result in the chip select never being asserted for an external access. In order to ensure this will not happen, program the IWAIT field to be more than twice the value of the CSWID field.
In the special case where IWAIT=111111 (external cycle termination via aeim_ecb_b), the chip select will assert as determined by the CSWID field, but negation will not occur until aeim_ecb_b is seen as asserted (polarity of which is controlled by CSPA bit).

These bits are cleared by hardware reset.</td><td>
<table>
<tr><th>CSWID</th><th>CS Assertion Delay From Start of Access</th><th>CS Negation Time Before End of Access</th></tr>
<tr><td>0000</td><td>0 clocks</td><td>0 clocks</td></tr>
<tr><td>0001</td><td>1 clock</td><td>0 clock</td></tr>
<tr><td>0010</td><td>2 clocks</td><td>1 clocks</td></tr>
<tr><td>0011</td><td>3 clocks</td><td>2 clocks</td></tr>
<tr><td>...</td><td>...</td><td>...</td></tr>
<tr><td>1110</td><td>14 clocks</td><td>13 clocks</td></tr>
<tr><td>1111</td><td>15 clocks</td><td>14 clocks</td></tr>
</table>
</td></tr>
<tr><td>Bits 19-18</td><td>CSx Secondary Configuration Register Reserved— These bits are reserved for future expansion. They are read as zero and should be written with zero for upward compatibility.</td><td>N/A</td></tr>
</table>

Table 48-11. CS Secondary Configuration Register Descriptions (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>CSIDLE[1:0]
Bits 17-16</td><td>Chip Select Idle Control— These bits are used for devices that require chip select to remain negated for a minimum period of time. When CSIDLE=00, chip select negates and reasserts with a minimum negation time of 1 clock. When CSIDLE>00, chip select is forced to remain negated for multiple clocks. The minimum negation time is determined by the CSIDLE field value.
Whenever consecutive accesses are made to a chip select, the chip select remains asserted for the duration of the accesses. If accesses are non-consecutive, then the CSIDLE field determines the minimum negation time of the chip select pin.

These bits are cleared by hardware reset.</td><td><table><tr><th>CSIDLE</th><th>CS Minimum Idle Time
(# Clocks)</th></tr><tr><td>00</td><td>1</td></tr><tr><td>01</td><td>2</td></tr><tr><td>10</td><td>3</td></tr><tr><td>11</td><td>4</td></tr></table></td></tr>
<tr><td>OEASS[3:0]
Bits 15-12</td><td>Output Enable Assertion Delay— These bits are used to determine when $\overline{OE}$ is asserted during read cycles. This is useful for preventing data bus contention between memories, when read one after the other, and when using the $\overline{OE}$ output to control the $\overline{BAA}$ pin on an AMD burst flash memory.
When burst mode is enabled (BEN=1), after the initial burst access, $\overline{OE}$ is asserted continuously for subsequent burst accesses, and is not affected by OEASS (see burst read timing diagrams for more detail). This allows maximum performance both for data bus contention prevention and when using $\overline{OE}$ to control $\overline{BAA}$.

These bits are set to ‘0001’ by hardware reset.</td><td><table><tr><th>OEASS</th><th>OE Assertion delay from Start of Cycle
(# Clock Edges)</th></tr><tr><td>0000</td><td>0</td></tr><tr><td>0001</td><td>1</td></tr><tr><td>0010</td><td>2</td></tr><tr><td>...</td><td>...</td></tr><tr><td>1110</td><td>14</td></tr><tr><td>1111</td><td>15</td></tr></table></td></tr>
<tr><td>Bits 11-10</td><td>CSx Secondary Configuration Register Reserved— These bits are reserved for future expansion. They are read as zero and should be written with zero for upward compatibility.</td><td>N/A</td></tr>
</table>

Table 48-11. CS Secondary Configuration Register Descriptions (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>EDEAD[1:0]
Bits 9-8</td><td>Extra Dead Cycles— These bits are used to add idle cycles inserted after an external read cycle to eliminate external data bus contention. This is useful for slow memory and peripherals that have long CS or OE to output data three-state times. Any subsequent external access will be affected (delayed) if EDEAD is non-zero with the exception of another read to the same chip select. This case is not affected since there could be no databus collision between an external memory and itself.
The extra dead cycle is added by delaying the start of the following external access (if no internal cycle intervenes between the two external accesses) by the EDEAD number of clock cycles. The wait state count is not affected on the chip select pin with EDEAD active.

These bits are cleared by hardware reset.</td><td><table><tr><th>EDEAD</th><th>Number of Dead Cycles Between Back-to-Back External Accesses to Different CS’s
(# Clocks)</th></tr><tr><td>00</td><td>0</td></tr><tr><td>01</td><td>1</td></tr><tr><td>10</td><td>2</td></tr><tr><td>11</td><td>3</td></tr></table></td></tr>
<tr><td>Bit 7</td><td>CSx Secondary Configuration Register Reserved— This bit is reserved for future expansion. It is read as zero and should be written with zero for upward compatibility.</td><td>N/A</td></tr>
</table>

Table 48-11. CS Secondary Configuration Register Descriptions (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>EBASS[2:0]
Bits 6-4</td><td>Enable Byte Assertion Delay— These bits are used to determine when the enable byte (EB) and aeim_rw_b (R/W) pins are asserted during a write to external memory. They cause the EB and R/W pins to delay their assertion from the beginning of the external bus cycle. This field is programmable in clock PHASES. The delay of the EB and R/W pin assertion will occur regardless of the state of the EBM bit (whether EBs are byte lane enables or write strobes). If the EB pins are used as byte lane enables, asserting them later will cause the write to be begun later, affording additional setup time on the address and data lines. EBASS=0 is a dangerous configuration for the following reasons:
Back to back writes (no idle clocks in between) will not allow the EB pins to negate, possibly causing invalid data to be written to multiple locations as address transitions with EB asserted.
EB will assert with the same timing as CS, and the external databus pins will be driven immediately with potentially invalid data (data is not valid until slightly after the following clock edge). This is a waste of power causing false transitions, and possibly causes contention on the databus if the previous access was an external read.
These bits are set to ‘001’ by hardware reset.</td><td><table><tr><th>EBASS</th><th>EB and R/W Assertion delay from Start of Cycle
(# Clock Edges)</th></tr><tr><td>000</td><td>0</td></tr><tr><td>001</td><td>1</td></tr><tr><td>010</td><td>2</td></tr><tr><td>...</td><td>...</td></tr><tr><td>110</td><td>6</td></tr><tr><td>111</td><td>7</td></tr></table></td></tr>
<tr><td>Bit 3</td><td>CSx Secondary Configuration Register Reserved— This bit is reserved for future expansion. It is read as zero and should be written with zero for upward compatibility.</td><td>N/A</td></tr>
<tr><td>EBNEG[2:0]
Bits 2-0</td><td>Enable Byte Negation Delay— These bits are used to determine when the enable byte (EB) and aeim_rw_b (R/W) pins are negated during an external memory access. They cause the EB and R/W pins to negate before the end of the external bus cycle. This field is programmable in clock PHASES.
The delay of the EB pin negation will occur regardless of the state of the EBM bit (whether EBs are byte lane enables or write strobes). If the EB pins are used as byte lane enables, negating them early will cause the write to be completed earlier (from the memory point of view), affording additional hold time on the address and data lines.

These bits are cleared by hardware reset.</td><td><table><tr><th>EBNEG</th><th>EB and R/W Negation delay Before End of Cycle
(# Clock Edges)</th></tr><tr><td>000</td><td>0</td></tr><tr><td>001</td><td>1</td></tr><tr><td>010</td><td>2</td></tr><tr><td>...</td><td>...</td></tr><tr><td>110</td><td>6</td></tr><tr><td>111</td><td>7</td></tr></table></td></tr>
</table>

CS0_WS_CTL	CS0 - CS5 Wait State Control Register	**$2888_0018**
CS1_WS_CTL		**$2888_0028**
CS2_WS_CTL		**$2888_0038**
CS3_WS_CTL		**$2888_0048**
CS4_WS_CTL		**$2888_0058**
CS5_WS_CTL		**$2888_0068**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
							WWAIT [1:0]				IWAIT [5:0]					
TYPE	r	r	r	r	r	r	rw	rw	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
													BWAIT [3:0]			
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 48-12. CS Wait State Control Register Descriptions

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>Bits 31-26</td><td>CSx Wait State Control Register Reserved— These bits are reserved for future expansion. They are read as zero and should be written with zero for upward compatibility.</td><td>N/A</td></tr>
<tr><td>WWAIT[1:0]
Bits 25-24</td><td>Write Wait States— These bits are used to determine if additional wait-states are added for write cycles. This is useful for writing to flash memories that require additional data setup time. These bits have no effect on read cycles.

These bits are cleared by hardware reset.</td><td><table><tr><th>WWAIT</th><th>Number of Additional Cycles Added for External Write Accesses</th></tr><tr><td>00</td><td>0</td></tr><tr><td>01</td><td>1</td></tr><tr><td>10</td><td>2</td></tr><tr><td>11</td><td>3</td></tr></table></td></tr>
<tr><td>Bits 23-22</td><td>CSx Wait State Control Register Reserved— These bits are reserved for future expansion. They are read as zero and should be written with zero for upward compatibility.</td><td>N/A</td></tr>
</table>

Table 48-12. CS Wait State Control Register Descriptions (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>IWAIT[5:0]
Bits 21-16</td><td>Initial Access Wait State Control— When burst mode is disabled (BEN = 0):
The IWAIT bits program the number of wait-states for an access to the external device connected to the associated Chip Select. When WWAIT is cleared, setting IWAIT=000000 results in one clock accesses, IWAIT=000001 results in 2 clock transfers, etc. until IWAIT=111110 results in 62 clock transfers. Setting IWAIT=111111 configures the AEIM to continually insert wait states until either the ECB input gets asserted (polarity of which is controlled by CSPA bit), or the bus watchdog counter expires (hresp=ERROR asserts).
The user would typically set WSC=000000 and WWAIT=00 for access to fast SRAM devices (one-clock read and write access) - CSWID=0000, OEASS=0000, IWAIT=000001 and WWAIT=00 for access to normal SRAM (two-clock read and write access) - CSWID=0000, OEASS=0000, IWAIT=000001 and WWS=01 for access to Flash memory (two-clock read access and three-clock write access) and CSWID and IWAIT to the appropriate values for access to an LCD controller.
When burst mode is enabled (BEN= 1):
These six bits program the number of additional system clock cycles required for the initial access of a burst sequence initiated by the AEIM to an external burst device. This value is specified for each individual burst device. Refer to the AEIM synchronous burst read timing diagrams for further detail.

These bits are initialized to ’111110’ (62 wait states) by hardware reset.</td><td>Synchronous:
<table><tr><th>IWAIT</th><th>BEN = 1
Initial Access Time (clks)</th></tr><tr><td>000000</td><td>1</td></tr><tr><td>000001</td><td>2</td></tr><tr><td>000010</td><td>3</td></tr><tr><td>....</td><td>....</td></tr><tr><td>111110</td><td>63</td></tr><tr><td>111111</td><td>Controlled by aeim_ecb_b input</td></tr></table>Asynchronous:<table><tr><th>IWAIT</th><th>BEN = 0
Read Access Time (clks)</th><th>BEN = 0
Write Access Time (clks)</th></tr><tr><td>000000</td><td>1</td><td>WWAIT + 1</td></tr><tr><td>000001</td><td>2</td><td>WWAIT + 2</td></tr><tr><td>000010</td><td>3</td><td>WWAIT + 3</td></tr><tr><td>....</td><td>....</td><td>....</td></tr><tr><td>111110</td><td>63</td><td>WWAIT + 63</td></tr><tr><td>111111</td><td colspan="2">Controlled by aeim_ecb_b input</td></tr></table></td></tr>
<tr><td>Bits 15-12</td><td>CSx Wait State Control Register Reserved— These bits are reserved for future expansion. They are read as zero and should be written with zero for upward compatibility.</td><td>N/A</td></tr>
</table>

Table 48-12. CS Wait State Control Register Descriptions (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>Bits 11-4</td><td>CSx Wait State Control Register Reserved— These bits are reserved for future expansion. They are read as zero and should be written with zero for upward compatibility.</td><td>N/A</td></tr>
<tr><td>BWAIT[3:0]
Bits 3-0</td><td>Burst Access Wait State Control— These bits are used to specify the expected number of wait states that will be needed for burst reads, and only affect AEIM operation when burst mode is enabled (BEN=1). As burst clock frequencies increase, it may become necessary to add wait states in order to compensate for memory access time, chip routing and pad delays, and AEIM data setup time requirements. Table on right shows the encoding of the BWAIT field.

These bits are cleared by hardware reset.</td><td>
<table>
<tr><th>BWAIT</th><th>Wait States on Normal Burst Accesses
(# Clocks)</th></tr>
<tr><td>0000</td><td>0</td></tr>
<tr><td>0001</td><td>1</td></tr>
<tr><td>0010</td><td>2</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>1110</td><td>14</td></tr>
<tr><td>1111</td><td>15</td></tr>
</table>
</td></tr>
</table>

CS0_BCLK_CTL **$2888_001C**
CS1_BCLK_CTL **$2888_002C**
CS2_BCLK_CTL CS0 - CS5 Burst Clock Control Register **$2888_003C**
CS3_BCLK_CTL **$2888_004C**
CS4_BCLK_CTL **$2888_005C**
CS5_BCLK_CTL **$2888_006C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
			BFEED	PBC		IBDLY [2:0]							BBDIR	BBDLY [2:0]		
TYPE	r	r	rw	rw	r	rw	rw	rw	r	r	r	r	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	IDIV [1:0]		BDIV [1:0]			IPULSE [2:0]										
TYPE	rw	rw	rw	rw	r	rw	rw	rw	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 48-13. CS Burst Clock Control Register Descriptions

Name	Description	Settings
Bits 31-30	**CSx Burst Clock Control Register Reserved—** These bits are reserved for future expansion. They are read as zero and should be written with zero for upward compatibility.	N/A
BFEED Bit 29	**Burst Block Feedback Control—** This bit selects between two read data latching mechanisms in the AEIM. It is ignored unless the chip select is configured into burst mode (BEN=1). The BCLK output pad feeds back the BCLK through the pad input buffer; the input to the AEIM is BCLKin. The delays incurred in chip level routing and the pad output and input buffers are modeled by this signal. When BFEED is set, read data is captured on the rising edges of BCLKin for initial and bursted accesses. This bit is cleared by hardware reset.	0 - Normal data capture is used (i.e. the same method employed for asynchronous reads). 1 - Burst clock feedback capture mode is used. Data is captured by the rising edge of the BCLK signal fed back from its output pad.

Table 48-13. CS Burst Clock Control Register Descriptions (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>PBC
Bit 28</td><td>Parameter Block Control— This bit is used to configure operation for an Intel burst flash memory containing a parameter block (a non-burstable address range subset of the memory). This bit is ignored unless burst mode is enabled.
For parameter block control to function correctly, the Intel burst flash must be configured into the bottom boot mode of operation. This locates the parameter block so that it spans addresses 000000 - 00FFFF of the memory, the lower 64K bytes. This approach is only compatible with a 64K byte parameter block, so it would be unsuitable for dual Intel memories on a 32-bit bus.
This bit is cleared by hardware reset.</td><td>0 - Parameter block mode disabled. The entire burst flash address range is considered burstable by the AEIM.

1 - Parameter block mode enabled. The lower 64K bytes of the burst flash address range is considered non-burstable by the AEIM. No burst accesses will be attempted to this range (all accesses will be initiated and terminated as initial accesses).</td></tr>
<tr><td>Bit 27</td><td>CSx Burst Clock Control Register Reserved— This bit is reserved for future expansion. It is read as zero and should be written with zero for upward compatibility.</td><td>N/A</td></tr>
<tr><td>IBDLY[2:0]
Bits 26-24</td><td>Initial Access BCLK Delay— These bits are used to control the system clock phase on which the external burst clock begins to transition during the initial access portion of an access to a burst memory. This degree of control is important to achieve the minimum number of wait states necessary in a given system timing environment. The starting point of the BCLK is programmable in clock EDGES from the beginning of the memory access. The first point at which a BCLK rising edge can occur is the rising edge of MCU_CLK coincident with the assertion of chip select.
It is the responsibility of the programmer to configure the chip select in such a way that BCLK will be low when control of the BCLK pin is handed off to the burst phase control bits. The handoff point is determined by IWAIT if BBDLY is zero. If BBDLY is non-zero, the handoff occurs BBDLY clock phases before (BBDIR=1) or after (BBDIR=0) IWAIT elapses. If BCLK is not low when handoff occurs, the BCLK output will behave unpredictably, and correct operation is not guaranteed.

These bits are cleared by hardware reset.</td><td><table><tr><th>IBDLY</th><th>Edge of MCU CLK that Corresponds to First BCLK Rising Edge of Initial Access</th></tr><tr><td>000</td><td>First Rising Edge of MCU_CLK</td></tr><tr><td>001</td><td>First Falling Edge of MCU_CLK</td></tr><tr><td>...</td><td>...</td></tr><tr><td>110</td><td>Fourth Rising Edge of MCU_CLK</td></tr><tr><td>111</td><td>Fourth Falling Edge of MCU_CLK</td></tr></table></td></tr>
<tr><td>Bits 23-20</td><td>CSx Burst Clock Control Register Reserved— These bits are reserved for future expansion. They are read as zero and should be written with zero for upward compatibility.</td><td>N/A</td></tr>
</table>

Table 48-13. CS Burst Clock Control Register Descriptions (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>BBDIR
Bit 19</td><td>Burst Access BCLK Delay Direction Control— This bit is used to control the direction of the delay for the burst phase BCLK specified by the Burst Access Clock Delay (BBDLY) field. If the direction is negative, the BCLK for the burst phase will start a specified number of clock phases before the burst phase is entered. If the direction is positive, the BCLK for the burst phase will start a specified number of clock phases after the burst phase is entered. The burst phase is considered to begin on the rising edge of the MCU_CLK that is used to capture initial access data into the AEIM.
This degree of BCLK control is important to achieve optimum pipelining of delay paths allowing the minimum number of wait states necessary in a given system timing environment.

This bit is cleared by hardware reset.</td><td>0 - Burst clock delay direction is positive (i.e. burst phase BCLK starts BBDLY edges after the burst phase begins)

1 - Burst clock delay direction is negative (i.e. burst phase BCLK starts BBDLY edges before the burst phase begins)

If BBDLY=0, burst phase BCLK starts on the rising edge of the MCU_CLK that is used to capture initial access data into the AEIM.</td></tr>
<tr><td>BBDLY[2:0]
Bits 18-16</td><td>Burst Access BCLK Delay— This bit is used to control the system clock phase on which the external burst clock begins to transition during the burst access phase of an access to a burst memory. The burst phase is considered to begin on the rising edge of the MCU_CLK that is used to capture initial access data into the AEIM.
The starting point of the BCLK is programmable in clock EDGES from the beginning of the burst phase memory access. The SIGN (direction) of the delay is controlled by the BBDIR bit.
It is the responsibility of the programmer to configure the chip select in such a way that BCLK will be low when control of the BCLK pin is handed off to the burst phase control bits. The handoff point is determined by IWAIT if BBDLY is zero. If BBDLY is non-zero, the handoff occurs BBDLY clock phases before (BBDIR=1) or after (BBDIR=0) IWAIT elapses. If BCLK is not low when handoff occurs, the BCLK output will behave unpredictably, and correct operation is not guaranteed.

These bits are cleared by hardware reset.</td><td><table><tr><th>BBDLY</th><th>Edge of MCU CLK that Corresponds to First BCLK Rising Edge of Burst Access</th></tr><tr><td>000</td><td>First Rising Edge of MCU_CLK</td></tr><tr><td>001</td><td>First Falling Edge of MCU_CLK</td></tr><tr><td>...</td><td>...</td></tr><tr><td>110</td><td>Fourth Rising Edge of MCU_CLK</td></tr><tr><td>111</td><td>Fourth Falling Edge of MCU_CLK</td></tr></table></td></tr>
<tr><td>IDIV[1:0]
Bits 15-14</td><td>Initial Access Clock Divisor— These bits control the division of the burst clock during the initial access portions of an access to a burst memory. These bits are useful to avoid exceeding the maximum frequency allowed by an external burst device, and can help minimize system power dissipation by preventing unnecessary signal transitions.

These bits are cleared by hardware reset.</td><td><table><tr><th>IDIV</th><th>Burst Clock Frequency for Initial Access</th></tr><tr><td>00</td><td>MCU_CLK</td></tr><tr><td>01</td><td>MCU_CLK / 2</td></tr><tr><td>10</td><td>MCU_CLK / 3</td></tr><tr><td>11</td><td>MCU_CLK / 4</td></tr></table></td></tr>
</table>

Table 48-13. CS Burst Clock Control Register Descriptions (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>BDIV[1:0]
Bits 13-12</td><td>Burst Access Clock Divisor— These bits control the division of the burst clock during the burst access portion of an access to a burst memory. These bits are useful to avoid exceeding the maximum frequency allowed by an external burst device, and can help minimize system power dissipation by preventing unnecessary signal transitions.

These bits are cleared by hardware reset.</td><td><table><tr><th>BDIV</th><th>Burst Clock Frequency for Burst Access</th></tr><tr><td>00</td><td>MCU_CLK</td></tr><tr><td>01</td><td>MCU_CLK / 2</td></tr><tr><td>10</td><td>MCU_CLK / 3</td></tr><tr><td>11</td><td>MCU_CLK / 4</td></tr></table></td></tr>
<tr><td>Bit 11</td><td>CSx Burst Clock Control Register Reserved— This bit is reserved for future expansion. It is read as zero and should be written with zero for upward compatibility.</td><td>N/A</td></tr>
<tr><td>IPULSE[2:0]
Bits 10-8</td><td>Initial Access Pulse Count— These bits are used to set the number of burst clock pulses emitted by the AEIM whenever an initial access to burst memory is performed. If the BCM bit is set (BCM = 1) in the AEIM Configuration Register, this will override the IPULSE field and cause the burst clock to run continuously.

If the pulse count is programmed for more pulses than will fit in an initial access, then the pulse train will be limited to the number allowed by the settings of the Initial Access Clock Divisor (IDIV), Initial Wait State Control (IWAIT), and Initial burst clock delay (IBDLY) fields. For example:
- Initial Access Pulse Count field is set to 4,
- Initial Access Clock Divisor set to 2 (divide by 2)
- Initial Access BCLK Delay set to 2 (2 clock phases)
- Wait State Control is set to 3 (3 wait states = a 4 clock access)
The total number of burst clocks emitted will be 2, since each burst clock takes 2 system clocks and the total access takes 4 system clocks. The burst BCLK delay (BBDLY) field can also appear to affect the initial access pulse count, but only if the BCLK Direction (BBDIR) control bit is set to allow a negative delay on the burst phase BCLK. For more information on BBDLY and BBDIR, see their bit descriptions.

These bits are cleared by hardware reset.</td><td><table><tr><th>IPULSE</th><th>Maximum Number of BCLK Pulses Emitted During Initial Access*</th></tr><tr><td>000</td><td>unlimited, pulse mode disabled</td></tr><tr><td>001</td><td>1</td></tr><tr><td>...</td><td>...</td></tr><tr><td>110</td><td>6</td></tr><tr><td>111</td><td>7</td></tr></table>* The actual number of pulses is dependent on IPULSE, IWAIT, IBDLY, and IDIV.</td></tr>
<tr><td>Bits 7-0</td><td>CSx Burst Clock Control Register Reserved— These bits are reserved for future expansion. They are read as zero and should be written with zero for upward compatibility.</td><td>N/A</td></tr>
</table>

48.7 External Bus Timing Diagrams

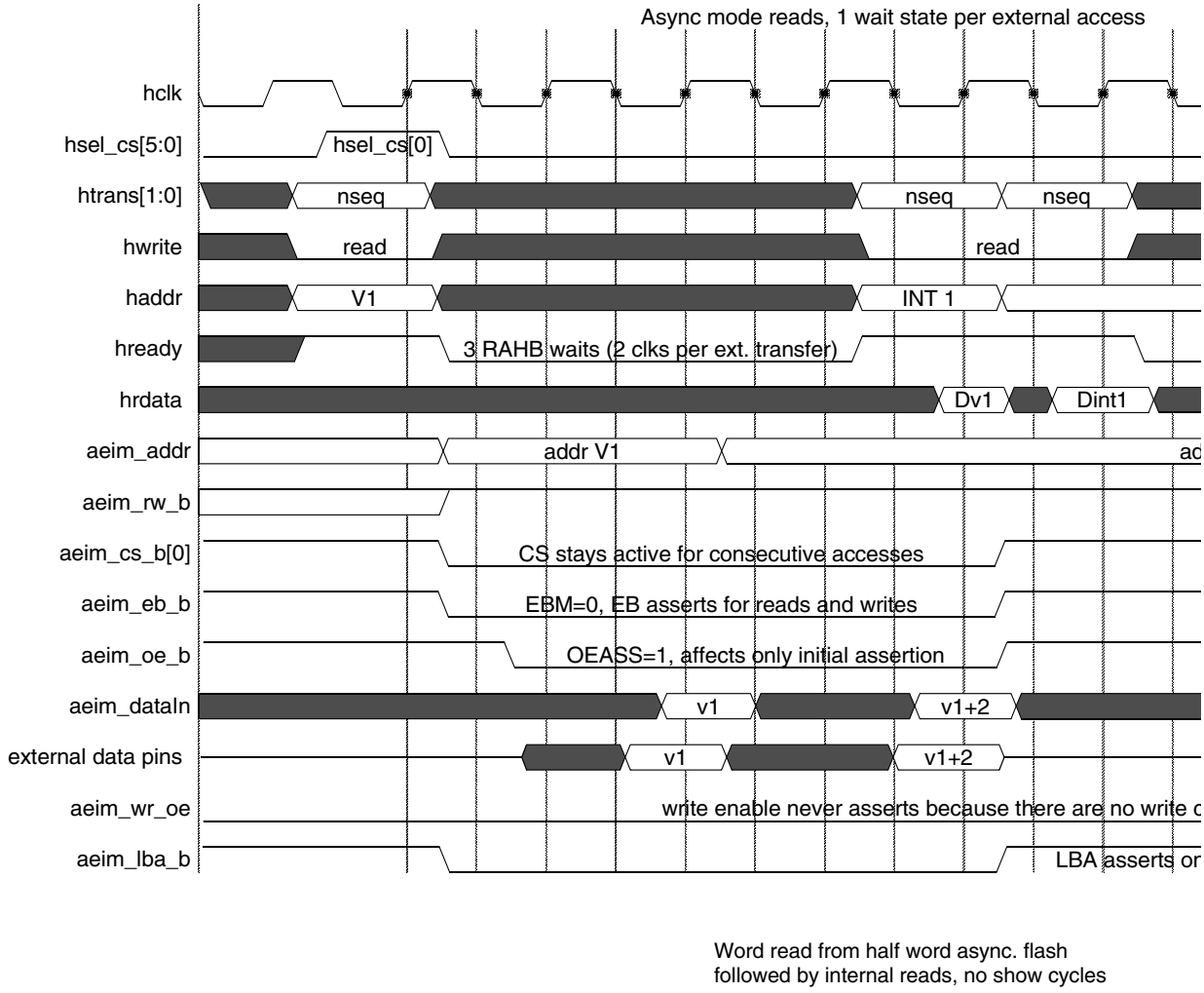


Figure 48-4. Async mode reads, 1 wait state per external access

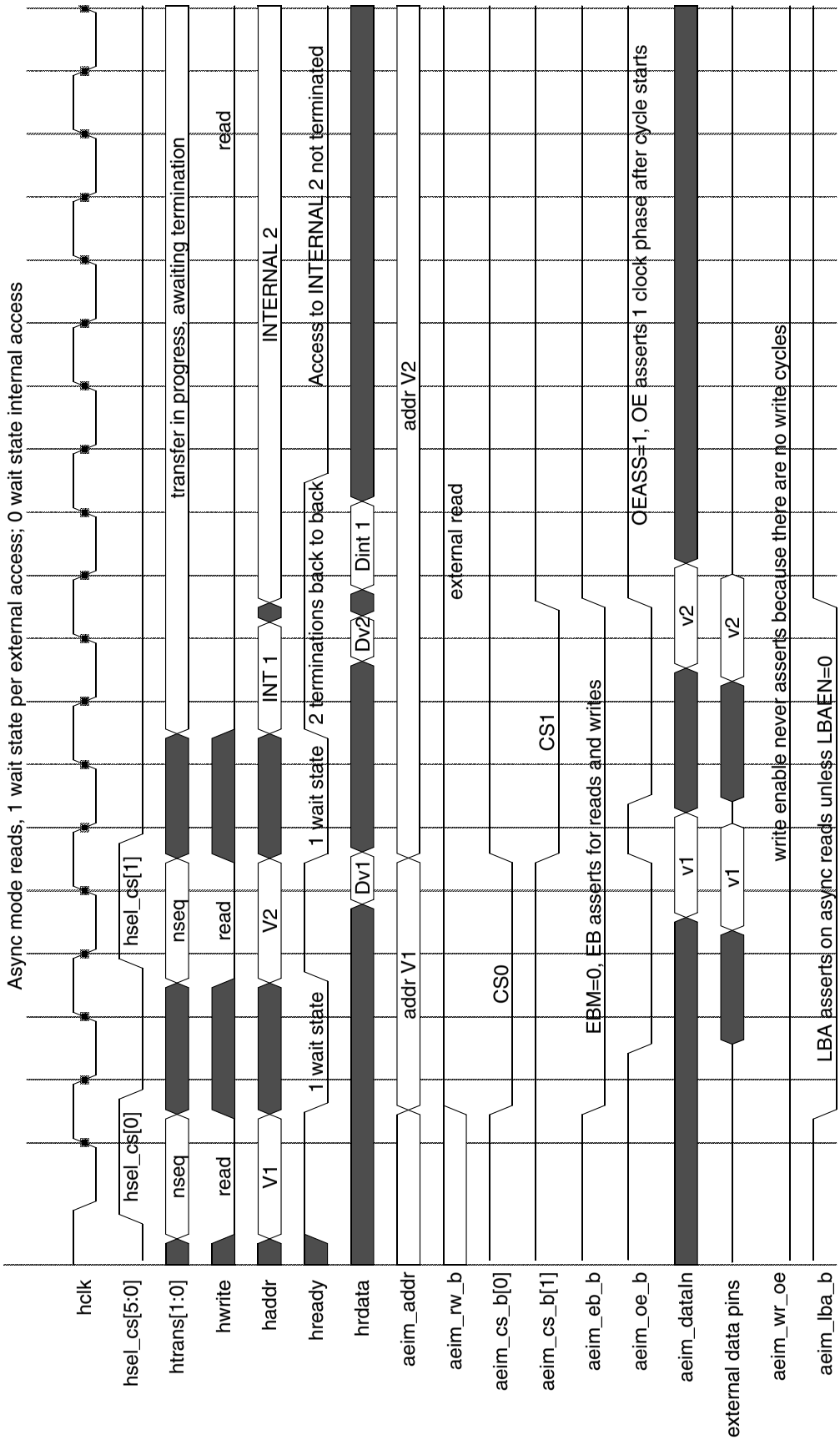


Figure 48-5. Async mode reads, 1 wait state per external access; 0 wait state internal access

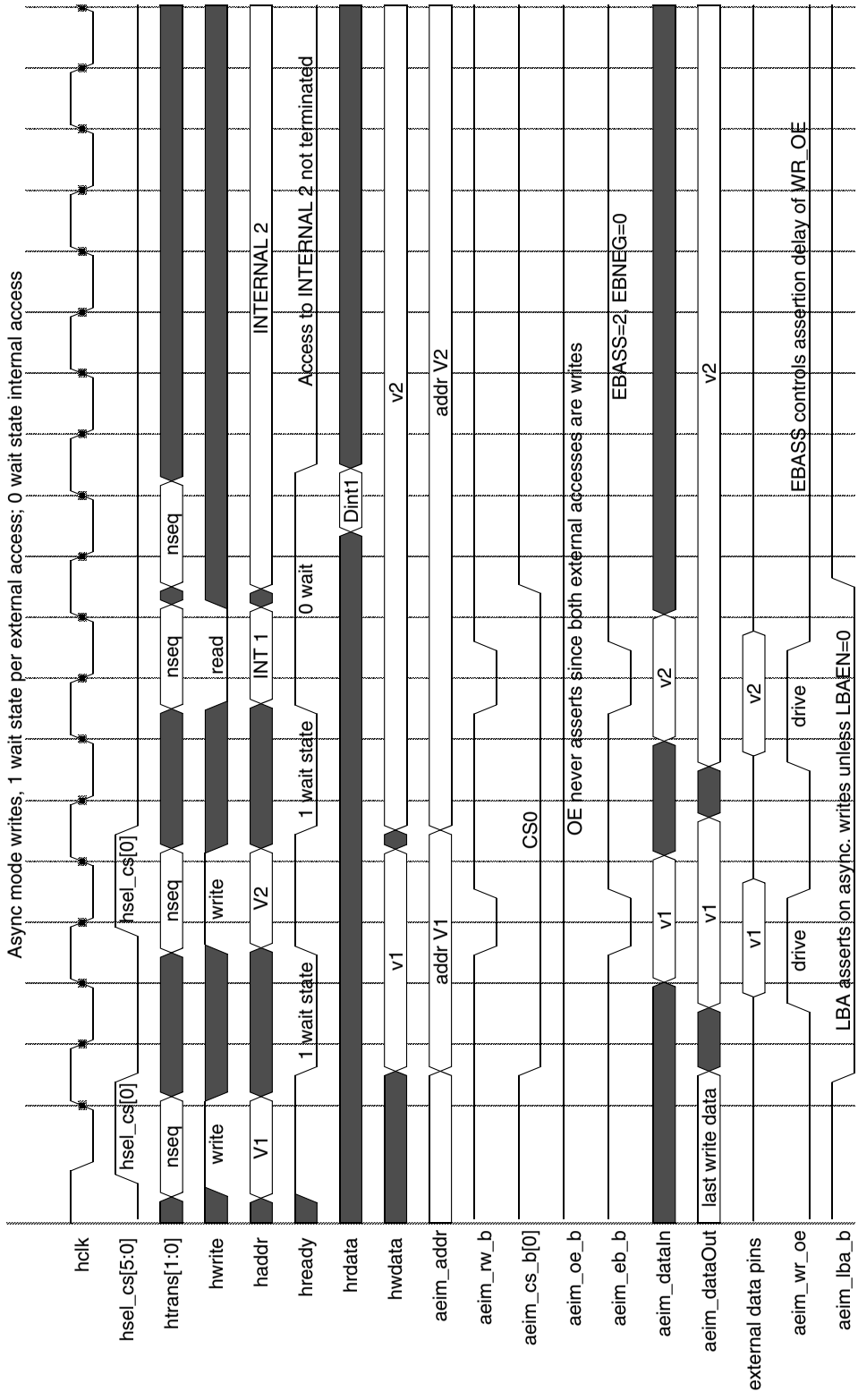


Figure 48-6. Async mode writes, 1 wait state per external access; 0 wait state internal access

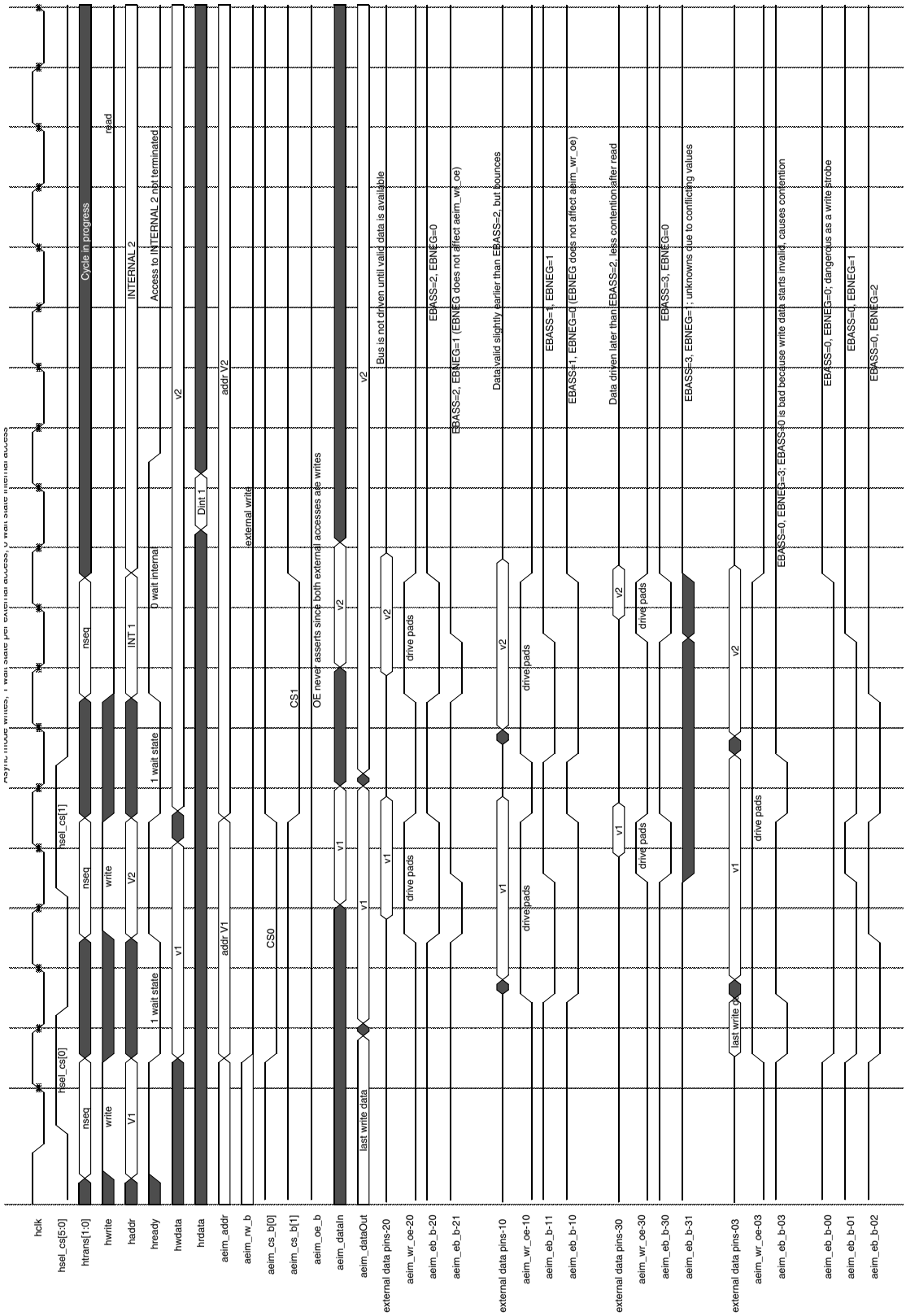

hclk
hsel_cs[5:0]
htrans[1:0]
hwrite
haddr
hready
hwdata
hrdata
aeim_addr
aeim_rw_b
aeim_cs_b[0]
aeim_cs_b[1]
aeim_oe_b
aeim_dataIn
aeim_dataOut
external data pins-20
aeim_wr_oe-20
aeim_eb_b-20
aeim_eb_b-21
external data pins-10
aeim_wr_oe-10
aeim_eb_b-11
aeim_eb_b-10
external data pins-30
aeim_wr_oe-30
aeim_eb_b-30
aeim_eb_b-31
external data pins-03
aeim_wr_oe-03
aeim_eb_b-03
aeim_eb_b-00
aeim_eb_b-01
aeim_eb_b-02
hsel_cs[0]
hsel_cs[1]
nseq
write
V1
V2
INT 1
INTERNAL 2
read
Cycle in progress
1 wait state
0 wait internal
Access to INTERNAL 2 not terminated
v1
v2
Dint 1
addr V1
addr V2
external write
CS0
CS1
OE never asserts since both external accesses are writes
last write data
drive pads
Bus is not driven until valid data is available
Data valid slightly earlier than EBASS=2, but bounces
EBASS=2, EBNEG=1 (EBNEG does not affect aeim_wr_oe)
EBASS=2, EBNEG=0
EBASS=1, EBNEG=1
EBASS=1, EBNEG=0 (EBNEG does not affect aeim_wr_oe)
Data driven later than EBASS=2, less contention after read
EBASS=3, EBNEG=0
EBASS=3, EBNEG=1; unknowns due to conflicting values
EBASS=0, EBNEG=3; EBASS=0 is bad because write data starts invalid, causes contention
EBASS=0, EBNEG=0; dangerous as a write strobe
EBASS=0, EBNEG=1
EBASS=0, EBNEG=2
Hyphenated suffixes refer to EBASS/EBNEG field values for waveform
Half word write to CS0 followed by half word write to CS1, no show cycles; Effect of EBASS and EBNEG fields on EB assertion and negation delays. Note EBASS determines aeim_wr_oe

Figure 48-7.

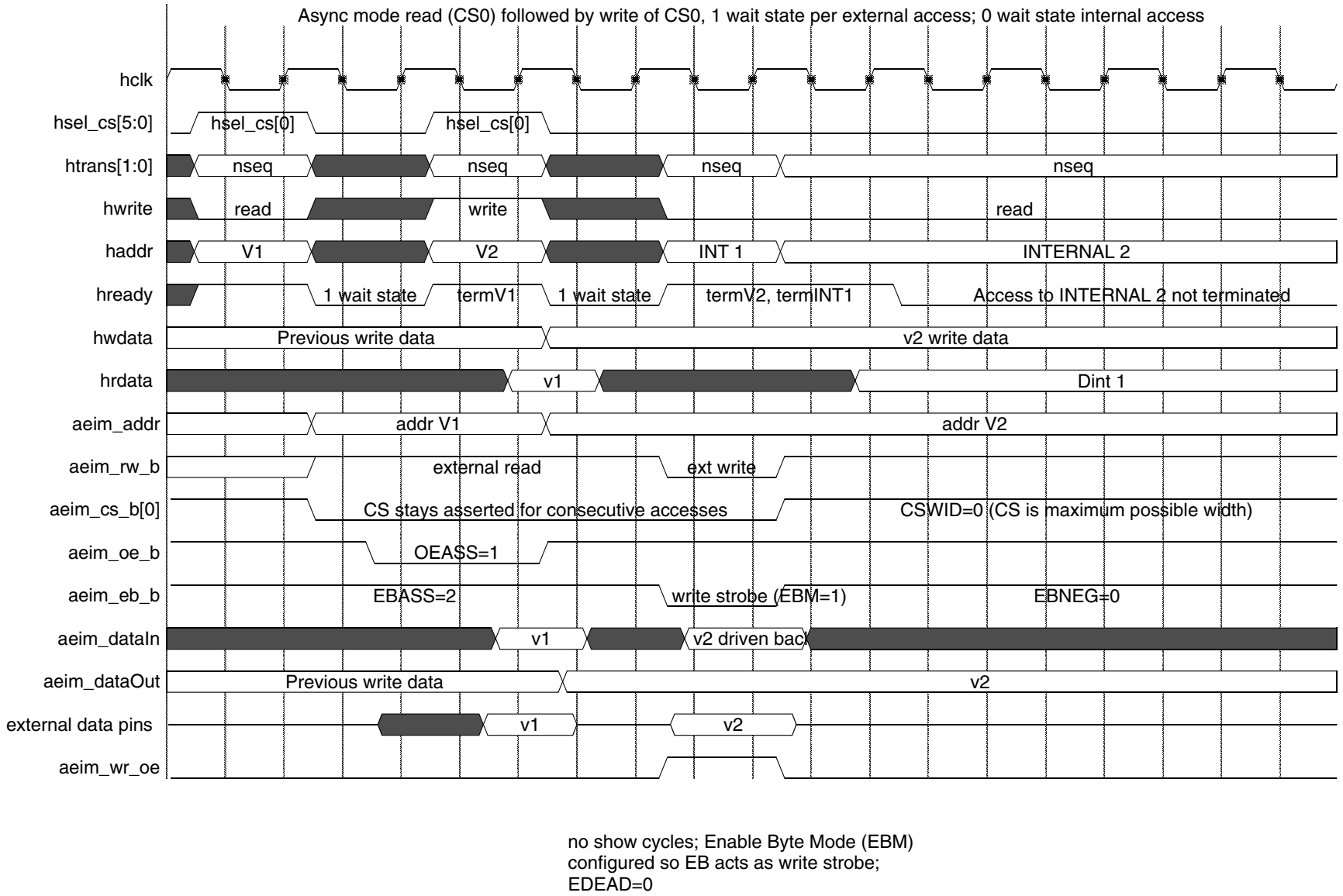


Figure 48-8. Async mode read (CS0) followed by write of CS0, 1 wait state per external access; 0 wait state internal access

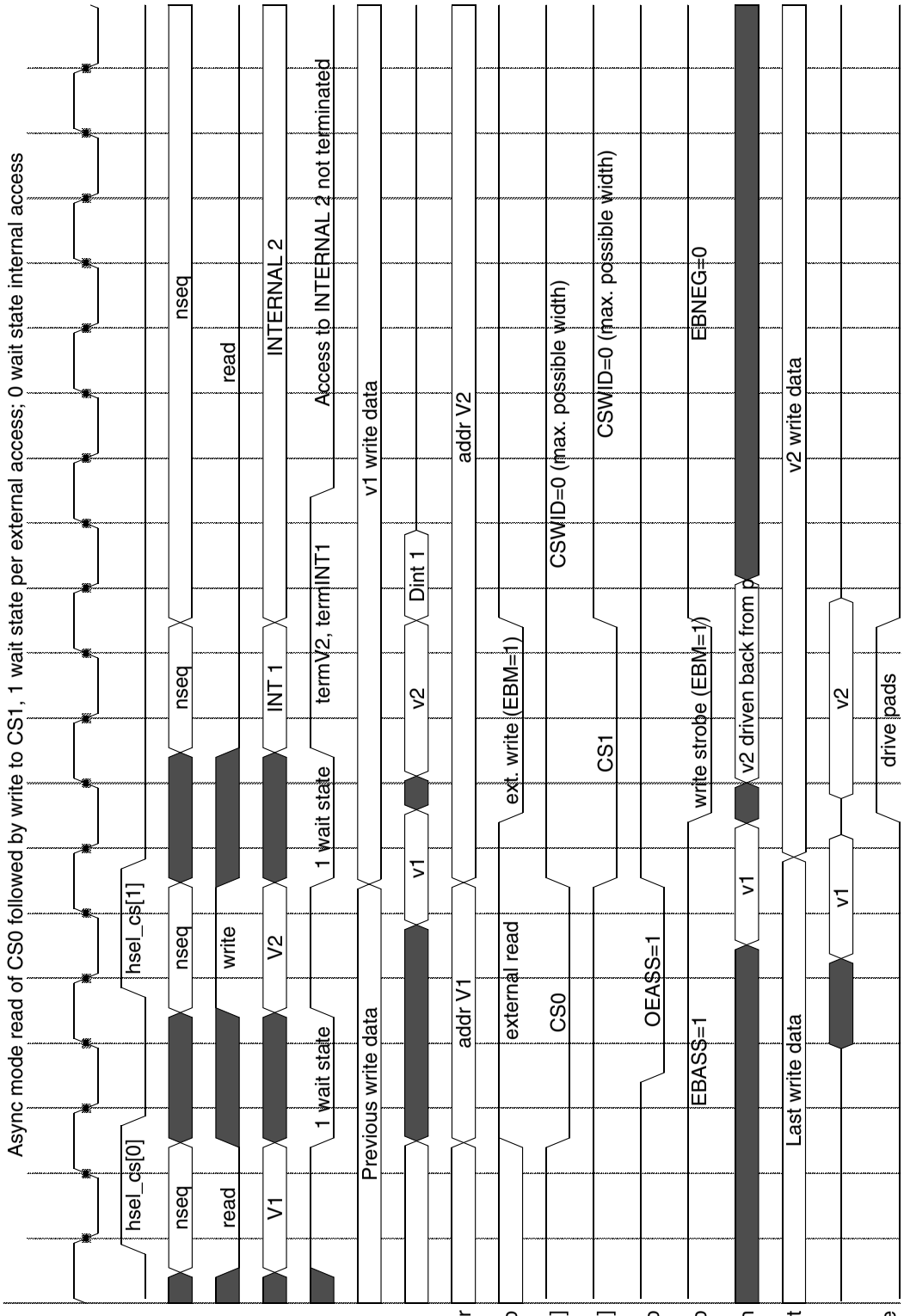


Figure 48-9. Async mode read of CS0 followed by write to CS1, 1 wait state per external access; 0 wait state internal access

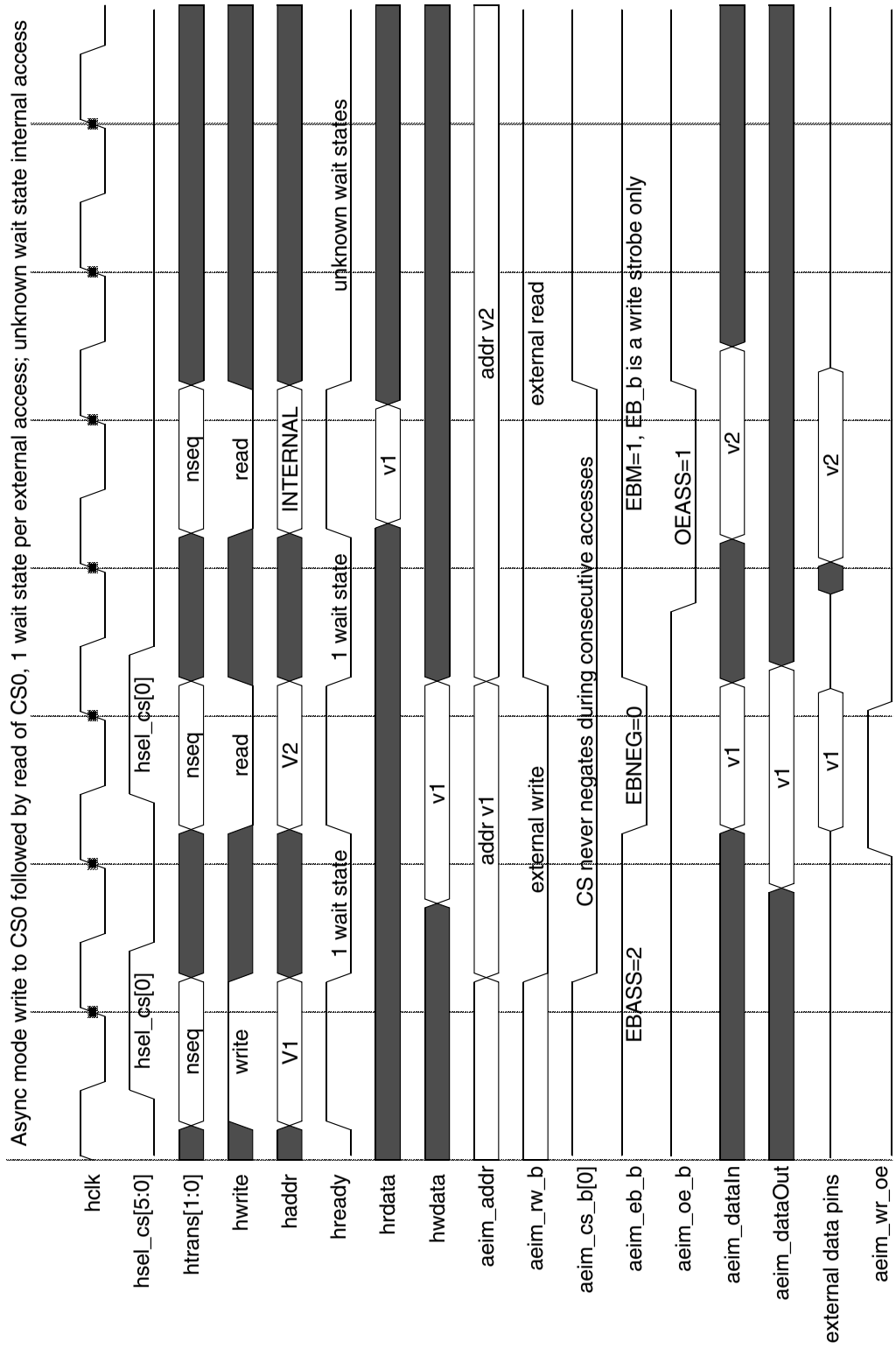


Figure 48-10. Async mode write to CS0 followed by read of CS0, 1 wait state per external access; unknown wait state internal access

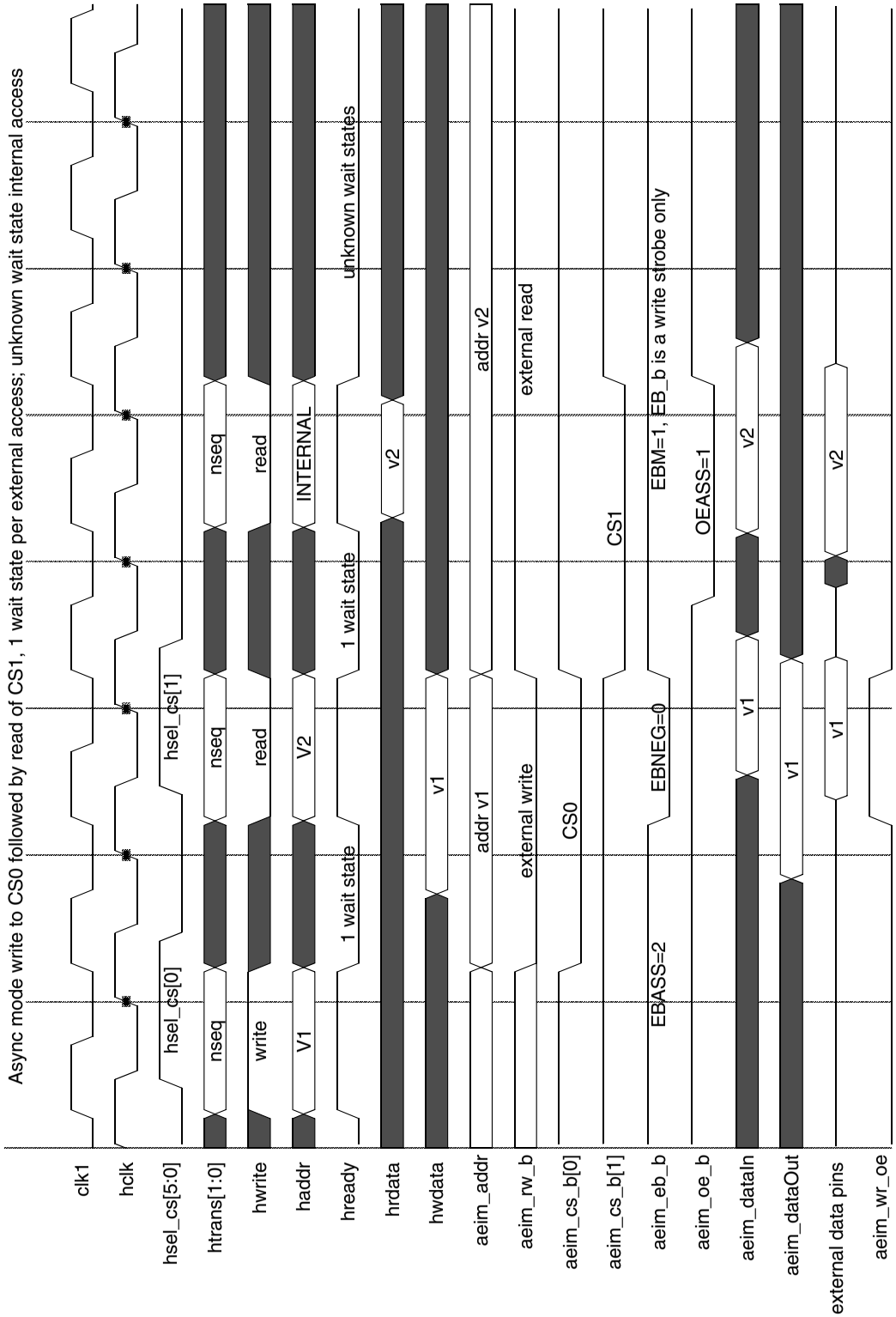


Figure 48-11. Async mode write to CS0 followed by read of CS1, 1 wait state per external access; unknown wait state internal access

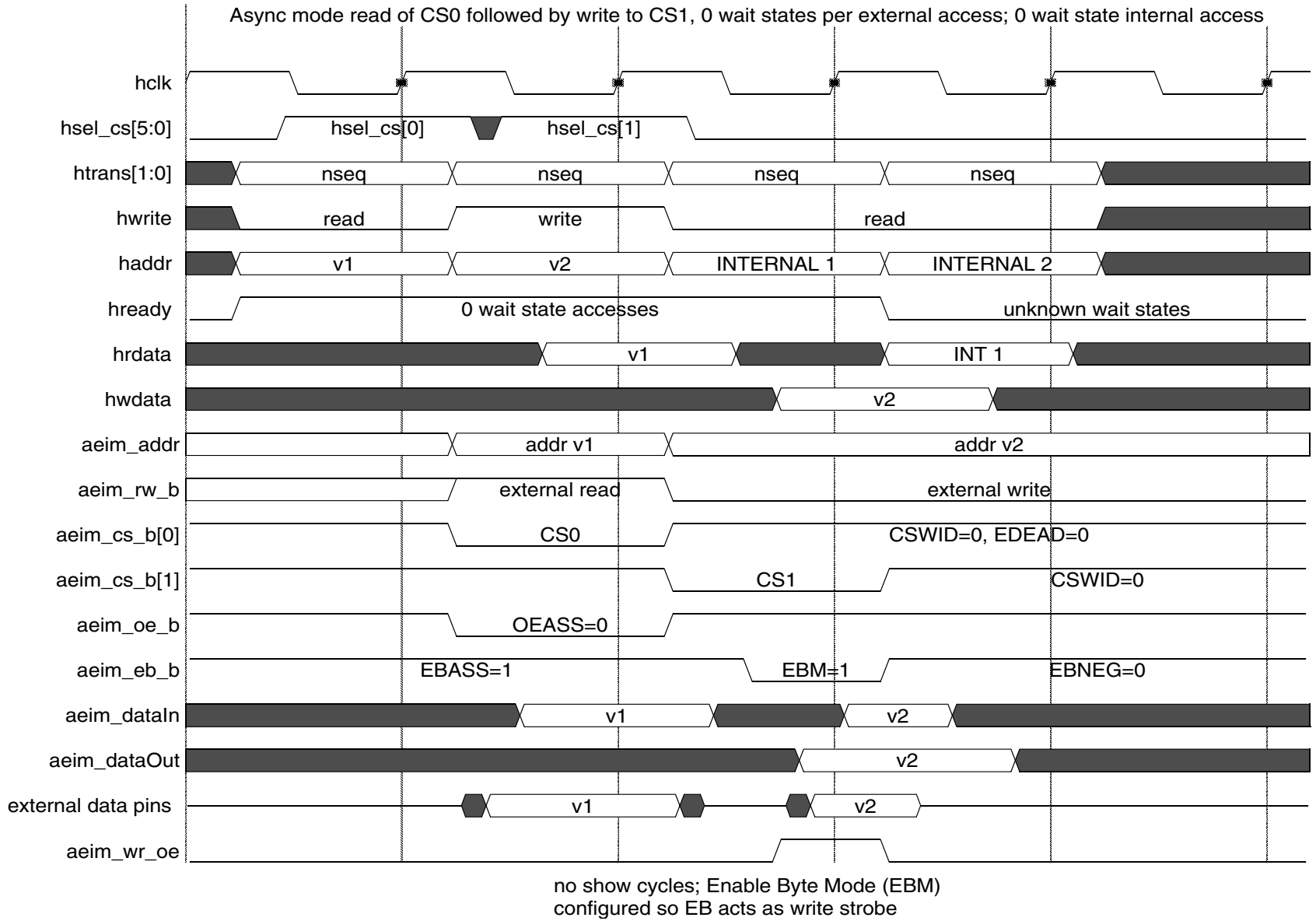
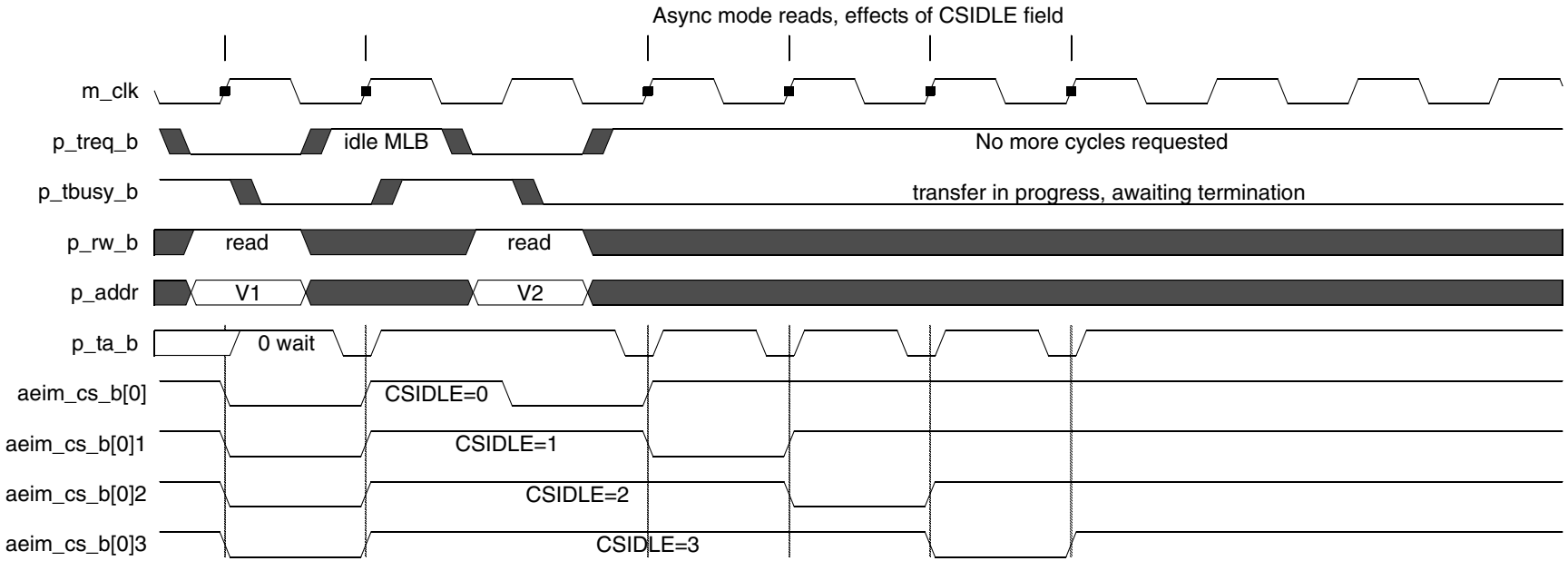
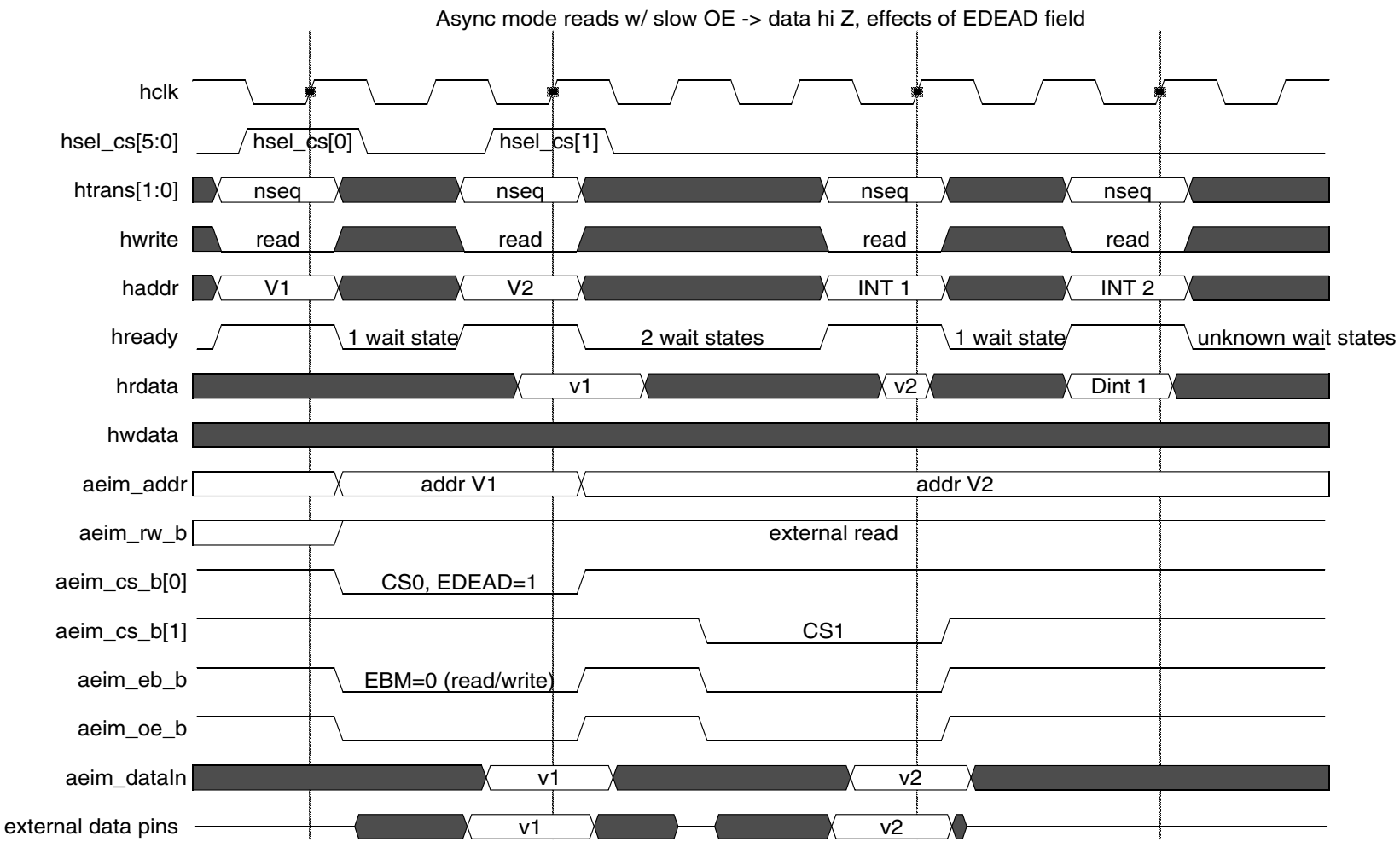


Figure 48-12. Async mode read of CS0 followed by write to CS1, 0 wait states per external access; 0 wait state internal access



Half word read from half word async. flash followed by idle cycle, followed by half word read from same flash. Flash is 0 wait state. Various TA assertions depict the times TA WOULD BE ASSERTED for each associated CSIDLE setting. The start of the 2nd CS0 access is delayed by 1 clock (idle MLB) plus the # of

Figure 48-13. Async mode reads, effects of CSIDLE field



Half word read from half word async. flash followed by half word read from another flash. Both flashes are one wait state, but since CS0 EDEAD=1 and an immediate access is attempted to CS1, the start of the CS1 access is delayed by 1 clock, and the effect to the MCU is an extra wait state when accessing CS1. This prevents external data bus contention between the devices.

Figure 48-14. Async mode reads w/ slow OE -> data hi Z, effects of EDEAD field

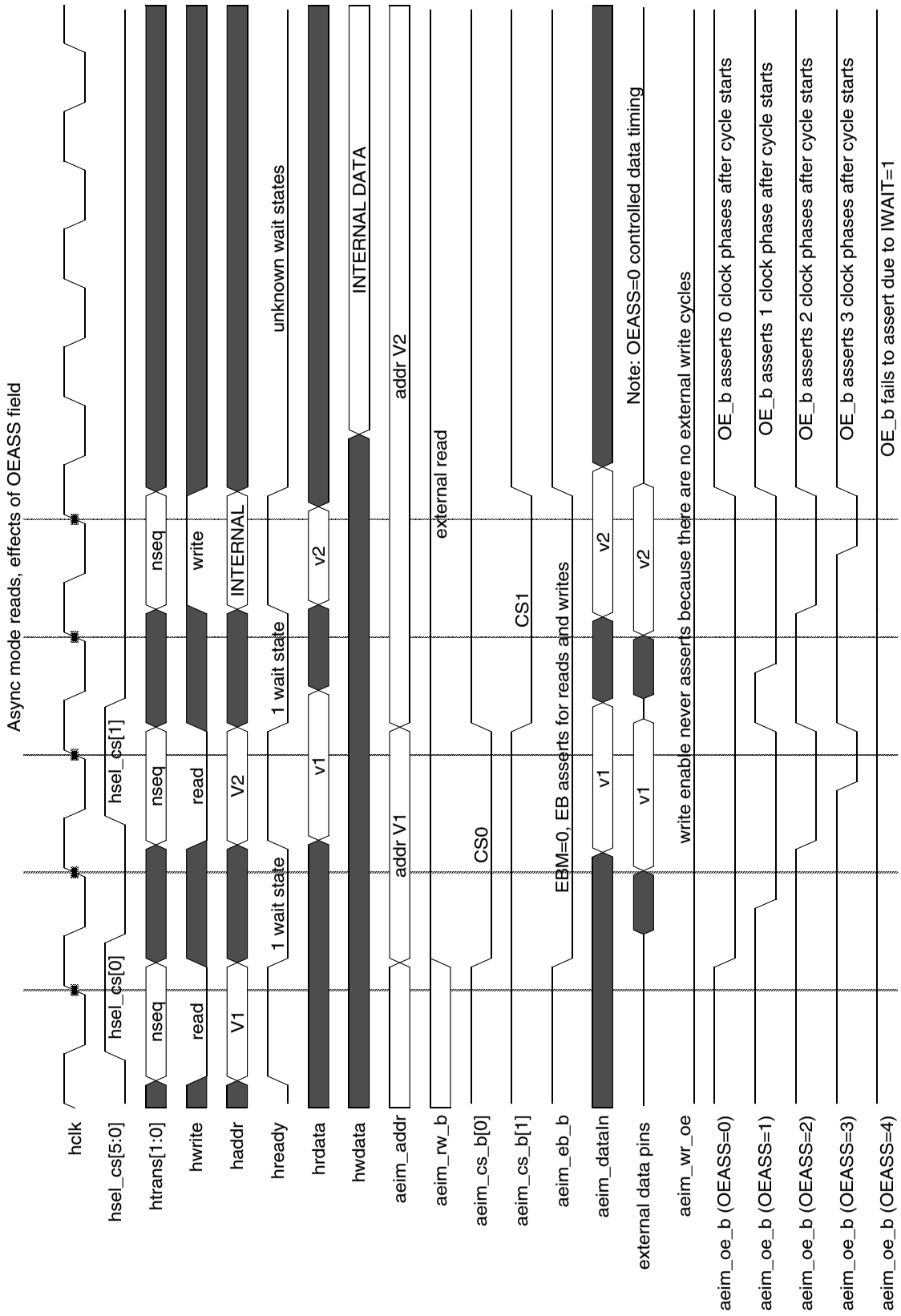


Figure 48-15. Async mode reads, effects of OEASS field

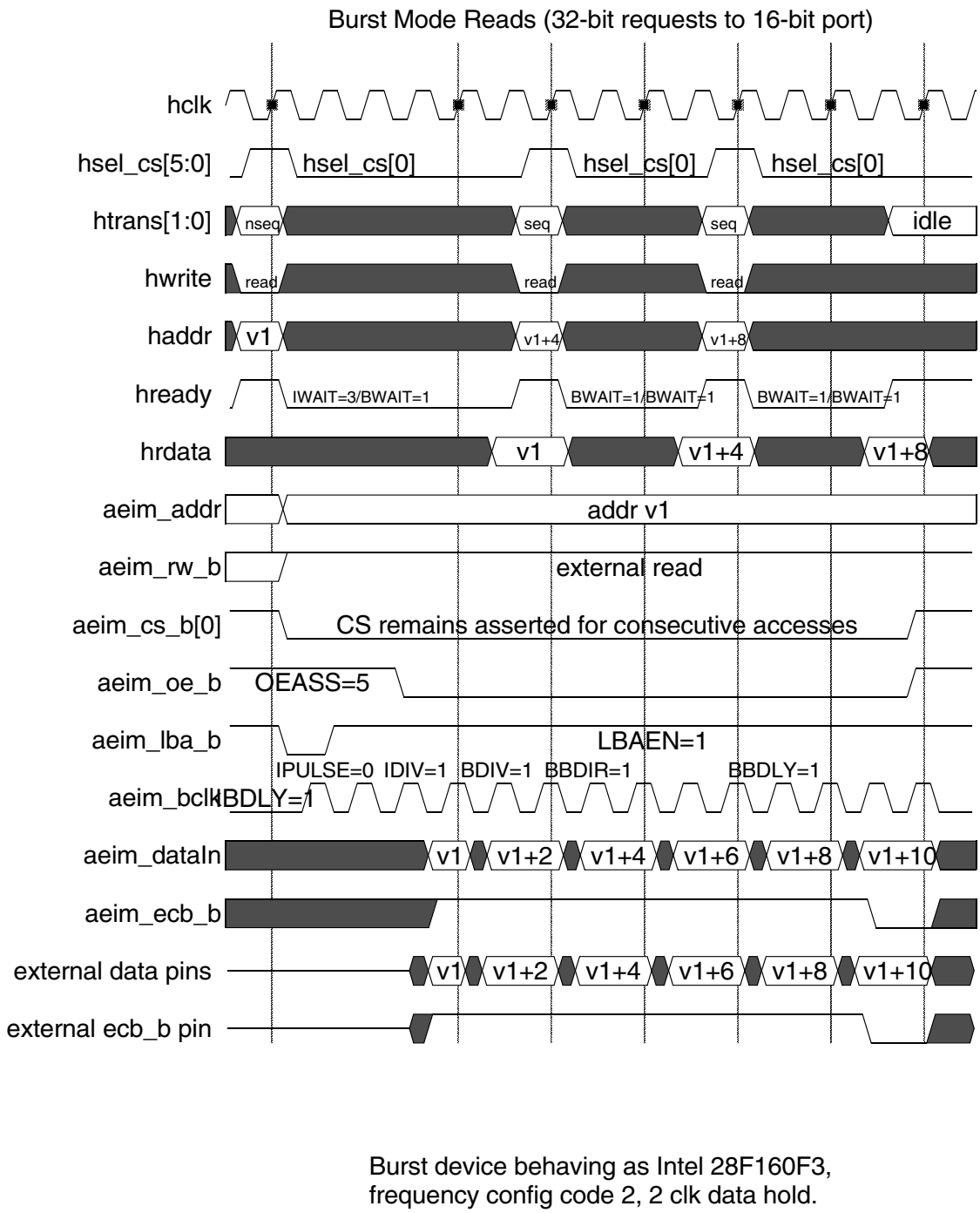
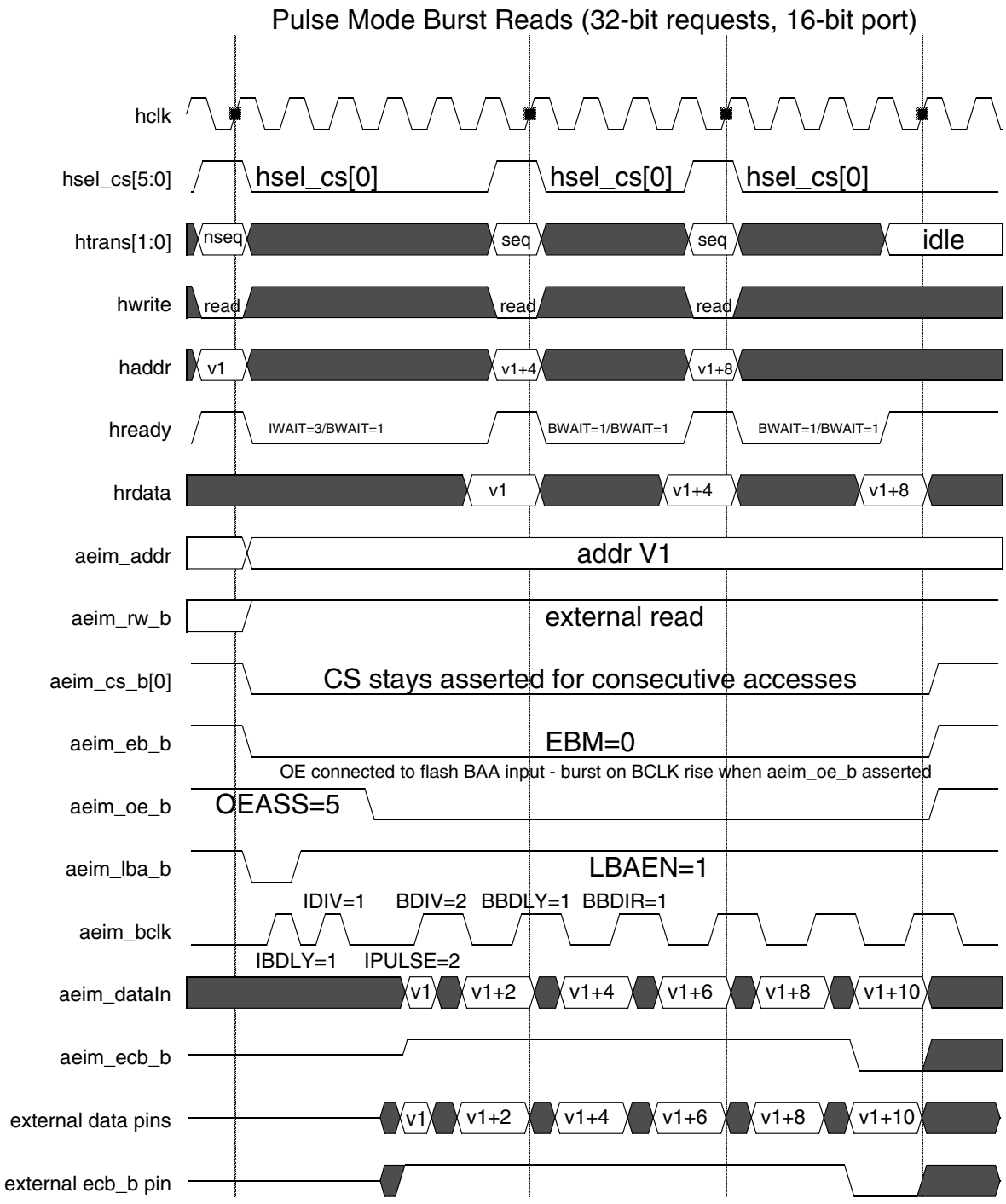


Figure 48-16. Burst Mode Reads (32-bit requests to 16-bit port)



Burst device behaving as AMD AM29BL162C,
OE tied to the flash BAA pin.

Figure 48-17. Pulse Mode Burst Reads (32-bit requests, 16-bit port)

Chapter 49
Multiple Queue Serial Peripheral Interface (MQSPI)

Revision History Table

Version	Date	Author	Changes
0.0	04/13/2002	Girjesh K. Soni	Initial version copied from Patriot INDY.
0.1	04/18/2002	Girjesh K. Soni	Modified pin diagram and pin table.PIG refrences has been removed.
0.2	05/07/03		Updated for LTE specification release.

49.1 Introduction

The MQSPI (Multiple Queue Serial Peripheral Interface) performs the serial programming operations to configure radio subsystems and selected peripherals. In this dual SPI configuration, the systems are typically partitioned into RF and baseband functions. This peripheral is designed to minimize the amount of MCU interaction necessary for multiple serial data transfers.

49.1.1 Basic Features

- Full-Duplex, Three-Wire Synchronous Transfers.
- Half-Duplex, Two-Wire Synchronous Transfers.
- Programmable Bit Rates
- Programmable Inactive Clock Polarity.
- Programmable Chip Select Polarity.
- Ten Chip Select Pins.
- 256 X 16 bit of RAM for Data I/O Storages sharable by both SPIs.
- Programmable Transfer Length - from 1 to 32 Bytes.
- Programmable Multiple Message - from 1 to 64 Messages.
- Each SPI functions only as a master SPI.

49.1.2 Extended Features

- Dual Independently Functioning SPIs.
- 32 Configurable Control Queues/Triggers
- 64 Programmable Control Data Registers (32 Mode and 32 Pointer Registers).
- Four FIFO Queues - a High and Low Priority Queue for Each SPI.
- 32 One-Cold Trigger Signals From the Layer 1 Timer.
- MCU Controlled Trigger Register.
- Two Interrupt Lines - One for Each SPI.
- Programmable Data Change on Rising/Falling Clock Edge.
- Programmable Data Latch on Rising/Falling Clock Edge.
- Separate Read/Write Data I/O Storage Pointers.
- Burst and multiple message transfers.
- Programmable 128 Clock Delay Before First Clock Edge.
- Programmable 128 Clock Delay After Transfer.
- Transmit and receive Data is LSB/MSB Selectable.
- DOZE Mode Capability.
- Serial Display Interface.

49.1.2.1 Dual Independently Functioning SPIs

The two independently functioning SPIs allow the MQSPI to program two peripherals simultaneously. The control logic for each SPI is the same. The only difference between SPI1 and SPI2 is in the RAM arbitration logic. See RAM Arbitration in Section 49.3.9, “RAM Arbitration,” for more information.

49.1.2.2 32 Configurable Control Queues

A Control Queue is made up of data for a transfer (I/O Storage Space in RAM) and a pointer to the data (Control Data Register). The Control Queues allow the MQSPI to perform serial transfers with little MCU intervention. Each Control Queue may be configured to use either of the two SPIs for data transfers.

49.1.2.3 64 Programmable Control Data Registers (32 Mode and 32 Pointer Register)

A Control Data Register is a 16 bit block of data (half-word) that resides in RAM. The Control Data Registers are located in the lower 64 addresses of RAM. The triggers stored in the FIFO queues are the addresses that access these Control Data Registers. There are two Control Data Registers per trigger. The lower address of the pair contains read and write pointer data; the next address contains transfer mode data such as qcs (Chip Select), qbl (Number of bytes in transfer.), qrc (Number of messages in transfer.) and qburst (Identifies burst mode.).

49.1.2.4 Four FIFO Queues - a High and Low Priority Queue for Each SPI

Each SPI has 2 FIFO Queues which store Control Queue triggers that are initiated by the MCU or the Layer 1 Timer. Each FIFO Queue can store up to 8 triggers. The FIFO is five bits wide to support 32 RAM address locations. This allows the MQSPI to function with a limited amount of supervision by the MCU.

49.1.2.5 32 One-Cold Trigger Signals From the Layer 1 Timer

These 32 signals are used to trigger data transfers. Each signal corresponds to a Control Queue. When a trigger is signaled, the corresponding Control Queue is executed using the appropriate Control Data and SPI. Only one trigger line may be low at a time. These triggers may be fired by programming the Layer 1 Timer to execute any of the MQSPI events.

49.1.2.6 MCU Controlled Trigger Register

The MTRIG Register is a six bit register that allows the MCU to trigger a Control Queue. The lower five bits of this register contain the trigger to be stored in the FIFO; the sixth bit is the ITG bit which when set initiates the trigger. The MCU may trigger any one of the 32 Control Queues. When a trigger is fired a Control Queue is executed as noted in Section 49.1.2.2, “32 Configurable Control Queues,” . There are Four FIFO queues, a High and Low Priority Queue for Each SPI. Only one trigger may be sent at a time.

49.1.2.7 Two Interrupt Lines

Each SPI has its own active low interrupt line. Certain maskable events listed in Table 49-14, SFLG Description may generate an interrupt. Two interrupts lines are used so that they may be prioritized at the discretion of the programmer.

49.1.2.8 Programmable Data Change/Latch Phase

The serial data in/out lines on each SPI may be configured to either change values on the rising edge of the clock and be latched on the falling edge of the clock or change values on the falling edge of the clock and be latched on the rising edge of the clock. This allows maximum flexibility with any peripheral to be added in the future. See Table 49-28, CSCFG0A Description -SPI Chip 0 Configuration A Register for more information.

49.2 PORT INFORMATION

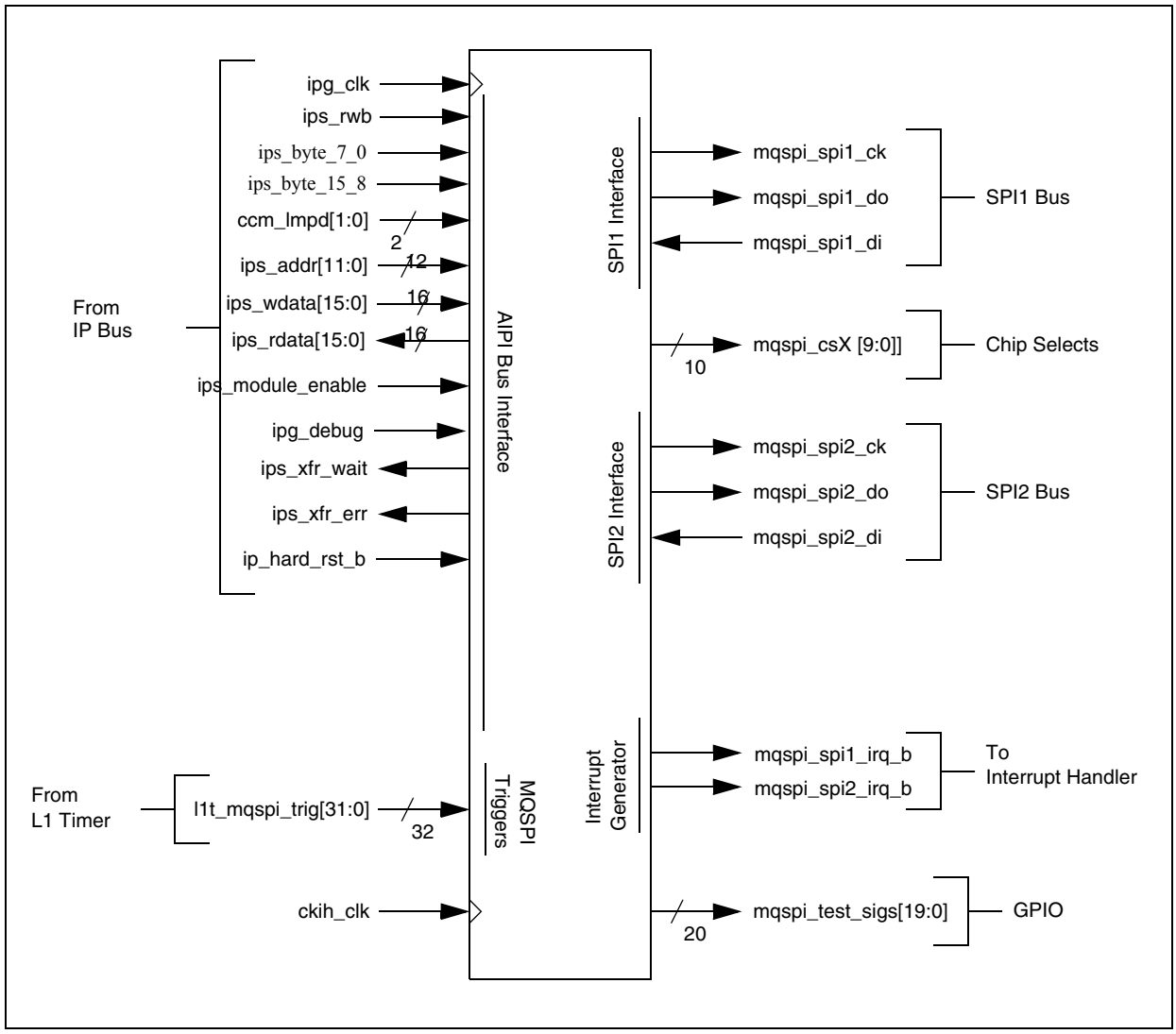


Figure 49-1. Interface Definition[1]

1. Scan related signals, such as scan_mode, test_reset, scan_en, etc. not shown on this diagram.

49.2.1 MQSPI Module Pin List

Table 49-1 is a list of the MQSPI pins.

Table 49-1. MQSPI Module Pin List

Pin Name	Direction	Description
ips_wdata	input	IP write data bus
ips_rdata	output	IP read data bus
ckih_clk	Input	PAT_REF clock input
l1t_mqspi_trig[31:0]	Input	L1 Timer Trigger inputs
ipg_clk	Input	MCU bus clock
mqspi_spi1_di	Input	SPI1 data input from chip pin
mqspi_spi2_di	Input	SPI2 data input from chip pin
ipg_debug	Input	Software Debug mode
ip_hard_rst_b	Input	System reset
ccm_lpmd[1:0]	Input	IP low power mode
ips_addr[11:0]	Input	IP address
ips_rwb	Input	IP module read/write
ips_module_enable	Input	IP module chip select
ipt_clk_se	Input	scan enable
ipt_scan_mode	Input	scan_mode
mqspi_csX [9:0]	Output	SPI chip select X output to pin. X = 0..9
mqspi_sdo_X [15:1]	Output	Scan data out X. X = 1..15
mqspi_spi1_ck	Output	SPI1 serial clock output to pin
mqspi_spi1_do	Output	SPI1 serial data output to pin
mqspi_spi1_irq_b	Output	SPI1 interrupt source
mqspi_spi2_ck	Output	SPI1 serial clock output to pin
mqspi_spi2_do	Output	SPI1 serial data output to pin
mqspi_spi2_irq_b	Output	SPI2 interrupt source
mqspi_test_sigs[19:0]	Output	Debug signals via GPIO. These are internal debug signals.
ips_xfr_wait	Output	IP transfer acknowledge
ips_xfr_err	Output	IP transfer error acknowledge

49.3 MQSPI ARCHITECTURE

The internal architecture of the MQSPI is shown in Figure on page 49-7. The MQSPI is composed of two SPI Control Blocks, a RAM Arbitration Unit, a RAM, two SPI blocks, a block of registers, four FIFO queues, a Trigger Router Block, and Peripheral Bus Interface.

49.3.1 MCU Peripheral Bus Interface

The MCU Peripheral Bus Interface provides a means of communication between the MQSPI and the MCU using an IPLAMB. .

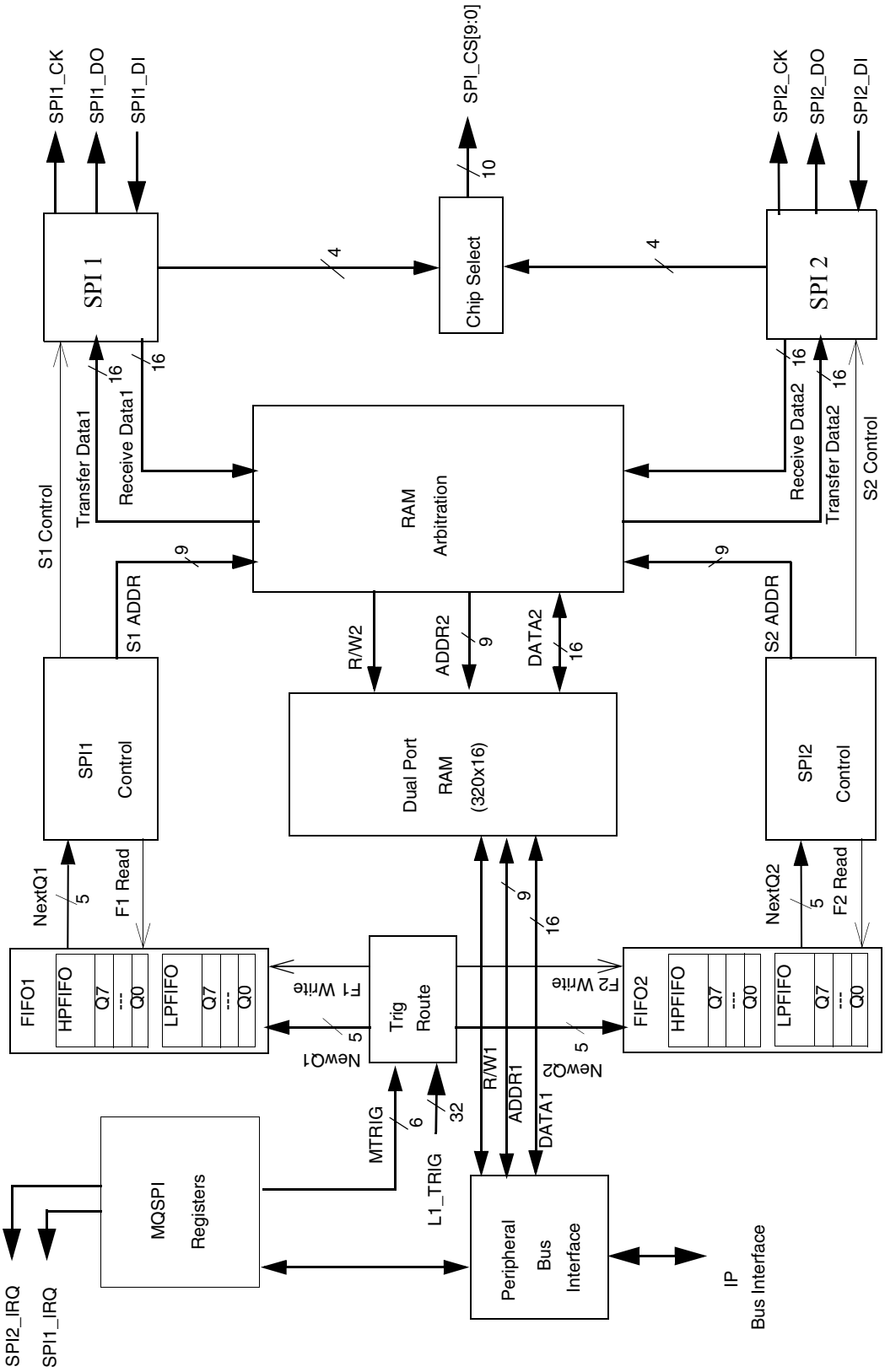


Figure 49-2. Top Level MQSPI Architecture

49.3.2 SPI1, SPI2

The SPI1 and SPI2 blocks are sequential circuits designed for the shifting out/in of serial data to/from peripherals. Each SPI contains logic for a Chip Select (CS), a Clock (CK), a data transfer operation, and a data receive operation. A block diagram of an SPI block is shown in Figure . The edges of the clock that change and latch the data are selectable for each SPI. See Table 49-13, SDEF Description - SPI Default Register and Table 49-28, CSCFG0A Description - SPI Chip Select 0 Configuration A Register as well as Figure 49-11 for more information.

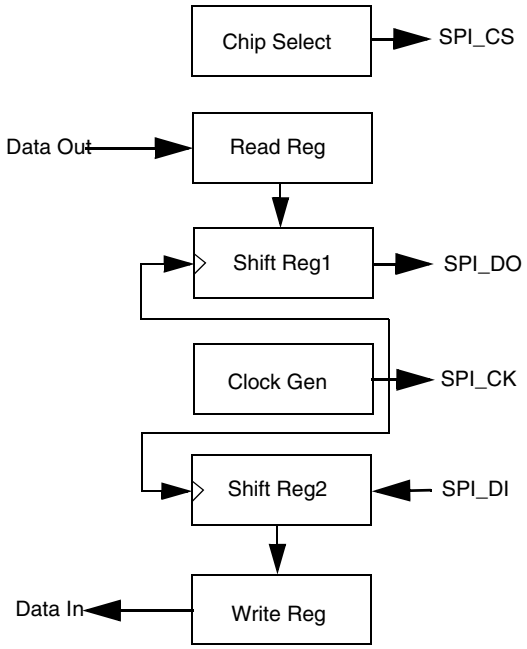


Figure 49-3. SPI Block Diagram.

49.3.3 Chip Select (CS)

The SPI_CS (Chip Select) is determined from the QCS bits in the CDMS register which is obtained from Control Data in the RAM. The QCS bits are encoded to pick 1 of 10 chip select pins on the MQSPI. The logic level of SPI_CS during a transfer is configurable.

49.3.4 Clock Gen

The Clock Gen block generates the clock signal to pin SPI_CK for the serial receive/transfer operation. The clock signal is derived from the system reference clock using 7 bit clock divider. Each chip select has seven bits labeled CKDIV in the CSCFG(0-9)A register to control the baud rate. The selectable clock rates are shown Table 49-29 in the chip select configuration register section.

49.3.4.1 Transfer Operation

Data transfers are allowed for both read and write operations to external SPI devices. Prior to initiating an SPI transfer, control and configuration registers must be set up in both the MQSPI's register space and in RAM.

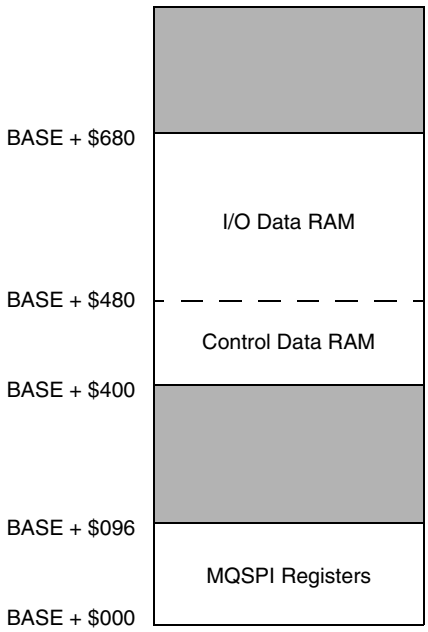


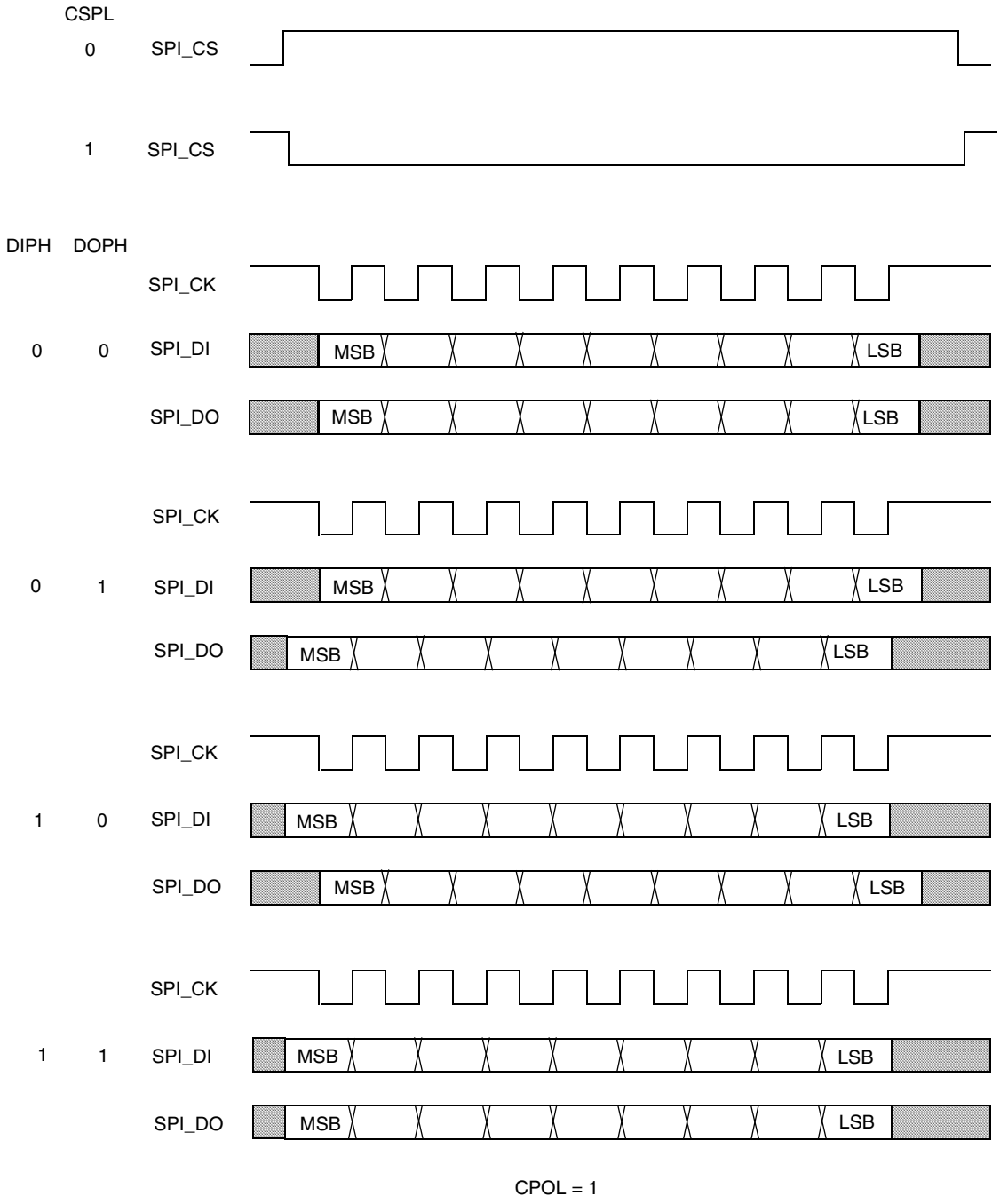
Figure 49-4. MQSPI Memory Map

49.3.4.2 SPI Transfer Setup

The following is a summary of the controls for both read and write operations which must be set up for each control queue prior to sending or receiving a message. More detailed descriptions of each register are in Section 49.3.13, “Detailed Register Descriptions,” on page 49-25.

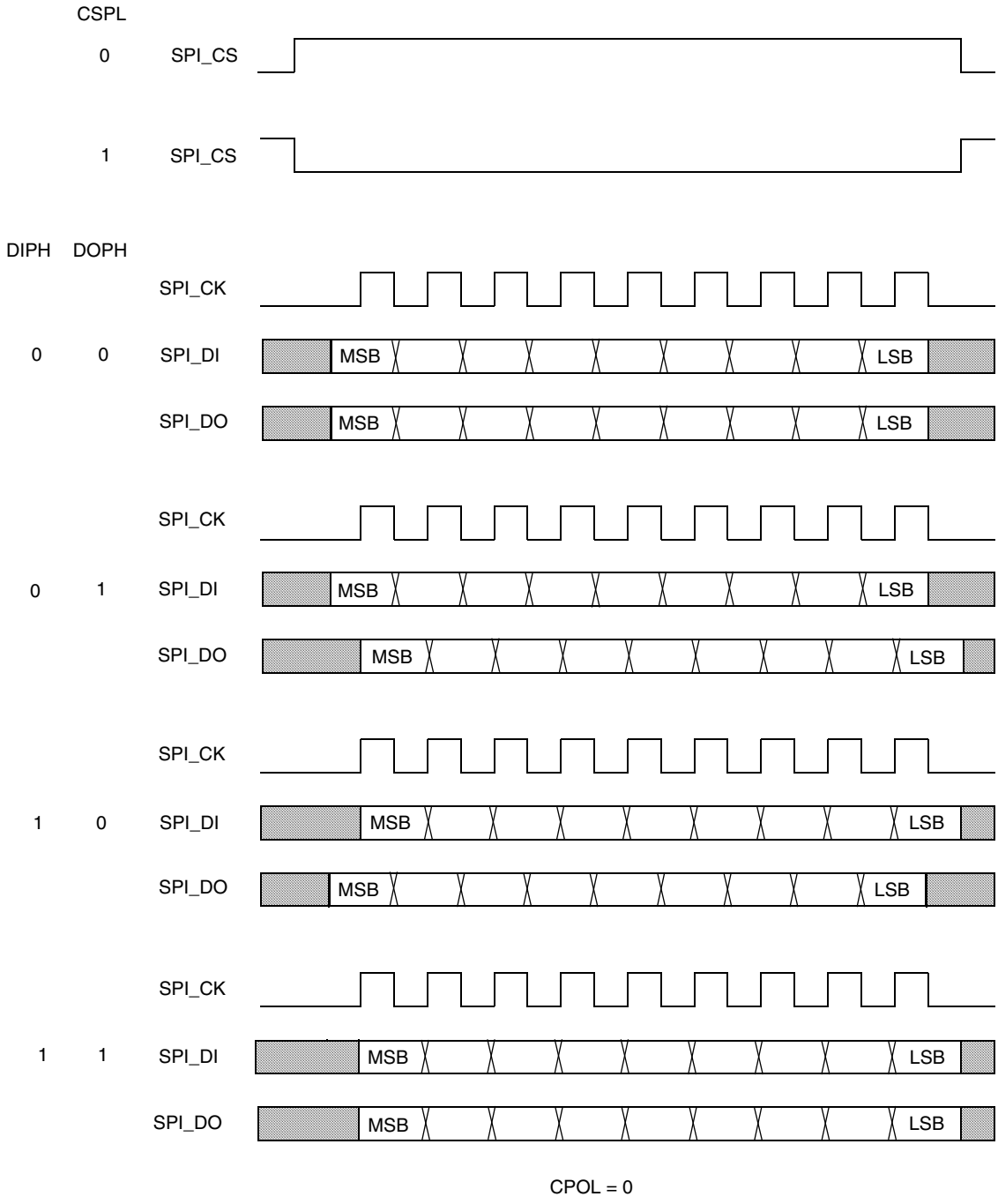
1. **SPICDP(0-31)** - Each of the 32 control queues has a Control Data Pointer Register, SPICDP(0-31), located in Control Data RAM. Each SPICDP register contains 2 address pointers into I/O Data Storage RAM:
   - QWSTR[7:0] - Start Location for Write Transfer Data. (messages to be sent to external SPI devices)
   - QRSTR[7:0] - Start Location for Read Transfer Data. (messages received from external SPI devices.)
2. **SPICDM(0-31)** - Each of the 32 control queues has a Control Data Mode Register, SPICDM, which contains additional control bits:
   - QCS[3:0] - select which chip select pin is associated with this queue
   - QBL[4:0] - Message Data Byte Length
   - QRC[5:0] - Multiple Message Repeat Count
   - QBURST - Enables Burst transfer for queues (0-3) only. Refer to Section 49.3.10 for more information on Burst mode.
3. **I/O Data RAM** - Messages to be sent to or received from external SPI devices are stored here, beginning at the addresses indicated by the SPICDP(0-31) registers.
4. **MCON** Register contains control bits to enable the MQSPI, to reset the MQSPI, and to reset Burst transfers.
5. **QCFG_L** and **QCFG_H** Registers control the assignment of each of the 32 control queues to either SPI1 or SPI2.

6. **STPR_L** and **STPR_H** Register control the priority, high or low, of each of the 32 control queues.
7. **SDEF** register controls the default setups for SPI1 and SPI2. This register controls the SPI clock polarity, and activity in low power modes.
8. **SPIM, STFE_L**, and **STFE_H** Registers determine which events may assert interrupts.
9. **QRE_L** and **QRE_H** Registers enable each queue for read operation in addition to write operations from external SPI devices.
10. **SDI_MD_L** and **SDI_MD_H** determine if the Serial Display Interface (SDI) function is enabled for each queue.
11. **CSCFG(0-9)A & CSCFG(0-9)B** - Each of the 10 chip select pins has two control registers which define configuration settings for SPI transfers specific to each chip select. The effect of the CSPL, DOPH, and DIPH bits is shown in Figure on page 49-11 and Figure on page 49-12
    - CKDIV[6:0] - Determines spi clock frequency.
    - M_L_SEL - Determines if the SPI transfer is MSB first or LSB first.
    - CSPL - Determines polarity of the chip select pin.
    - DOPH - Determines if the SPI_DO pin changes on the positive or negative edge of the spi clock.
    - DIPH - Determines if the SPI_DI pin changes on the positive or negative edge of the spi clock.
    - DAT_CNT - Programmable minimum delay between SPI transfers.
    - DBC_CNT - Programmable delay between the assertion of the chip select pin and the first spi clock edge.
12. **MTRIG** - A write to this register which sets the ITG will generate a trigger to initiate the SPI transfer associated with the selected QUEUE.



KEY: It is expected that the slave spi will latch data on SPI_DO on the edge of external SPI_CK where data is stable. Similarly, the MQSPI will latch SPI_DI on the internal edge where SPI_DI is stable as shown. For example, if DIPH=0/ DOPH=1, then slave captures SPI_DO on falling edge of external SPI_CK and MQSPI captures SPI_DI on rising edge of internal SPI_CK.

Figure 49-5. SPI timing CPOL=1



KEY: It is expected that the slave spi will latch data on SPI_DO on the edge of external SPI_CK where data is stable. Similarly, the MQSPI will latch SPI_DI on the internal edge where SPI_DI is stable as shown. For example, if DIPH=1/ DOPH=0, then slave captures SPI_DO on rising edge of external SPI_CK and MQSPI captures SPI_DI on falling edge of internal SPI_CK.

Figure 49-6. SPI timing CPOL =0

Status registers include the SPI1 Control Data Pointer Status register (CDPS1), SPI1 Control Data Mode Status register (CDMS1), SPI2 Control Data Pointer Status register (CDPS2), and SPI2 Control Data Mode Status register (CDMS2).

The CDPS1, CDMS1, CDPS2, and CDMS2 registers correspond to the Control Data Pointer and Mode registers for SPI1 and SPI2, respectively. For Control Queues configured as burst mode (only for triggers 0 through 3), these registers will indicate the BCDPS and BCDMS register contents for the active trigger. Refer to Section 49.3.10 for more information on burst mode.

49.3.4.3 SPI Transmit (Write) Operation

The SPI transfer operation to write to an external SPI device uses a 16 bit Read Register and a 16 bit PISO (Parallel-In- Serial-Out) Shift Register to shift data out. The Read Register is a buffer between the RAM and Shift Register for pre-loaded data. At the beginning of a transfer operation, a half-word of data is pre-loaded from I/O Data RAM into the Read Register. Then that half-word is transferred to the Shift Register while another half-word is transferred from RAM into the Read Register.

The SPI_CS pin is activated a programmable number of clock cycles before the first active edge of SPI_CLK. Data is transmitted either MSB or LSB first. After a half-word has been shifted out through the SPI_DO pin, the half-word in the Read Register is transferred into the Shift Register while another half-word is transferred from the RAM into the Read Register.

The operation continues until the number of bytes transferred equals the value stored in QBL + 1. After the transfer is complete and the Chip Select is de-activated an extension of the delay of the Chip Select de-active period can be programmed. The activated and de-activated values of SPI_CS depend on the polarity chosen in the CSPL bit in each of the CSCFG(0-9)A registers.

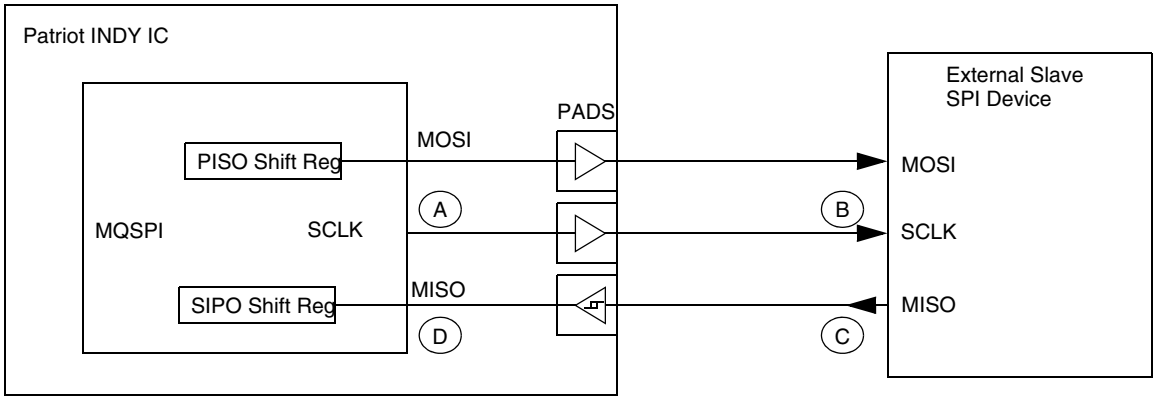
49.3.4.4 SPI Receive (Read) Operation

The receive operation to read data from an external SPI device takes place concurrently with the transfer operation if the QRE bit for that Control Queue is set in the QRE register. This operation uses a 16 bit Write Register and a 16 bit SIPO Shift Register to shift data in. The Write Register is a buffer between the Shift Register and RAM. The Shift Register serially clocks data in MSB or LSB first through the SPI_DI pin at the same rate as the transfer operation.

When a half-word has been read, it is stored in the write register where it can then be written to the location in RAM pointed to by the QRSTR register bits in the corresponding queue's SPICDP Register. The operation continues until the transfer operation ends (Bytes transferred = QBL + 1).

49.3.4.5 SPI Receive Operation Timing Considerations

The maximum operating frequency of the SPI receive operation is dependant upon the setup time for data on the MISO input pin. The required setup time is a function of several factors including the response time of the slave SPI devices, the delay through the Patriot INDY chip pads for the SCLK and MISO pins, and the rise/fall times on SCLK and MISO pins. There are some cases where these delays may prevent receive operation at some of the higher SPI clock frequency selections allowed by the setting of the CKDIV bits in the Chip Select Configuration Registers. This section explains the timing issues that must be considered by the system designer.



The timing path that limits the maximum SCLK frequency for MQSPI receive operations goes through points A, B, C, & D. This path does not affect write only transfers to the external slave SPI device.

Figure 49-7. Timing Path Limits max MQSPI Receive Frequency

There is a critical timing path, T_{dly}, (see Figure 49-7) that limits the maximum SCLK frequency that can be selected for receive operation. This timing path does not effect SPI write (transmit only) operation. This path is the summation of (1) the time required for an edge SCLK to travel from the MQSPI through the SCLK chip pad to the external slave SPI device, (2) the time required for the slave device to respond to the SCLK edge by shifting the next bit onto the MISO line, and (3) the time required for the new bit to travel through the MISO chip pad and arrive in time to be latched by the MQSPI's receive shift register.

$$T_{dly} = T_{(\text{delay through SCLK pad})} + T_{(\text{response time of Slave})} + T_{(\text{delay through MISO pad})}$$

The required relationship between T_{dly} and the SCLK frequency selection that allows the SPI receive operation to function correctly is detailed in Table 49-2. This relationship is affected by the settings of CPOL in the SPI Default Register (SDEF) and DIPH and DOPH in the Chip Select Configuration Registers. The DIPH and DOPH bit settings control which edge of SCLK that MISO and MOSI data is changed as shown in Figure on page 49-11 and Figure on page 49-12.

Table 49-2. Register Settings Affect on Maximum SCLK Frequency

Control register bit settings			Required relationship between SCLK and T_{dly}	Type
CPOL	DOPH	DIPH		
0	0	0	SCLK period / 2 > T_{dly}	1
0	0	1	N/A	3
0	1	0	SCLK period > T_{dly}	2
0	1	1	SCLK period / 2 > T_{dly}	1
1	0	0	SCLK period / 2 > T_{dly}	1
1	0	1	SCLK period > T_{dly}	2
1	1	0	N/A	3
--	1	1	SCLK period / 2 > T_{dly}	1

There are 3 types of relationships shown in Table 49-2. These are described below. Most applications will use Type 1 or Type 2 settings. Figure illustrates the timing of signals through points A, B, C, & D from Figure 49-7 on page 49-14. Figure (a) shows an example of how a receive operation works using Type 1 control register settings. Figure (b) shows the same case as Figure (a), except that the SCLK frequency is increased to the point where the setup time on the MISO pin is violated, and the receive operation fails. Figure (c) shows a solution using the SCLK frequency of Figure (b) with Type 2 control register settings to delay the latching of received data on the MISO pin by an extra 1/2 SCLK cycle, allowing for successful receive operation.

Type 1: For the cases where DIPH and DOPH are both equal to 0 or both equal to 1, then the MISO and MOSI pins should change on the same SCLK edge, and each receiving device should latch received data on the opposite SCLK edge. This allows only 1/2 an SPI clock period for the events associated with T_{dly}.

Type 2: For some higher speed SPI receive operations, the type 1 settings of Table 49-2 do not provide enough setup time after taking into account the response time of the slave SPI and pad delays. The type 2 settings allow for a full SCLK cycle for the delays associated with T_{dly}.

For example, running the GCAP IC at 13 MHz would require a T_{dly} value less than 38ns. The GCAP device has a response time of 33ns from SCLK edge to a change in MISO. Using Type 1 settings, this would leave only 5ns for the delay through the SCLK and MISO pads which is not sufficient. However when Type 2 settings are selected, the maximum T_{dly} is extended to 76 ns and timing is easily met.

Type 3: With the Type 3 settings, the receive data on MISO is latched on the same SCLK edge that causes the MOSI bit to change. This requires a negative setup time.

Figure (a): Receive operation is successful using Type 1 control register settings.

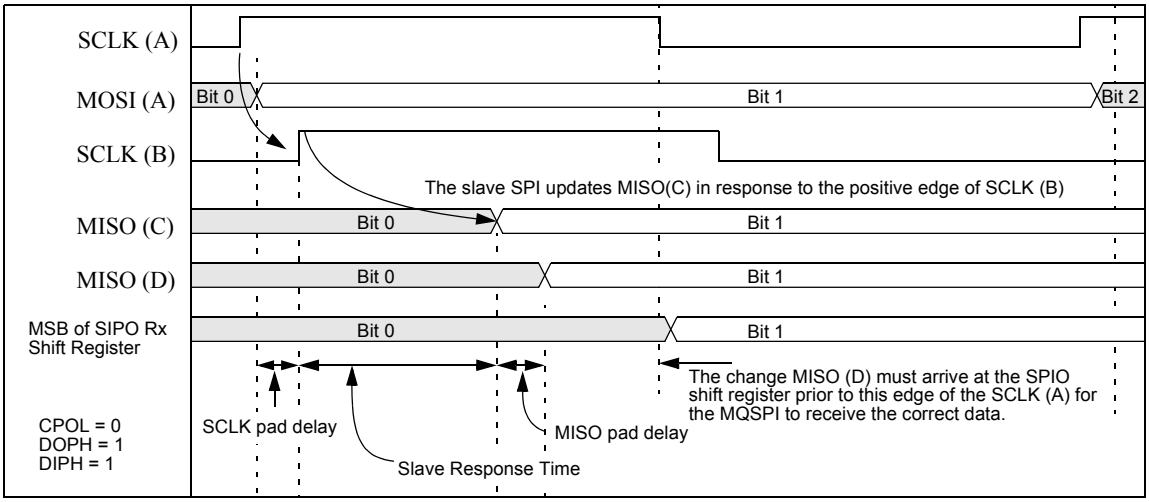


Figure (b): Receive operation fails using Type 1 control register settings with higher SCLK frequency.

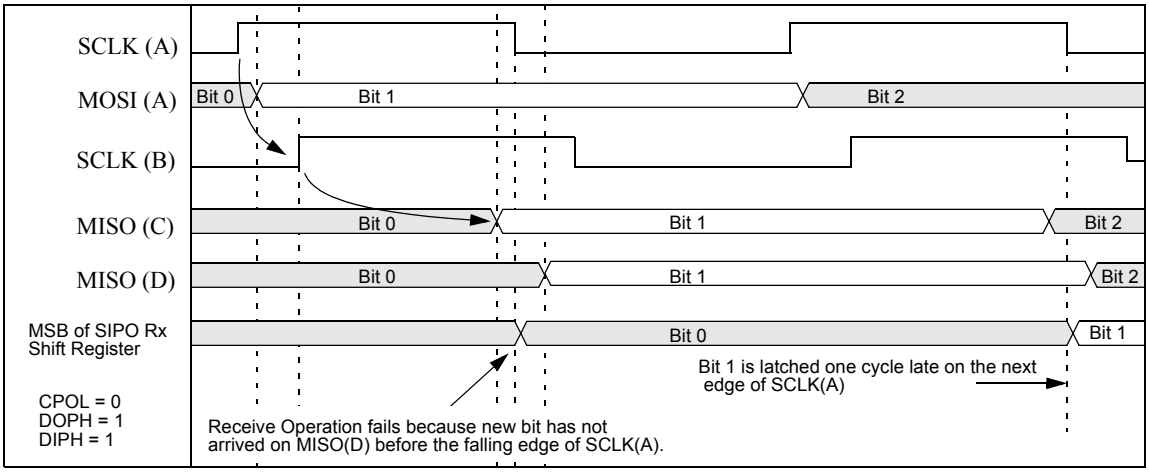


Figure (c): Receive operation successful using Type 2 control register settings with higher SCLK frequency.

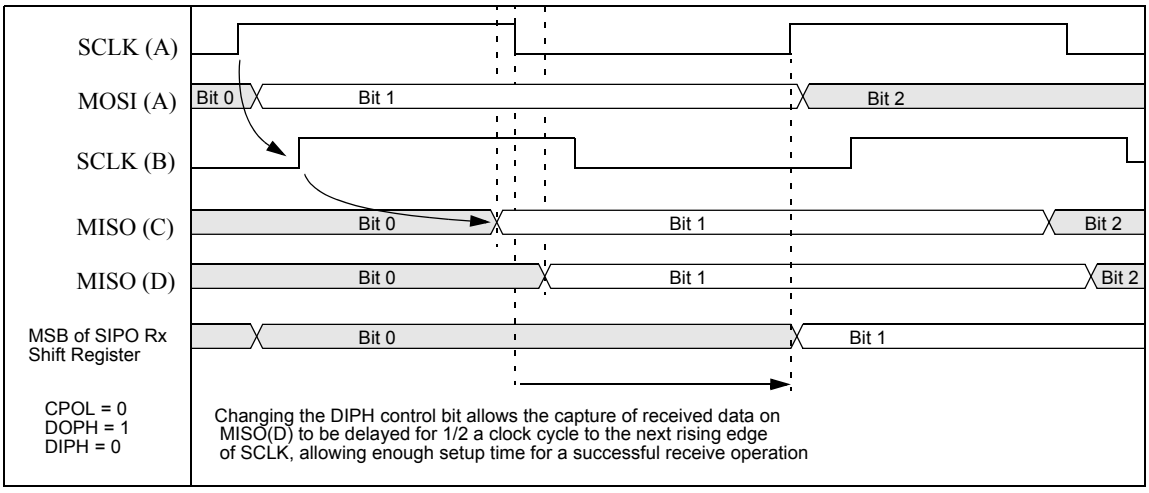


Figure 49-8. Receive Operation Timing Diagram.

49.3.5 SPI Control

The SPI Control Block contains logic to control data transfers through the MQSPI. Each SPI has a separate control block so that one may be initiated independent of the other.

49.3.6 MQSPI Registers

All registers are contained in this block. In addition, maskable interrupts are controlled by this block. The active low level interrupts can be configured to respond to the following events:

- Completion of an SPI data transfer (STF).
- Queue Overflow Error (QOE). Receiving a trigger when a FIFO queue is full.
- Receiving a trigger while an SPI is active (TSA).
- FIFO Queue is NOT FULL (QNFL).

If an SPI interrupt line is activated, either a “write 1 to clear” operation must be done or one or more bits in the SPIM register will need to be cleared to clear that interrupt. The interrupts are enabled using the SPIM register. See Table 49-15, SPIM Description - SPI Interrupt Mask Register, for more information.

There are 10 pairs of Chip Select configurable registers. CSCFG(0-9)A and CSCFG(0-9)B. This allows transfer related programmable features like Chip Select Polarity or DAT (Delay After Transfer) and DBC (Delay Before Clock) to be mapped to the chip select. See "SPI Chip Select 0 Configuration A Register," on page 49-61 and "SPI Chip Select 0 Configuration B Register," on page 49-65 for more information.

49.3.7 FIFO Queues

The MQSPI contains four FIFO queues which store up to eight Control Queue triggers each. Two FIFO queues are for SPI1 triggers. Each FIFO has a Low and a High Priority queue. The status bit APQ (Active Priority Queue) indicates whether the triggered queue is from a Low or High priority FIFO. See Sections "SPI1 Queue Status Register A," on page 49-74 and "SPI1 Queue Status Register B," on page 49-75 for more information. Triggers stored in the High Priority Queues are executed first, before any triggers in the Low Priority Queues. The other FIFO queue is for SPI2 triggers.

Each FIFO contains two pointers that keep track of the Beginning of the FIFO queue (BOQP), and the End of the FIFO queue (EOQP). These pointers can be read from the SPI1 Queue Status Register B (see Table 49-34 on page 49-75), and SPI2 Queue Status Register B (see Table 49-36 on page 49-76) registers.

49.3.8 Trig Route

The Trig Route block receives Control Queue triggers from the MCU and Layer 1 Timer and directs the trigger to the proper FIFO based on the QCFG register. If a Control Queue trigger is initiated for an SPI by the Layer 1 Timer and the MCU simultaneously, the Layer 1 Timer trigger will be handled first.

49.3.9 RAM Arbitration

The RAM Arbitration unit controls the Arbitration of the Dual Port RAM between SPI1 and SPI2. Software will control MCU acesses to the RAM to ensure that collisions are avoided. The highest priority goes to SPI1 followed by SPI2. After SPI1 or SPI2 have been granted access to the RAM, they are allowed to hold on to it for as long as they need it.

49.3.10 Burst and Multiple Message Modes.

Each of the MQSPI's 32 control queues can be configured as one of these types of messages:

- For a **Single Message**, one message is sent upon receiving the trigger.
- A **Multiple Message** is composed several single messages, and requires only one trigger to send all the messages associated with that queue.
- **Burst Transfers** are similar to multiple messages except that a separate trigger for the same queue is required for each single message associated with the burst transfer. Burst transfers are only available on control queues 0-3.

49.3.10.1 Multiple Messages

Multiple message transfers require only one trigger to send all the messages associated with the transfer. In the multiple message mode, the SPICDM register associated with each queue (Refer to Section 49-6, "SPICDM Description," on page 49-31 and Table 49-6) will set the structure of the multiple message according to the following fields:

- QCS - Defines which Chip Select is associated with this queue,
- QBL - The number of bytes in each single message of the multiple message,
- QRC - The number of messages
- QBURST must be set to '0' to indicate that the message is not a Burst transfer.

The QST1A/QST2A Registers store the SPI1 and SPI2 multiple message status information. See Table 49-33/Table 49-35. This register provides the current number of messages written (MW), the number of bytes written (BW) and the current active trigger.

The CDMS1 and CDMS2 status register display the SPICDM(0-31) value associated with the current SPI transfer. The pointer information for reads and writes to RAM for the current SPI transfer is defined in the CDPS1 and CSPS2 status registers. (Refer to the descriptions for these registers beginning on page 49-57)

49.3.10.2 Burst Transfers

Burst transfer queues are like multiple message queues, except that one trigger is required for each single message in the transfer. Burst messages can be interleaved with other Burst messages or Multiple messages. In the Burst mode; the SPICDM Register (Refer to Section "SPICDM Control Data Mode," on page 49-31) for each queue will set the initial structure of the Burst message by defining values for QCS, QBL and QRC as described in Section 49.3.10.1, and setting the QBURST field to 1. The address pointer information for reads and writes to RAM is initially defined in the SPICDP Register (Refer to Section "SPICDP Control Data Pointer," on page 49-30). Burst transfers can occur for only triggers 0,1,2 and 3. If Burst transfers are given for triggers 4 and above the transfer will default to a multiple message transfer.

The Burst Control Data Mode Registers (BCDM) and Burst Control Data Pointer Registers (BCDP) are holding register that store pointer information indicating which message in the burst transfer is next to be executed. There is one BCDP and one BCDM register for each queue on each SPI. (See BCDM and BCDP register descriptions beginning on page 49-91).

The BMA (Burst Mode Active) bits in the BMAS register indicate if a particular Burst queue is in progress. The SPI's look at these bits to determine if parameters for the transfer should be loaded into the CDMS and CDPS status registers from the SPICDM or SPICDP registers (needed for the first message of the burst transfer), or from the appropriate BCDP or BCDM register. There are four Burst Mode active bits per SPI; one for each of the Burst triggers. The BMA bit gets set after the completion of the first transfer; the QRC (message count) is monitored and when the last message is sent the BMA bit is cleared. The

BMA bits are brought out as status in the BMAS Register (See Section "Burst Mode Active Status Register," on page 49-96). Before DOZE can be activated all messages including Burst messages must be complete. Software may want to check the status of the BMA bits.

When the SPI receives a trigger which is configured for burst mode, it will look at the BMA bit in the BMAS register for this queue to determine if the burst queue is in progress. If the BMA bit is 0, then the SPI will transmit the first message of the burst queue. The CDMS and CDPS status registers are loaded with the SPICDM and SPICDP values associated with the current queue, and these values will determine the parameters of the transfer.

On the completion of the transfer of the first message in the burst queue, the BMA bit for that queue will be set in the BMAS register. The QRC value in the CDMS status register will be decremented to indicate that one message has been sent, and the new CDMS value will be stored in the BCDM register associated with that queue and SPI. The address pointers in the CDPS status register will be incremented to point to the next message in the burst queue, and the new CDPS value will be stored in the BCDP register associated with that queue and SPI. The next time that a trigger for this burst queue is received, the SPI will see that the BMA bit is 1 and will load the CDMS and CDPS registers from the appropriate BCDM and BCDP registers to define the parameters for the transfer. When the final message of the burst transfer is sent, the BMA bit will be cleared to zero.

As an example; for a two message two byte Burst transfer the initial mode and pointer information will come from the SPICDM and SPICDP Registers associated with that particular queue. After the first transfer the updated pointer and mode information is loaded into the Burst holding registers. The next time the same trigger is fired and the QBURST is set; the Burst information from the BCDP and BCDM holding registers is retrieved and the current Burst transfer is completed. If the same trigger is fired again and the QBURST is set; the mode and pointer information will come from the SPICDM and SPICDP Registers to start over from the beginning of the Burst message.

49.3.11 Serial Display Function

The Serial Display Interface (SDI) mode is dedicated to the SPI_CS0, SPI_CS1, SPI_CS2, and SPI_CS3 pins. This will allow communication to up to two serial display devices. The SDI_MD_L and SDI_MD_H registers determine if Serial Display Mode is enabled for a particular trigger. When Serial Display Mode is enabled, the SPI_CS(3-0) pins take on special SDI functions for triggers mapped to Chip Selects (3-0) in the SPICDM Register.

The SPI_CS0 and SPI_CS1 will be paired together for communication to one serial display device and SPI_CS2 and SPI_CS3 will be paired together to communicate to a second serial display device. SPI_CS0 and SPI_CS2 will be used as the chip selects and the SPI_CS1 and SPI_CS3 pins become the Serial Display Data and Control select bits.

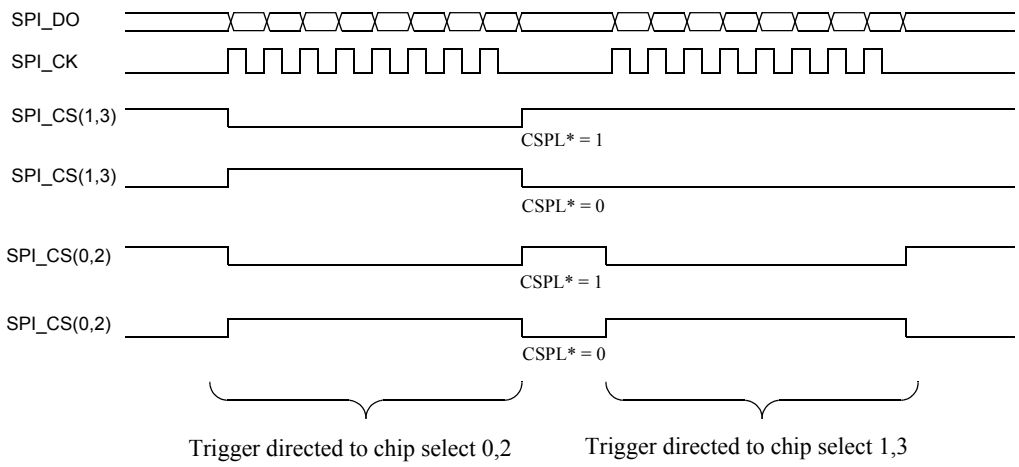
The logic value of SPI_CS1 is determined by which chip select the trigger is mapped to and the value of the CSPL (chip select polarity) bit in the CSCFG1A register. If the trigger is mapped to Chip select 0, the SDI mode bit is enabled, and the CSPL bit is clear, the SPI_CS1 pin will have a logic 1 value during the transfer. If the trigger is mapped to Chip Select 1, the SPI_CS1 pin will have a logic 0 value as shown in Figure 49-9 on page 49-20.

Similarly, if the trigger is mapped to Chip select 2, SDI mode bit is enabled, and the CSPL bit in CSCFG3A is clear, the SPI_CS3 pin will have a logic 1 value during the transfer. If the trigger is mapped to Chip Select 3, the SPI_CS3 pin will have a logic 0 value.

The Chip Select Configuration Registers A & B for each chip select determine the parameters (such as clock speed and data phase) for the SPI transfer when the corresponding chip select is indicated in the QCS bits of the SPICDM register. Therefore, it is advisable when using SDI mode that software configure the CSCFG0A, CSCFG0B, CSCFG1A and CSCFG1B registers with the same parameters for SPI_CS0 and

SPI_CS1. Similarly, the configuration registers associated with SPI_CS2 and SPI_CS3 should the same if the SDI function is used on those pins. See the Chip Select Configuration Register descriptions beginning on page 49-61 for more information.

If a Chip Select other than 0,1,2, or 3 is mistakenly selected for use when the SDI_MD_L and SDI_MD_H registers are enabled the transfers will be normal transfers (non-Serial Display). The Serial Display interface can be used in single message, multiple message or burst modes. Figure 49-9 shows the Serial Display operation on the chip select pins.



*NOTE: There is a separate CSPL control bit in the Chip Select Config. Register A, CSCFG(0-9)A, associated with each chip select pin.

Figure 49-9. Serial Display Interface (SDI) Function (Hip6w version)

To program the Serial Display Interface (SDI) function:

A) The appropriate bit in the SDI_MD_L or SDI_MD_H register needs to be set to a ‘1’. The trigger maps to the SDI_MD_L and SDI_MD_H Registers; for example SDI_MD_L(0) maps to trigger 0 and SDI_MD_H(0) maps to trigger 16.

B) The CSCFG0A/CSCFG2A bit 2 (CSPL - Chip Select Polarity; when ‘1’ the deselected Chip Select = ‘1’.) needs to be set to the desired Chip Select polarity for the SPI_CS0/SPI_CS2 pin.

C) Finally, a trigger needs to be sent that is mapped to CHIP select 0, 1, 2, or 3. The trigger is mapped based on the value stored in qcs (Chip select word) in the SPICDM(0-31) RAM control word.

49.3.12 Register Summary

This Section contains a Register Summary of the MQSPI. Section 49.3.14, “RAM,” contains a description of RAM memory. Section 49.3.14.3, “Register Descriptions,” contains a description of the registers. All addresses listed are offset from the base address of the MQSPI which is $2485_1000. They are also all listed in reference to the MCU processor, and are all 16 bit half-words starting on even addresses.

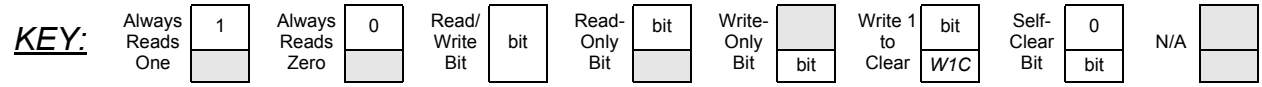


Table 49-3. MQSPI Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCON ($2485_1000)	R	0	0	0	0	0	0	BRST3	BRST2	BRST1	BRST0	MQIE	SPI1E	SPI2E	GRST	RST1	RST2
	W																
QCFG_L ($2485_1002)	R	Q15	Q14	Q13	Q12	Q11	Q10	Q9	Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
	W																
QCFG_H ($2485_1004)	R	Q31	Q30	Q29	Q28	Q27	Q26	Q25	Q24	Q23	Q22	Q21	Q20	Q19	Q18	Q17	Q16
	W																
MTRIG ($2485_1006)	R	0	0	0	0	0	0	0	0	0	0	ITG	SPTG4	SPTG3	SPTG2	SPTG1	SPTG0
	W																
STPR_L ($2485_1008)	R	STPR15	STPR14	STPR13	STPR12	STPR11	STPR10	STPR9	STPR8	STPR7	STPR6	STPR5	STPR4	STPR3	STPR2	STPR1	STPR0
	W																
STPR_H ($2485_100A)	R	STPR31	STPR30	STPR29	STPR28	STPR27	STPR26	STPR25	STPR24	STPR23	STPR22	STPR21	STPR20	STPR19	STPR18	STPR17	STPR16
	W																
SDEF ($2485_100C)	R	0	0	0	0	0	0	0	0	DBG2	DBG1	WAIT2	WAIT1	DOZE2	DOZE1	CPOL2	CPOL1
	W																
SFLG ($2485_100E)	R	HP QOE1	HP QNFL1	HP TSA1	0	LP QOE1	LP QNFL1	LP TSA1	STF1	HP QOE2	HP QNFL2	HP TSA2	0	LP QOE2	LP QNFL2	LP TSA2	STF2
	W	W1C		W1C		W1C		W1C		W1C		W1C		W1C		W1C	
SPIM ($2485_1010)	R	HP QOE1E	HP QNFL1E	HP TSA1E	0	LP QOE1E	LP QNFL1E	LP TSA1E	STF1E	HP QOE2E	HP QNFL2E	HP TSA2E	0	LP QOE2E	LP QNFL2E	LP TSA2E	STF2E
	W																
STFF_L ($2485_1012)	R	STFF15	STFF14	STFF13	STFF12	STFF11	STFF10	STFF9	STFF8	STFF7	STFF6	STFF5	STFF4	STFF3	STF F2	STFF1	STFF0
	W																
STFF_H ($2485_1014)	R	STFE31	STFE30	STFE29	STFE28	STFE27	STFE26	STFE25	STFE24	STFE23	STFE22	STFE21	STFE20	STFE19	STF E18	STFE17	STFE16
	W																
STFE_L ($2485_1016)	R	STFE15	STFE14	STFE13	STFE12	STFE11	STFE10	STFE9	STFE8	STFE7	STFE6	STFE5	STFE4	STFE3	STF E2	STFE1	STFE0
	W																
STFE_H ($2485_1018)	R	STFE31	STFE30	STFE29	STFE28	STFE27	STFE26	STFE25	STFE24	STFE23	STFE22	STFE21	STFE20	STFE19	STF E18	STFE17	STFE16
	W																
QRE_L ($2485_101A)	R	QRE15	QRE14	QRE13	QRE12	QRE11	QRE10	QRE9	QRE8	QRE7	QRE6	QRE5	QRE4	QRE3	QRE2	QRE1	QRE0
	W																
QRE_H ($2485_101C)	R	QRE31	QRE30	QRE29	QRE28	QRE27	QRE26	QRE25	QRE24	QRE23	QRE22	QRE21	QRE20	QRE19	QRE18	QRE17	QRE16
	W																
SDI_MD-L ($2485_101E)	R	SDI-MD_15	SDI-MD_14	SDI-MD_13	SDI-MD_12	SDI-MD_11	SDI-MD_10	SDI-MD_9	SDI-MD_8	SDI-MD_7	SDI-MD_6	SDI-MD_5	SDI-MD_4	SDI-MD_3	SDI-MD2	SDI-MD_1	SDI-MD_0
	W																
SDI_MD_H ($2485_1020)	R	SDI-MD_31	SDI-MD_30	SDI-MD_29	SDI-MD_28	SDI-MD_27	SDI-MD_26	SDI-MD_25	SDI-MD_24	SDI-MD_23	SDI-MD_22	SDI-MD_21	SDI-MD_20	SDI-MD_19	SDI-MD_18	SDI-MD_17	SDI-MD_16
	W																
CDPS1 ($2485_1022)	R	QWSTR7	QWSTR6	QWSTR5	QWSTR4	QWSTR3	QWSTR2	QWSTR1	QWSTR0	QRSTR7	QRSTR6	QRSTR5	QRSTR4	QRSTR3	QRSTR2	QRSTR1	QRSTR0
	W																

Table 49-3. MQSPI Register Summary (Continued)

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CDMS1 ($2485_1024)	R	QBRST	QRC5	QRC4	QRC3	QRC2	QRC1	QRC0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
	W																
CDPS2 ($2485_1026)	R	QW STR7	QW STR6	QW STR5	QW STR4	QW STR3	QW STR2	QW STR1	QW STR0	QR STR7	QR STR6	QR STR5	QR STR4	QR STR3	QR STR2	QR STR1	QR STR0
	W																
CDMS2 ($2485_1028)	R	QBRST	QRC5	QRC 4	QRC3	QRC2	QRC1	QRC0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
	W																
CSCFG0A ($2485_102A)	R	0	0	0	0	CK DIV6	CK DIV5	CK DIV4	CK DIV3	CK DIV2	CK DIV1	CK DIV0	0	M_L_ SEL	CSPL	DOPH	DIPH
	W																
CSCFG0B ($2485_102C)	R	0	0	DAT_ CNT_ 6	DAT_ CNT_ 5	DAT_ CNT_ 4	DAT_ CNT_ 3	DAT_ CNT_ 2	DAT_ CNT_1	DAT_ CNT_ 0	DBC_ CNT_ 6	DBC_ CNT_ 5	DBC_ CNT_ 4	DBC_ CNT_ 3	DBC_ CNT_ 2	DBC_ CNT_ 1	DBC_ CNT_ 0
	W																
CSCFG1A ($2485_102E)	R	0	0	0	0	CK DIV6	CK DIV5	CK DIV4	CK DIV3	CK DIV2	CK DIV1	CK DIV0	0	M_L_ SEL	CSPL	DOPH	DIPH
	W																
CSCFG1B ($2485_1030)	R	0	0	DAT_ CNT_ 6	DAT_ CNT_ 5	DAT_ CNT_ 4	DAT_ CNT_ 3	DAT_ CNT_ 2	DAT_ CNT_1	DAT_ CNT_ 0	DBC_ CNT_ 6	DBC_ CNT_ 5	DBC_ CNT_ 4	DBC_ CNT_ 3	DBC_ CNT_ 2	DBC_ CNT_ 1	DBC_ CNT_ 0
	W																
CSCFG2A ($2485_1032)	R	0	0	0	0	CK DIV6	CK DIV5	CK DIV4	CK DIV3	CK DIV2	CK DIV1	CK DIV0	0	M_L_ SEL	CSPL	DOPH	DIPH
	W																
CSCFG2B ($2485_1034)	R	0	0	DAT_ CNT_ 6	DAT_ CNT_ 5	DAT_ CNT_ 4	DAT_ CNT_ 3	DAT_ CNT_ 2	DAT_ CNT_1	DAT_ CNT_ 0	DBC_ CNT_ 6	DBC_ CNT_ 5	DBC_ CNT_ 4	DBC_ CNT_ 3	DBC_ CNT_ 2	DBC_ CNT_ 1	DBC_ CNT_ 0
	W																
CSCFG3A ($2485_1036)	R	0	0	0	0	CK DIV6	CK DIV5	CK DIV4	CK DIV3	CK DIV2	CK DIV1	CK DIV0	0	M_L_ SEL	CSPL	DOPH	DIPH
	W																
CSCFG3B ($2485_1038)	R	0	0	DAT_ CNT_ 6	DAT_ CNT_ 5	DAT_ CNT_ 4	DAT_ CNT_ 3	DAT_ CNT_ 2	DAT_ CNT_1	DAT_ CNT_ 0	DBC_ CNT_ 6	DBC_ CNT_ 5	DBC_ CNT_ 4	DBC_ CNT_ 3	DBC_ CNT_ 2	DBC_ CNT_ 1	DBC_ CNT_ 0
	W																
CSCFG4A ($2485_103A)	R	0	0	0	0	CK DIV6	CK DIV5	CK DIV4	CK DIV3	CK DIV2	CK DIV1	CK DIV0	0	M_L_ SEL	CSPL	DOPH	DIPH
	W																
CSCFG4B ($2485_103C)	R	0	0	DAT_ CNT_ 6	DAT_ CNT_ 5	DAT_ CNT_ 4	DAT_ CNT_ 3	DAT_ CNT_ 2	DAT_ CNT_1	DAT_ CNT_ 0	DBC_ CNT_ 6	DBC_ CNT_ 5	DBC_ CNT_ 4	DBC_ CNT_ 3	DBC_ CNT_ 2	DBC_ CNT_ 1	DBC_ CNT_ 0
	W																
CSCFG5A ($2485_103E)	R	0	0	0	0	CK DIV6	CK DIV5	CK DIV4	CK DIV3	CK DIV2	CK DIV1	CK DIV0	0	M_L_ SEL	CSPL	DOPH	DIPH
	W																
CSCFG5B ($2485_1040)	R	0	0	DAT_ CNT_ 6	DAT_ CNT_ 5	DAT_ CNT_ 4	DAT_ CNT_ 3	DAT_ CNT_ 2	DAT_ CNT_1	DAT_ CNT_ 0	DBC_ CNT_ 6	DBC_ CNT_ 5	DBC_ CNT_ 4	DBC_ CNT_ 3	DBC_ CNT_ 2	DBC_ CNT_ 1	DBC_ CNT_ 0
	W																
CSCFG6A ($2485_1042)	R	0	0	0	0	CK DIV 6	CK DIV 5	CK DIV 4	CK DIV 3	CK DIV 2	CK DIV 1	CK DIV 0	0	M_L_ SEL	CSPL	DOPH	DIPH
	W																
CSCFG6B ($2485_1044)	R	0	0	DAT_ CNT_ 6	DAT_ CNT_ 5	DAT_ CNT_ 4	DAT_ CNT_ 3	DAT_ CNT_ 2	DAT_ CNT_1	DAT_ CNT_ 0	DBC_ CNT_ 6	DBC_ CNT_ 5	DBC_ CNT_ 4	DBC_ CNT_ 3	DBC_ CNT_ 2	DBC_ CNT_ 1	DBC_ CNT_ 0
	W																
CSCFG7A ($2485_1046)	R	0	0	0	0	CK DIV 6	CK DIV 5	CK DIV 4	CK DIV 3	CK DIV 2	CK DIV 1	CK DIV 0	0	M_L_ SEL	CSPL	DOPH	DIPH
	W																
CSCFG7B ($2485_1048)	R	0	0	DAT_ CNT_ 6	DAT_ CNT_ 5	DAT_ CNT_ 4	DAT_ CNT_ 3	DAT_ CNT_ 2	DAT_ CNT_1	DAT_ CNT_ 0	DBC_ CNT_ 6	DBC_ CNT_ 5	DBC_ CNT_ 4	DBC_ CNT_ 3	DBC_ CNT_ 2	DBC_ CNT_ 1	DBC_ CNT_ 0
	W																
CSCFG8A ($2485_104A)	R	0	0	0	0	CK DIV6	CK DIV5	CK DIV4	CK DIV3	CK DIV2	CK DIV1	CK DIV0	0	M_L_ SEL	CSPL	DOPH	DIPH
	W																
CSCFG8B ($2485_104C)	R	0	0	DAT_ CNT_ 6	DAT_ CNT_ 5	DAT_ CNT_ 4	DAT_ CNT_ 3	DAT_ CNT_ 2	DAT_ CNT_1	DAT_ CNT_ 0	DBC_ CNT_ 6	DBC_ CNT_ 5	DBC_ CNT_ 4	DBC_ CNT_ 3	DBC_ CNT_ 2	DBC_ CNT_ 1	DBC_ CNT_ 0
	W																

Table 49-3. MQSPI Register Summary (Continued)

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSCFG9A ($2485_104E)	R	0	0	0	0	CK DIV6	CK DIV5	CK DIV4	CK DIV3	CK DIV2	CK DIV1	CK DIV0	0	M_L_ SEL	CSPL	DOPH	DIPH
	W																
CSCFG9B ($2485_1050)	R	0	0	DAT_ CNT_ 6	DAT_ CNT_ 5	DAT_ CNT_ 4	DAT_ CNT_ 3	DAT_ CNT_ 2	DAT_ CNT_1	DAT_ CNT_ 0	DBC_ CNT_ 6	DBC_ CNT_ 5	DBC_ CNT_ 4	DBC_ CNT_ 3	DBC_ CNT_ 2	DBC_ CNT_ 1	DBC_ CNT_ 0
	W																
QST1A ($2485_1052)	R	MW5	MW4	MW3	MW2	MW1	MW0	BW4	BW3	BW2	BW1	BW0	AT4	AT 3	AT2	AT1	AT0
	W																
QST1B ($2485_1054)	R	0	0	HALT _ACK	APQ	HPBO QP2	HPBO QP1	HPBO QP0	HPEO QP2	HPEO QP1	HPEO QP0	LPBO QP2	LPBO QP1	LPBO QP0	LPEO QP2	LPEO QP1	LPEO QP0
	W																
QST2A ($2485_1056)	R	MW5	MW4	MW3	MW2	MW1	MW0	BW4	BW3	BW2	BW1	BW0	AT4	AT 3	AT2	AT1	AT0
	W																
QST2B ($2485_1058)	R	0	0	HALT _ACK	APQ	HP BOQP 2	HP BOQP 1	HP BOQP 0	HP EOQP 2	HP EOQP 1	HP EOQP 0	LP BOQP 2	LP BOQP 1	LP BOQP 0	LP EOQP 2	LP EOQP 1	LP EOQP 0
	W																
SST ($2485_105A)	R	0	0	LT1 4	LT1 3	LT1 2	LT1 1	LT1 0	LT2 4	LT2 3	LT2 2	LT2 1	LT2 0	SPIB1	SPIB2	LPQ1	LPQ2
	W																
SHPFQ1A ($2485_105C)	R	0	FQ24	FQ23	FQ2 2	FQ21	FQ20	FQ14	FQ13	FQ12	FQ11	FQ10	FQ04	FQ03	FQ02	FQ01	FQ0 0
	W																
SHPFQ1B ($2485_105E)	R	0	FQ54	FQ53	FQ5 2	FQ51	FQ50	FQ44	FQ43	FQ42	FQ41	FQ40	FQ34	FQ33	FQ32	FQ31	FQ3 0
	W																
SHPFQ1C ($2485_1060)	R	0	0	0	0	0	0	FQ74	FQ73	FQ72	FQ71	FQ70	FQ6 4	FQ63	FQ62	FQ61	FQ60
	W																
SLPFQ1A ($2485_1062)	R	0	FQ24	FQ23	FQ22	FQ21	FQ20	FQ14	FQ13	FQ12	FQ11	FQ10	FQ04	FQ03	FQ02	FQ01	FQ00
	W																
SLPFQ1B ($2485_1064)	R	0	FQ54	FQ53	FQ52	FQ51	FQ50	FQ44	FQ43	FQ42	FQ41	FQ40	FQ34	FQ33	FQ32	FQ31	FQ30
	W																
SLPFQ1C ($2485_1066)	R	0	0	0	0	0	0	FQ74	FQ73	FQ72	FQ71	FQ70	FQ64	FQ63	FQ62	FQ61	FQ60
	W																
SHPFQ2A ($2485_1068)	R	0	FQ24	FQ23	FQ22	FQ21	FQ20	FQ14	FQ13	FQ12	FQ11	FQ10	FQ04	FQ03	FQ02	FQ01	FQ00
	W																
SHPFQ2B ($2485_106A)	R	0	FQ54	FQ53	FQ52	FQ51	FQ50	FQ44	FQ43	FQ42	FQ41	FQ40	FQ34	FQ33	FQ32	FQ31	FQ30
	W																
SHPFQ2C ($2485_106C)	R	0	0	0	0	0	0	FQ74	FQ73	FQ72	FQ71	FQ70	FQ64	FQ63	FQ62	FQ61	FQ60
	W																
SLPFQ2A ($2485_106E)	R	0	FQ24	FQ23	FQ22	FQ21	FQ20	FQ14	FQ13	FQ12	FQ1	FQ10	FQ04	FQ03	FQ02	FQ01	FQ00
	W																
SLPFQ2B ($2485_1070)	R	0	FQ54	FQ53	FQ5 2	FQ51	FQ50	FQ44	FQ43	FQ42	FQ41	FQ40	FQ34	FQ33	FQ32	FQ31	FQ30
	W																
SLPFQ2C ($2485_1072)	R	0	0	0	0	0	0	FQ74	FQ73	FQ72	FQ71	FQ70	FQ64	FQ63	FQ62	FQ61	FQ60
	W																
BCDP0S1 ($2485_1074)	R	QW STR7	QW STR6	QW STR5	QW STR4	QW STR3	QW STR2	QW STR1	QW STR0	QR STR7	QR STR6	QR STR5	QR STR4	QR STR3	QR STR2	QR STR1	QR STR0
	W																
BCDM0S1 ($2485_1076)	R	QBRS T	QRC5	QRC 4	QRC3	QRC2	QRC1	QRC0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
	W																
BCDP1S1 ($2485_1078)	R	QW STR7	QW STR6	QW STR5	QW STR4	QW STR3	QW STR2	QW STR1	QW STR0	QR STR7	QR STR6	QR STR5	QR STR4	QR STR3	QR STR2	QR STR1	QR STR0
	W																

Table 49-3. MQSPI Register Summary (Continued)

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCDM1S1 ($2485_107A)	R	QBURST	QRC5	QRC 4	QRC3	QRC2	QRC1	QRC0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
	W																
BCDP2S1 ($2485_107C)	R	QW STR7	QW STR6	QW STR5	QW STR4	QW STR3	QW STR2	QW STR1	QW STR0	QR STR7	QR STR6	QR STR5	QR STR4	QR STR3	QR STR2	QR STR1	QR STR0
	W																
BCDM2S1 ($2485_107E)	R	QBURST	QRC5	QRC 4	QRC3	QRC2	QRC1	QRC0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
	W																
BCDP3S1 ($2485_1080)	R	QW STR7	QW STR6	QW STR5	QW STR4	QW STR3	QW STR2	QW STR1	QW STR0	QR STR7	QR STR6	QR STR5	QR STR4	QR STR3	QR STR2	QR STR1	QR STR0
	W																
BCDM3S1 ($2485_1082)	R	QBURST	QRC5	QRC 4	QRC3	QRC2	QRC1	QRC0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
	W																
BCDP0S2 ($2485_1084)	R	QW STR7	QW STR6	QW STR5	QW STR4	QW STR3	QW STR2	QW STR1	QW STR0	QR STR7	QR STR6	QR STR5	QR STR4	QR STR3	QR STR2	QR STR1	QR STR0
	W																
BCDM0S2 ($2485_1086)	R	QBURST	QRC 5	QRC 4	QRC 3	QRC 2	QRC 1	QRC 0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
	W																
BCDP1S2 ($2485_1088)	R	QW STR7	QW STR6	QW STR5	QW STR4	QW STR3	QW STR2	QW STR1	QW STR0	QR STR7	QR STR6	QR STR5	QR STR4	QR STR3	QR STR2	QR STR1	QR STR0
	W																
BCDM1S2 ($2485_108A)	R	QBURST	QRC5	QRC4	QRC3	QRC2	QRC1	QRC0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
	W																
BCD1P2S2 ($2485_108C)	R	QW STR7	QW STR6	QW STR5	QW STR4	QW STR3	QW STR2	QW STR1	QW STR0	QR STR7	QR STR6	QR STR5	QR STR4	QR STR3	QR STR2	QR STR1	QR STR0
	W																
BCDM2S2 ($2485_108E)	R	QBURST	QRC5	QRC4	QRC3	QRC2	QRC1	QRC0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
	W																
BCDP3S2 ($2485_1090)	R	QW STR7	QW STR6	QW STR5	QW STR4	QW STR3	QW STR2	QW STR1	QW STR0	QR STR7	QR STR6	QR STR5	QR STR4	QR STR3	QR STR2	QR STR1	QR STR0
	W																
BCDM3S2 ($2485_1092)	R	QBURST	QRC5	QRC4	QRC3	QRC2	QRC1	QRC 0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
	W																
BMAS ($2485_1094)	R	0	0	0	0	0	0	0	0	BMA_2_0	BMA_2_1	BMA_2_2	BMA_2_3	BMA_1_0	BMA_1_1	BMA_1_2	BMA_1_3
	W																
(2485_1096-2485_13FE)	R	Reserved															
	W																
SPICDP ($2485_1400+4*n)	R	QW STR7	QW STR6	QW STR5	QW STR4	QW STR3	QW STR2	QW STR1	QW STR0	QR STR7	QR STR6	QR STR5	QR STR4	QR STR3	QR STR2	QR STR1	QR STR0
	W																
SPICDM ($2485_1402+4*n)	R	QBRST	QRC5	QRC4	QRC3	QRC2	QRC1	QRC0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
	W																

Table 49-3. MQSPI Register Summary (Continued)

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I/O Dara RAM (2485_1480-2485_167E)	R	I/O Data RAM															
	W																
*Note: n = (0-31)																	

49.3.13 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various GPIO registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

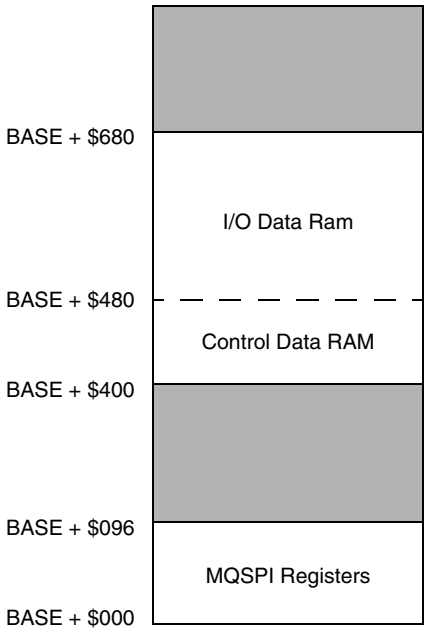


Figure 49-10. MQSPI Memory Map

49.3.14 RAM

The RAM contains 320 locations of half-words that store all of the data for an SPI transfer/receive operation. It is divided into two parts: 256 locations for I/O Data Space, and 64 locations for Control Data Registers. See Table 49-4 for a description of control RAM contents.

Table 49-4. MQSPI RAM Memory Map

Location	Description
$2485_1400	Control Data Pointer Register 0 (SPICDP0)
$2485_1402	Control Data Mode Register 0 (SPICDM0)
$2485_1404	Control Data Pointer Register 1 (SPICDP1)
$2485_1406	Control Data Mode Register 1 (SPICDM1)
$2485_1408	Control Data Pointer Register 2 (SPICDP2)
$2485_140A	Control Data Mode Register 2 (SPICDM2)
$2485_140C	Control Data Pointer Register 3 (SPICDP3)
$2485_140E	Control Data Mode Register 3 (SPICDM3)
$2485_1410	Control Data Pointer Register 4 (SPICDP4)
$2485_1412	Control Data Mode Register 4 (SPICDM4)
$2485_1414	Control Data Pointer Register 5 (SPICDP5)

Table 49-4. MQSPI RAM Memory Map (Continued)

Location	Description
$2485_1416	Control Data Mode Register 5 (SPICDM5)
$2485_1418	Control Data Pointer Register 6 (SPICDP6)
$2485_141A	Control Data Mode Register 6 (SPICDM6)
$2485_141C	Control Data Pointer Register 7 (SPICDP7)
$2485_141E	Control Data Mode Register 7 (SPICDM7)
$2485_1420	Control Data Pointer Register 8 (SPICDP8)
$2485_1422	Control Data Mode Register 8 (SPICDM8)
$2485_1424	Control Data Pointer Register 9 (SPICDP9)
$2485_1426	Control Data Mode Register 9 (SPICDM9)
$2485_1428	Control Data Pointer Register 10 (SPICDP10)
$2485_142A	Control Data Mode Register 10 (SPICDM10)
$2485_142C	Control Data Pointer Register 11 (SPICDP11)
$2485_142E	Control Data Mode Register 11 (SPICDM11)
$2485_1430	Control Data Pointer Register 12 (SPICDP12)
$2485_1432	Control Data Mode Register 12 (SPICDM12)
$2485_1434	Control Data Pointer Register 13 (SPICDP13)
$2485_1436	Control Data Mode Register 13 (SPICDM13)
$2485_1438	Control Data Pointer Register 14 (SPICDP14)
$2485_143A	Control Data Mode Register 14 (SPICDM14)
$2485_143C	Control Data Pointer Register 15 (SPICDP15)
$2485_143E	Control Data Mode Register 15 (SPICDM15)
$2485_1440	Control Data Pointer Register 16 (SPICDP16)
$2485_1442	Control Data Mode Register 16 (SPICDM16)
$2485_1444	Control Data Pointer Register 17 (SPICDP17)
$2485_1446	Control Data Mode Register 17 (SPICDM17)
$2485_1448	Control Data Pointer Register 18 (SPICDP18)
$2485_144A	Control Data Mode Register 18 (SPICDM18)
$2485_144C	Control Data Pointer Register 19 (SPICDP19)
$2485_144E	Control Data Mode Register 19 (SPICDM19)
$2485_1450	Control Data Pointer Register 20 (SPICDP20)

Table 49-4. MQSPI RAM Memory Map (Continued)

Location	Description
$2485_1452	Control Data Mode Register 20 (SPICDM20)
$2485_1454	Control Data Pointer Register 21 (SPICDP21)
$2485_1456	Control Data Mode Register 21 (SPICDM21)
$2485_1458	Control Data Pointer Register 22 (SPICDP22)
$2485_145A	Control Data Mode Register 22 (SPICDM22)
$2485_145C	Control Data Pointer Register 23 (SPICDP23)
$2485_145E	Control Data Mode Register 23 (SPICDM23)
$2485_1460	Control Data Pointer Register 24 (SPICDP24)
$2485_1462	Control Data Mode Register 24 (SPICDM24)
$2485_1464	Control Data Pointer Register 25 (SPICDP25)
$2485_1466	Control Data Mode Register 25 (SPICDM25)
$2485_1468	Control Data Pointer Register 26 (SPICDP26)
$2485_146A	Control Data Mode Register 26 (SPICDM26)
$2485_146C	Control Data Pointer Register 27 (SPICDP27)
$2485_146E	Control Data Mode Register 27 (SPICDM27)
$2485_1470	Control Data Pointer Register 28 (SPICDP28)
$2485_1472	Control Data Mode Register 28 (SPICDM28)
$2485_1474	Control Data Pointer Register 29 (SPICDP29)
$2485_1476	Control Data Mode Register 29 (SPICDM29)
$2485_1478	Control Data Pointer Register 30 (SPICDP30)
$2485_147A	Control Data Mode Register 30 (SPICDM30)
$2485_147C	Control Data Pointer Register 31 (SPICDP31)
$2485_147E	Control Data Mode Register 31 (SPICDM31)
$2485_1480 - $2485_167E	I/O Data Space

The RAM data will be shared between SPI1 and SPI2. However, SPI data transfers will be done independently of each other. This allows the MCU to customize SPI queue memory needs for each of the SPI interfaces.

The RAM is intended to be partitioned into active frame data and next frame data, although no logic exists to separate the two. The MCU should program the MQSPI RAM in the frame prior to when the transfers take place.

49.3.14.1 I/O Data Storage RAM

The Data I/O Storage RAM (MCU Address, Hex, $2485_1480 - $2485_167E) is used to hold the data to be transferred or read through one of the SPI ports. The MQSPI has the capability to send various length data transfers. For this reason, the Data I/O Storage RAM is 256 X 16.

The data to be stored in the I/O Data Space is the actual data that will be transferred through the SPI. Data will be transferred in integer units of bytes. Separate read and write pointers allow Read Data from the SPI Bus to not overwrite the Write Data. Packing of data is not allowed, where an odd byte transfer will always waste 1 byte of storage. Even though data is transferred in units of bytes, data must be stored in the I/O Data Space in units of half-words.

49.3.14.2 Control Data RAM

The MQSPI has 32 Control Queues for preconfiguring up to 32 transfers. There are two Control Data RAM words for each MQSPI Control Queue for a total of 64 words. One of the words is Pointer Control which is for specifying the read and write offset pointers for the transfer data within the Data I/O Storage RAM. Separate read and write data pointers allow the read data that is read from the SPI bus to not over-write the write data. The second word is Mode Control which is for specifying burst modes, multiple message modes, message lengths, and chip selects.

The Control Data RAM is organized such that Control Queue 0's Control Data Pointer register is located at address $400 and its Control Data Mode register is located at address $402. This repeats going up in memory for each consecutive Control Queue, with Control Queue 1 starting at address $404. Since the Control Data is in RAM, there is no defined reset value.

Each of the 32 control queues has a Control Data Pointer Register associated with it as listed in Table 49-4 on page 49-26. This register contains address pointers to I/O Data Storage RAM for the start address of the messages associated with each queue.

SPICDP(0-31) SPICDP Control Data Pointer **Addr** $2485_1**400**+4*n

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QW STR7	QW STR6	QW STR5	QW STR4	QW STR3	QW STR2	QW STR1	QW STR0	QR STR7	QR STR6	QR STR5	QR STR4	QR STR3	QR STR2	QR STR1	QR STR0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

***Note**: n = (0-31)

Table 49-5. SPICDP Description

Name	Description	Settings
QWSTR[7:0] Bits 15-8	**Starting Location of Write Transfer Data —** QWSTR contains the starting location in RAM for the data to be written out of the SPI. Note: Because the internal RAM address differs from the external address, an adjustment must be made to the address before it is stored in QWSTR. The adjustment is: QWSTR = (DStart - $480)/2, where: DStart = Offset Starting address of data to be transferred (written) out of the SPI, $480 = Offset Address of the beginning of the SPI I/O Data RAM.	QWSTR = (DStart - $480)/2
QRSTR[7:0] Bits 7-0	**Starting Location of Read Transfer Data —** QRSTR contains the starting location in RAM for the data to be read to be stored from the SPI. Note: Because the internal RAM address differs from the external address, an adjustment must be made to the address before it is stored in QRSTR. The adjustment is: QRSTR = (DStart - $480)/2, where: DStart = Offset Starting address of data to be stored from the SPI read transfer, $480 = Offset Address of the beginning of the SPI I/O Data RAM.	QRSTR = (DStart - $480)/2

Each of the 32 control queues has a Control Data Mode Register associated with it as listed in Table 49-4 on page 49-26.

SPICDM(0-31) SPICDM Control Data Mode **Addr** $2485_1402+4*n

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QBRST	QRC5	QRC4	QRC3	QRC2	QRC1	QRC0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

***Note**: n = (0-31)

Table 49-6. SPICDM Description

Name	Description	Settings
QBRST Bit 15	**Queue Burst Mode Enable—** The QBRST (Available only for Control Queues 0 through 3) selects burst mode operation when multiple message mode is used (that is, QRC[5:0] is non-zero). When burst mode is enabled and more than one message is to be sent (QRC[5:0] > 0), every corresponding trigger received, either from the L1 Timer or using the MTRIG register, will result in only 1 of the multiple messages to be sent. The first trigger received will send the first message, a second trigger will send the second message, etc. The contents of the Mode Control and Pointer Control data are temporarily stored in holding registers (registers BCDPxSy, and BCDMxSy, where BCDPxSy is the Pointer Control and BCDMxSy is the Mode Control, x is triggers 0, 1, 2, or 3, and y is SPI1 or SPI2). After all messages have been sent, the message sequence will repeat by re-reading the Control data RAM provided that there are any more triggers and the QBRST bit is still set.	0 = Burst mode disabled 1 = Burst mode enabled
QRC[5:0] Bits 14-9	Queue Multiple Message Repeat Count— QRC specifies the number of messages to be transferred. The SPI chip select goes inactive momentarily (minimum 3.5 spi clock cycles) between each message sent. This allows multiple SPI messages to be sent consecutively using only a single trigger. If burst mode is enabled (QBRST bit is set), then a trigger must be received for each message.	000000 = 1 message will be transferred … 111111 = 64 messages will be transferred
QBL[4:0] Bits 8-4	Queue Message Data Byte Length— QBL specifies the number of whole bytes to be transferred per message. When multiple messages are used (QRC[5:0] > 0), QBL specifies only the number of a single message and not all of the messages.	00000 = 1 byte message … 11111 = 32 byte message

Table 49-6. SPICDM Description (Continued)

Name	Description	Settings
QCS[3:0] Bits 3-0	Queue Chip Select— QCS chooses which chip select pin to use for the SPI transfer.	0000 = SPI_CS[0] 0001 = SPI_CS[1] 0010 = SPI_CS[2] 0011 = SPI_CS[3] 0100 = SPI_CS[4] 0101 = SPI_CS[5] 0110 = SPI_CS[6] 0111 = SPI_CS[7] 1000 = SPI_CS[8] 1001 = SPI-CS[9]

.

49.3.14.3 Register Descriptions

The MCON register contains information to enable and reset the MQSPI.

MCON MQSPI Control Register **Addr** $2485_1**000**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
							BRST 3	BRST 2	BRST 1	BRST 0	MQIE	SPI1E	SPI2E	GRST	RST1	RST2
TYPE	r	r	r	r	r	r	rwm	rwm	rwm	rwm	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0

Table 49-7. MCON Description

Name	Description	Settings
Bits 15-10	Reserved	N/A
BRST[3:0] Bits 9-6	**Burst Reset for Control Queues 0, 1, 2, and 3—** Setting any of the BRST bits has the effect of terminating (resetting) the respective Control Queue that is currently operating in the burst mode. The bits are automatically cleared at the completion of the reset sequence. If a BRST bit is set while as SPI transfer for that queue is in progress, the reset will not take affect until the completion of the current transfer	0 = Normal Mode 1 = Initiate a reset of Control Queue in burst mode
MQIE Bit 5	**MQSPI Enable—** This bit is used to disable the MQSPI to reduce power consumption when the MQSPI is not used for extended periods of time. MQIE disables the MQSPI by gating off the clock input to the MQSPI. This bit must be cleared before the SPI1E and SPI2E bits have any affect. Clearing the MQIE, SPI1E, and SPI2E bits can be done simultaneously with a single write into MCON register.	0 = Enable MQSPI 1 = Disable MQSPI
SPI1E Bit 4	**SPI1 Enable—**This bit is used to disable only SPI1 to reduce power consumption when SPI1 is not used for extended periods of time. SPI1E, when set to ‘1’, disables SPI1 by gating off the clock input to SPI1. MQIE must be cleared before this bit has any affect.	0 = Enable SPI1 1 = Disable SPI1
SPI2E Bit 3	**SPI2 Enable—** This bit is used to disable only SPI2 to reduce power consumption when SPI2 is not used for extended periods of time. SPI2E, when set to ‘1’, disables SPI2 by gating off the clock input to SPI2. MQIE must be cleared before this bit has any affect.	0 = Enable SPI2 1 = Disable SPI2

Table 49-7. MCON Description (Continued)

Name	Description	Settings
GRST Bit 2	**Global MQSPI Reset—** Asserting GRST begins a global reset sequence of the MQSPI. At the completion of the reset sequence, GRST is automatically cleared. A reset takes two clock cycles.	0 = Normal Mode 1 = Initiate a Global Reset
RST1 Bit 1	**SPI1 Reset— ** Asserting RST1 begins a reset sequence of SPI1. At the completion of the reset sequence, RST1 is automatically cleared. This bit resets all SPI1 functions except the registers. A reset takes two clock cycles.	0 = Normal Mode 1 = Initiate a reset of SPI1
RST2 Bit 0	SPI2 Reset— Asserting RST2 begins a reset sequence of SPI2. At the completion of the reset sequence, RST2 is automatically cleared. This bit resets all SPI2 functions except the registers. A reset takes two clock cycles.	0 = Normal Mode 1 = Initiate a reset of SPI2

QCFG_L Control Queue Configuration Low Register **Addr** $2485_1**002**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Q15	Q14	Q13	Q12	Q11	Q10	Q9	Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-8. QCFG_L Description

Name	Description	Settings
Q[15:0] Bits 15-0	**SPI Queue Assignment for triggers 15 through 0—** The QCFG_L register determines to which SPI each Control Queue will be connected for triggers 15 through 0. This register would be used in conjunction with the QCFG_H register.	0 = When triggered, Control Queue will be executed using SPI1 1 = When triggered, Control Queue will be executed using SPI2

QCFG_H Control Queue Configuration High Register **Addr** $2485_1**004**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Q31	Q30	Q29	Q28	Q27	Q26	Q25	Q24	Q23	Q22	Q21	Q20	19	Q18	Q17	Q16
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-9. QCFG_H Description

Name	Description	Settings
Q[31:16] Bits 15-0	**SPI Queue Assignment for triggers 31 through 16—** The QCFG_H register determines to which SPI each Control Queue will be connected for triggers 31 through 16. This register would be used in conjunction with the QCFG_L register.	0 = When triggered, Control Queue will be executed using SPI1 1 = When triggered, Control Queue will be executed using SPI2

The MTRIG register is used to trigger a Control Queue. To initiate a trigger, a 5 digit binary number corresponding to the Control Queue is written to the SPTG[4:0] bits. Setting the ITG bit will initiate the trigger contained in SPTG[4:0]. Both ITG and SPTG[4:0] may be set with the same write access. This trigger will then be sent to a FIFO queue before executing the Control Queue on one of the SPIs.

MTRIG MCU Trigger Register **Addr** $2485_1**006**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
											ITG	SPTG 4	SPTG 3	SPTG 2	SPTG 1	SPTG 0
TYPE	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-10. MTRIG Description

Name	Description	Settings
Bits 15-6	Reserved	N/A
ITG Bit 5	**Initiate Trigger —** Setting this bit will initiate the trigger contained in SPTG[4:0]. This bit will be cleared automatically after the trigger is queued. This bit may be set at the same time a new trigger is selected in SPTG[4:0].	0 = Normal Mode 1 = Initiate a trigger that is selected in SPTG[4:0]
SPTG[4:0] Bits 4-0	SPI Trigger Selection —These bits select which SPI Control Queue will be triggered when the ITG bit is set.	00000 = Selects Control Queue 0 … 11111 = Selects Control Queue 31

STPR_L SPI Trigger Priority Low Register

Addr $2485_1008

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	STPR 15	STPR 14	STPR 13	STPR 12	STPR 11	STPR 10	STPR 9	STPR 8	STPR 7	STPR 6	STPR 5	STPR 4	STPR 3	STPR 2	STPR 1	STPR 0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-11. STPR_L Description

Name	Description	Settings
STPR[15:0] Bits 15-0	**SPI Trigger Priority Assignment for triggers 15 through 0—** The STPR_L register determines to which priority FIFO, high or low, each Control Queue will be queued for triggers 15 through 0. Each SPI has its own high and low priority FIFO queue. All triggers in the high priority FIFO queue will be executed before any triggers in the low priority FIFO queue are executed. This register would be used in conjunction with the STPR_H register. Use the QCFG_H and QCFG_L registers for trigger assignments of SPI1 and SPI2.	0 = When triggered, Control Queue will be executed using the Low Priority FIFO queue 1 = When triggered, Control Queue will be executed using the High Priority FIFO queue

STPR_H SPI Trigger Priority High Register **Addr** $2485_100A

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	STPR 31	STPR 30	STPR 29	STPR 28	STPR 27	STPR 26	STPR 25	STPR 24	STPR 23	STPR 22	STPR 21	STPR 20	STPR 19	STPR 18	STPR 17	STPR 16
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-12. STPR_H Description

Name	Description	Settings
STPR[31:16] Bits 15-0	**SPI Trigger Priority Assignment for triggers 31 through 16—** The STPR_H register determines to which priority FIFO, high or low, each Control Queue will be queued for triggers 31 through 16. Each SPI has its own high and low priority FIFO queue. All triggers in the high priority FIFO queue will be executed before any triggers in the low priority FIFO queue are executed. This register would be used in conjunction with the STPR_L register.	0 = When triggered, Control Queue will be executed using the Low Priority FIFO queue 1 = When triggered, Control Queue will be executed using the High Priority FIFO queue

This register controls the default setups of SPI1 and SPI2.

SDEF SPI Default Register **Addr** $2485_100C

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									DBG2	DBG1	WAIT 2	WAIT 1	DOZE 2	DOZE 1	CPOL 2	CPOL 1
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-13. SDEF Description

Name	Description	Settings
Bits 15-6	Reserved	N/A
DBG[2:1] Bits 5-4	**Debug mode Enable bits for SPI2 and SPI1—** Debug mode, when activated, will cause the SPI(1,2) to halt pending transfers; current transfers will be completed. Each SPI can be dozed independently or together in debug mode. This operation is similar to the description for Wait Mode below, except that multiple and burst messages will be halted at the next single message boundary NOTE: Debug Mode Control is not implemented on ~~cdr3~~ versions of the MQSPI	0 = Selected SPI enabled when in Debug mode 1 = Selected SPI disabled when in Debug mode
WAIT[2:1] Bits 5-4	**Wait mode Enable bits for SPI2 and SPI1—** WAIT mode, when activated, will cause the SPI(1,2) to halt pending transfers; current transfers will be completed. Each SPI can be dozed independently or together. Either SPI is dozed when its Wait Mode Enable bit is set and the system is in Wait Mode. While in WAIT mode; triggers sent to the MQSPI will be stored in empty fifo queues ready for use when the MQSPI is no longer in WAIT mode. If WAIT mode is activated while a transfer has started; the transfer will complete before WAIT mode takes effect For the multiple message and burst modes this means that all messages in the transfer must finish before WAIT mode is activated. Burst triggers can be interleaved with other burst or non-burst triggers. NOTE: Wait Mode Control is not implemented on ~~cdr3~~ versions of the MQSPI	0 = Selected SPI enabled when in Wait mode 1 = Selected SPI disabled when in Wait mode

Table 49-13. SDEF Description

Name	Description	Settings
DOZE[2:1] Bits 3-2	**Doze mode Enable bits for SPI2 and SPI1—** DOZE mode, when activated, will cause the SPI(1,2) to halt pending transfers; current transfers will be completed. SPI operation will be the same for Doze mode as described above for Wait Mode.	0 = Selected SPI enabled when in Doze mode 1 = Selected SPI disabled when in Doze mode
CPOL[2:1] Bits 1-0	Clock Polarity for SPI2 and SPI1—See Figure 49-11 and Figure	0 = The inactive state value of the clock is low 1 = The inactive state value of the clock is high

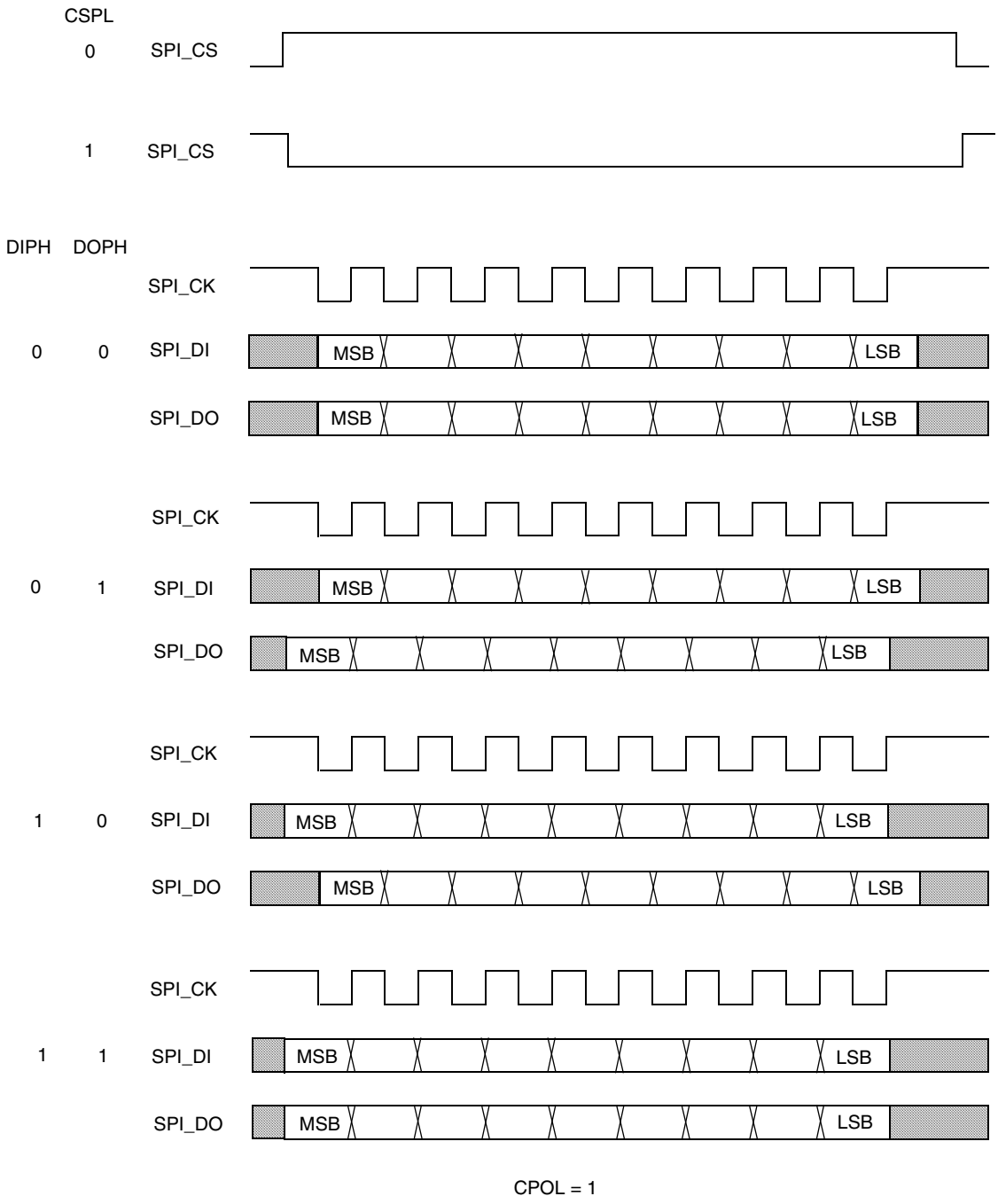


Figure 49-11. SPI timing CPOL=1

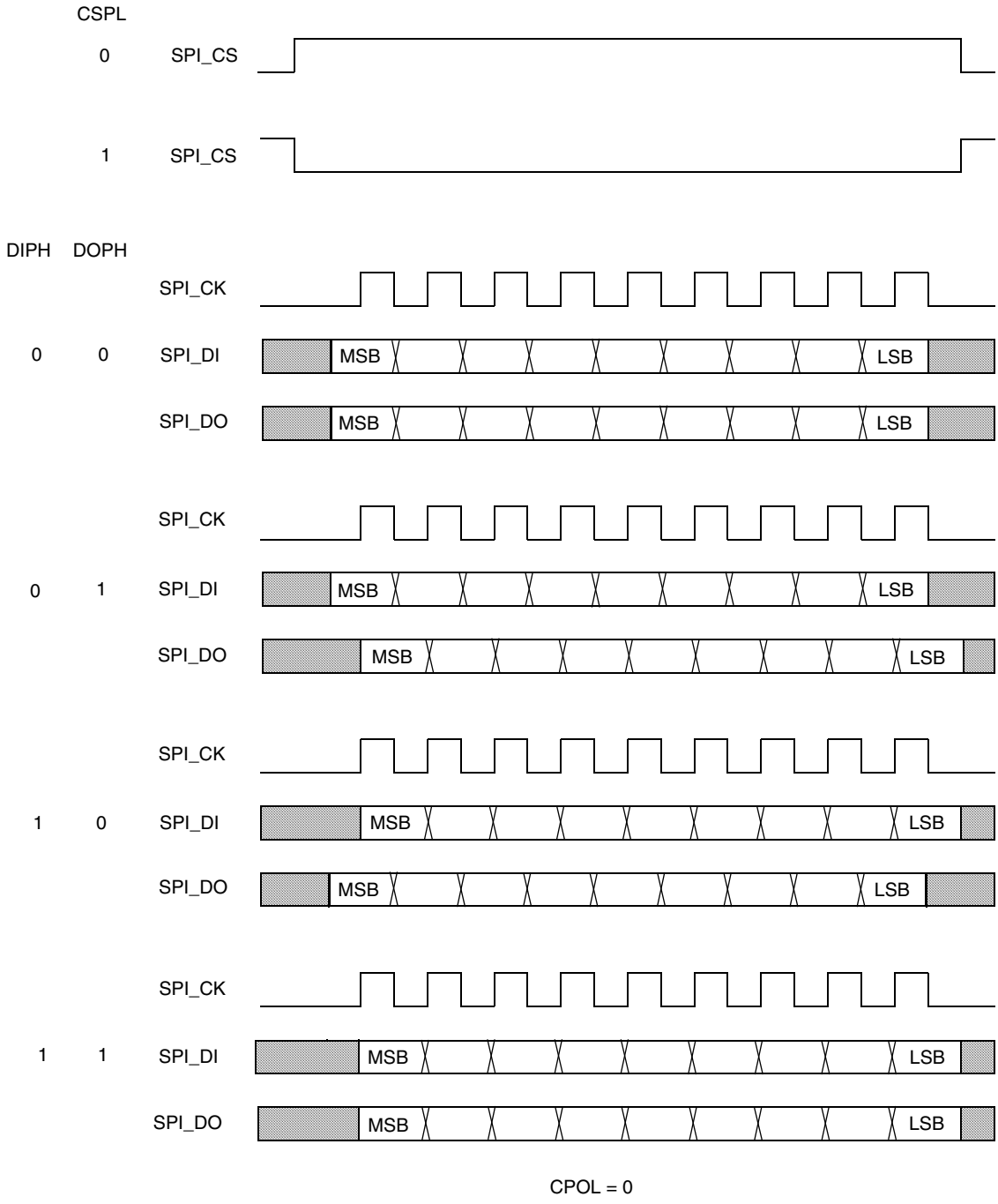


Figure 49-12. SPI timing CPOL =0

This register contains the interrupt flags. A flag bit is set when the event attached to that flag occurs. Most flag bits can be cleared by writing a 1 to it. Clearing the flag also removes the interrupt signal. References to HP and LP refer to High Priority and Low Priority, and references to 1 and 2 refer to SPI1 and SPI2.

SFLG SPI Flag Register **Addr** $2485_100E

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	HP QOE 1	HP QNFL 1	HP TSA1		LP QOE1	LP QNFL 1	LP TSA1	STF1	HP QOE2	HP QNFL 2	HP TSA2		LP QOE2	LP QNFL 2	LP TSA2	STF2
TYPE	w1c	r	w1c	r	w1c	r	w1c	r	w1c	r	w1c	r	w1c	r	w1c	r
RESET	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Table 49-14. SFLG Description

Name	Description	Settings
HPQOE1 Bit 15	Queue Overflow Error	0 = No Error 1 = The FIFO Trigger Queue is full and received a new trigger to store. The new trigger will be ignored. This bit is a write-1-to-clear bit.
HPQNFL1 Bit 14	**FIFO Queue Not Full—** If a number of transfers are needed to be performed that exceeds the FIFO trigger queue length, enough triggers can be initiated to fill the FIFO queue, causing the QNFL bit to go low, then wait for the interrupt caused by a space being open in the FIFO queue (QNFL = 1) before sending another trigger. These bits are read-only bits and cannot be cleared with a write. To mask the interrupt, the corresponding QNFLE bit in the SPIM register must be cleared.	0 = FIFO Trigger Queue is full. 1 = FIFO Trigger Queue is not full.
HPTSA1 Bit 13	Received Trigger While SPI Active	1 = A trigger was received while the SPI it was meant for was active. This is a write-1-to-clear bit.
Bit 12	Reserved	N/A
LPQOE1 Bit 11	Queue Overflow Error	0 = No Error 1 = The FIFO Trigger Queue is full and received a new trigger to store. The new trigger will be ignored. This bit is a write-1-to-clear bit.
LPQNFL1 Bit 10	**FIFO Queue Not Full—** If a number of transfers needed to be performed that exceeds the FIFO trigger queue length, enough triggers can be initiated to fill the FIFO queue, causing the QNFL bit to go low, then wait for the interrupt caused by a space being open in the FIFO queue (QNFL = 1) before sending another trigger. These bits are read-only bits and cannot be cleared with a write. To mask the interrupt, the corresponding QNFLE bit in the SPIM register must be cleared.	0 = FIFO Trigger Queue is full. 1 = FIFO Trigger Queue is not full.

Table 49-14. SFLG Description (Continued)

Name	Description	Settings
LPTSA1 Bit 9	Received Trigger While SPI Active	1 = A trigger was received while the SPI it was meant for was active. This is a write-1-to-clear bit.
STF1 Bit 8	**SPI Transfer Finished—** STF1 and STF2 are read-only bits. They can only be cleared by clearing the appropriate STFF_H and STFF_L register bit(s) using a write-1-to-clear, or by clearing the proper STFE bit in the STFE_H and STFE_L registers. The logic for this bit is controlled by the STFF, STFE, and QCFG registers. STF = STFE[31] & STFF[31] | STFE[30] & STFF[30] | … STFF[0] & STFE[0] The flag is further qualified by the QCFG register. STF1 will only be set for SPI1 transfers and STF2 will only be set for SPI2 transfers based on QCFG. The STF bit will be set after a transfer as long as the corresponding EN bit in the STFE register is set. However, an interrupt will only be generated if the STFE bit in the SPIM register, the STFE bit in the STFE_H/STFE_L register and the corresponding bit in the STFF_H/STFF_L register are set. The STFF bit should be cleared before setting the STFE bit in the STFE_H/STFE_L register since a prior transfer may have set the STFF flag.	1 = A SPI transfer finished
HPQOE2 Bit 7	Queue Overflow Error	0 = No Error 1 = The FIFO Trigger Queue is full and received a new trigger to store. The new trigger will be ignored. This bit is a write-1-to-clear bit.
HPQNFL2 Bit 6	**FIFO Queue Not Full—** If a number of transfers needed to be performed that exceeds the FIFO trigger queue length, enough triggers can be initiated to fill the FIFO queue, causing the QNFL bit to go low, then wait for the interrupt caused by a space being open in the FIFO queue (QNFL = 1) before sending another trigger. These bits are read-only bits and cannot be cleared with a write. To mask the interrupt, the corresponding QNFLE bit in the SPIM register must be cleared.	0 = FIFO Trigger Queue is full. 1 = FIFO Trigger Queue is not full.
HPTSA2 Bit 5	Received Trigger While SPI Active	1 = A trigger was received while the SPI it was meant for was active. This is a write-1-to-clear bit.
Bit 4	Reserved	N/A

Table 49-14. SFLG Description (Continued)

Name	Description	Settings
LPQOE2 Bit 3	Queue Overflow Error	0 = No Error 1 = The FIFO Trigger Queue is full and received a new trigger to store. The new trigger will be ignored. This bit is a write-1-to-clear bit.
LPQNFL2 Bit 2	**FIFO Queue Not Full—** If a number of transfers needed to be performed that exceeds the FIFO trigger queue length, enough triggers can be initiated to fill the FIFO queue, causing the QNFL bit to go low, then wait for the interrupt caused by a space being open in the FIFO queue (QNFL = 1) before sending another trigger. These bits are read-only bits and cannot be cleared with a write. To mask the interrupt, the corresponding QNFLE bit in the SPIM register must be cleared.	0 = FIFO Trigger Queue is full. 1 = FIFO Trigger Queue is not full.
LPTSA2 Bit 1	Received Trigger While SPI Active	1 = A trigger was received while the SPI it was meant for was active. This is a write-1-to-clear bit.
STF2 Bit 0	**SPI Transfer Finished—** STF1 and STF2 are read-only bits. They can only be cleared by clearing the appropriate STFF_H and STFF_L register bit(s) using a write-1-to-clear, or by clearing the proper STFE bit in the STFE_H and STFE_L registers. The logic for this bit is controlled by the STFF, STFE, and QCFG registers. STF = STFE[31] & STFF[31] | STFE[30] & STFF[30] | … STFF[0] & STFE[0] The flag is further qualified by the QCFG register. STF1 will only be set for SPI1 transfers and STF2 will only be set for SPI2 transfers based on QCFG. The STF bit will be set after a transfer as long as the corresponding EN bit in the STFE register is set. However, an interrupt will only be generated if the STFE bit in the SPIM register, the STFE bit in the STFE_H/STFE_L register and the corresponding bit in the STFF_H/STFF_L register are set. The STFF bit should be cleared before setting the STFE bit in the STFE_H/STFE_L register since a prior transfer may have set the STFF flag.	1 = A SPI transfer finished

The SPIM register is used to select which events will generate an interrupt. Masking an interrupt still allows polling of the SFLG register, but without generating an interrupt. References to HP and LP refer to High Priority and Low Priority, and references to 1 and 2 refer to SPI1 and SPI2.

SPIM SPI Interrupt Mask Register **Addr** $2485_1**010**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	HP QOE 1E	HP QNFL 1E	HP TSA1 E		LP QOE1 E	LP QNFL 1E	LP TSA1 E	STF1 E	HP QOE2 E	HP QNFL 2E	HP TSA2 E		LP QOE2 E	LP QNFL 2E	LP TSA2 E	STF2 E
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-15. SPIM Description

Name	Description	Settings
HPQOE1E Bit 15	Queue Overflow Error Enable	0 = Mask interrupt 1 = Enable corresponding QOE interrupt
HPQNFL1E Bit 14	**FIFO Trigger Queue Not Full Enable**	0 = Mask interrupt 1 = Enable corresponding QNFL interrupt
HPTSA1E Bit 13	Received Trigger While SPI Active Enable	0 = Mask Interrupt 1 = Enable corresponding TSA interrupt
Bit 12	Reserved	N/A
LPQOE1E Bit 11	Queue Overflow Error Enable	0 = Mask interrupt 1 = Enable corresponding QOE interrupt
LPQNFL1E Bit 10	FIFO Trigger Queue Not Full Enable	0 = Mask interrupt 1 = Enable corresponding QNFL interrupt
LPTSA1E Bit 9	Received Trigger While SPI Active Enable	0 = Mask Interrupt 1 = Enable corresponding TSA interrupt
STF1E Bit 8	SPI Transfer Finished Enable	0 = Mask Interrupt 1 = Enable corresponding STF interrupt
HPQOE2E Bit 7	Queue Overflow Error Enable	0 = Mask interrupt 1 = Enable corresponding QOE interrupt
HPQNFL2 Bit 6	FIFO Trigger Queue Not Full Enable	0 = Mask interrupt 1 = Enable corresponding QNFL interrupt
HPTSA2E Bit 5	Received Trigger While SPI Active Enable	0 = Mask Interrupt 1 = Enable corresponding TSA interrupt
Bit 4	Reserved	N/A
LPQOE2E Bit 3	Queue Overflow Error Enable	0 = Mask interrupt 1 = Enable corresponding QOE interrupt
LPQNFL2E Bit 2	FIFO Trigger Queue Not Full Enable	0 = Mask interrupt 1 = Enable corresponding QNFL interrupt

Table 49-15. SPIM Description (Continued)

Name	Description	Settings
LPTSA2E Bit 1	Received Trigger While SPI Active Enable	0 = Mask Interrupt 1 = Enable corresponding TSA interrupt
STF2E Bit 0	SPI Transfer Finished Enable	0 = Mask Interrupt 1 = Enable corresponding STF interrupt

STFF_L SPI Transfer Finished Flag Low Register **Addr** $2485_1012

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	STFF 15	STFF 14	STFF 13	STFF 12	STFF 11	STFF 10	STF9	STFF 8	STFF 7	STFF 6	STFF 5	STFF 4	STFF 3	STFF 2	STFF 1	STFF 0
TYPE	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 49-16. STFF_L Description

Name	Description	Settings
STFF[15:0] Bits 15-0	**SPI Transfer Finished Flag for triggers 15 through 0—** The STFF_L register, in combination with the STFF_H register, contains a flag for each Control Queue. After a transfer finishes, the flag corresponding to the Control Queue that was used will be set. For example, if trigger 5 is initiated causing Control Queue 5 to be used for the transfer, then after the transfer has ended, bit 5 in the STFF_L register will be set. The reset value for STFF_L is 0xFFFF since all transfers are finished, that is, not active.	0 = Transfer not finished 1 = Transfer finished (not active)

STFF_H SPI Transfer Finished Flag High Register **Addr** $2485_1014

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	STFF 31	STFF 30	STFF 29	STFF 28	STFF 27	STFF 26	STFF 25	STFF 24	STFF 23	STFF 22	STFF 21	STFF 20	STFF 19	STF F 18	STFF 17	STFF 16
TYPE	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 49-17. STFF_H Description

Name	Description	Settings
STFF[31:16] Bits 15-0	**SPI Transfer Finished Flag for triggers 31 through 16—**The STFF_H register, in combination with the STFF_L register, contains a flag for each Control Queue. After a transfer finishes, the flag corresponding to the Control Queue that was used will be set. For example, if trigger 25 is initiated causing Control Queue 25 to be used for the transfer, then after the transfer has ended, bit 9 in the STFF_H register will be set. The reset value for STFF_H is 0xFFFF since all transfers are finished, that is, not active.	0 = Transfer not finished 1 = Transfer finished (not active)

STFE_L SPI Transfer Finished Flag Enable Low Register **Addr** $2485_1016

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	STFE 15	STFE 14	STFE 13	STFE 12	STFE 11	STFE 10	STFE 9	STFE 8	STFE 7	STFE 6	STFE 5	STFE 4	STFE 3	STFE 2	STFE 1	STFE 0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-18. STFE_L Description

Name	Description	Settings
STFE[15:0] Bits 15-0	**SPI Transfer Finished Flag Enable for triggers 15 through 0—**The STFE_L register, in combination with the STFE_H register, contains enable bits to enable the corresponding STFF bits in the STFF_L register to cause the STF1 or STF2 bits in the SPIM register to become set after a transfer finishes.	0 = Not enabled for interrupt 1 = Enable corresponding STFF bit in the STFF_L register to cause an interrupt when set

STFE_H — SPI Transfer Finished Flag Enable High Register — **Addr** $2485_1018

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	STFE 31	STFE 30	STFE 29	STFE 28	STFE 27	STFE 26	STFE 25	STFE 24	STFE 23	STFE 22	STFE 21	STFE 20	STFE 19	STF E 18	STFE 17	STFE 16
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-19. STFE_H Description

Name	Description	Settings
STFE[31:16] Bits 15-0	**SPI Transfer Finished Enable for triggers 31 through 16—** The STFE_H register, in combination with the STFE_L register, contains enable bits to enable the corresponding STFF bits in the STFF_H register to cause the STF1 or STF2 bits in the SPIM register to become set after a transfer finishes	0 = Not enabled for interrupt 1 = Enable corresponding STFF bit in the STFF_H register to cause an interrupt when set.

QRE_L — Queue Read Enable Low Register

Addr $2485_101A

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QRE15	QRE14	QRE13	QRE12	QRE11	QRE10	QRE9	QRE8	QRE7	QRE6	QRE5	QRE4	QRE3	QRE2	QRE1	QRE0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-20. QRE_L Description

Name	Description	Settings
QRE[15:0] Bits 15-0	Control Queue Read Enable for triggers 15 through 0— The QRE_L register, in combination with the QRE_H register, contains enable bits to do a read from a peripheral in addition to a write during the same transfer.	0 = Only write to the peripheral during a transfer 1 = Read from and write to the peripheral during a transfer

QRE_H — Queue Read Enable High Register

Addr $2485_101C

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QRE31	QRE30	QRE29	QRE28	QRE27	QRE26	QRE25	QRE24	QRE23	QRE22	QRE21	QRE20	QRE19	QRE18	QRE17	QRE16
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-21. QRE_L Description

Name	Description	Settings
QRE[31:16] Bits 15-0	**Control Queue Read Enable for triggers 31 through 16—**The QRE_H register, in combination with the QRE_L register, contains enable bits to do a read from a peripheral in addition to a write during the same transfer.	0 = Only write to the peripheral during a transfer 1 = Read from and write to the peripheral during a transfer

SDI_MD_L Serial Display Interface Mode Low Register **Addr** $2485_101E

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	SDI-MD_15	SDI-MD_14	SDI-MD_13	SDI-MD_12	SDI-MD_11	SDI-MD_10	SDI-MD_9	SDI-MD_8	SDI-MD_7	SDI-MD_6	SDI-MD_5	SDI-MD_4	SDI-MD_3	SDI-MD_2	SDI-MD_1	SDI-MD_0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-22. SDI_MD_L Description

Name	Description	Settings
SDI-MD[15:0] Bits 15-0	**Serial Display Interface Mode enable bits—**The SDI_MD_L register, in combination with the SDI_MD_H register contains bits which enable the serial display interface. There is one enable bit per trigger.	0 = Serial Display disabled 1 = Serial Display enabled

SDI_MD_H — Serial Display Interface Mode High Register — Addr $2485_1020

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	SDI-MD_31	SDI-MD_30	SDI-MD_29	SDI-MD_28	SDI-MD_27	SDI-MD_26	SDI-MD_25	SDI-MD_24	SDI-MD_23	SDI-MD_22	SDI-MD_21	SDI-MD_20	SDI-MD_19	SDI-MD_18	SDI-MD_17	SDI-MD_16
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-23. SDI_MD_H Description

Name	Description	Settings
SDI-MD[31:16] Bits 15-0	**Serial Display Interface Mode enable bits—**The SDI_MD_H register, in combination with the SDI_MD_L register, contains bits which enable the serial display interface. There is one enable bit per trigger.	0 = Serial Display disabled 1 = Serial Display enabled

The CDPS1 register contains the control pointer data used for the current SPI1 transfer. The data for this register comes from a Control Data Pointer register in RAM. When SPI1 is active from a trigger, the data stored in the selected Control Queue's Control Data Pointer register is loaded into this register. This is a read-only register.

CDPS1 SPI1 Control Data Pointer Status Register **Addr** $2485_1022

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QW STR7	QW STR6	QW STR5	QW STR4	QW STR3	QW STR2	QW STR1	QW STR0	QR STR7	QR STR6	QR STR5	QR STR4	QR STR3	QR STR2	QR STR1	QR STR0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-24. CDPS1 Description

Name	Description	Settings
QWSTR[7:0] Bits 15-8	**Starting Location of Write Transfer Data—** QWSTR initially contains the starting location in RAM for the data to be written out of the SPI. The QWSTR becomes a real-time pointer that is updated as processing of the transfer progresses.	QWSTR = (DStart - $480)/2 (See description in SPICDP register on page 49-30)
QRSTR[7:0] Bits 7-0	Starting Location of Read Transfer Data— QRSTR initially contains the starting location in RAM for the data to be stored when reading from the SPI. The QRSTR becomes a real-time pointer that is updated as processing of the transfer progresses.	QRSTR = (DStart - $480)/2 (See description in SPICDP register on page 49-30)

The CDMS1 register contains the control mode data used for the current SPI1 transfer. The data for this register comes from a Control Data Mode register in RAM. When SPI1 is active from a trigger, the data stored in the selected Control Queue's Control Data Mode register is loaded into this register. This is a read-only register.

CDMS1 SPI1 Control Data Mode Status Register **Addr** $2485_1**024**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QBRST	QRC5	QRC4	QRC3	QRC2	QRC1	QRC0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-25. CDMS1 Description

Name	Description	Settings
QBRST Bit 15	Queue Burst Mode Enable (Available only for Control Queues 0 through 3)— The QBRST bit is only valid for Control Queues 0 through 3, else it reads as 0.	0 = Burst mode disabled 1 = Burst mode enabled
QRC[5:0] Bits 14-9	Queue Multiple Message Repeat Count— QRC initially specifies the number of messages to be transferred. If burst mode is enabled (QBRST bit is set), then this indicates the number of remaining messages to be transferred.	000000 = 1 message to be transferred … 111111 = 64 messages to be transferred
QBL[4:0] Bits 8-4	Queue Message Data Byte Length—QBL indicates the number of whole bytes to be transferred per message.	00000 = 1 byte message … 11111 = 32 byte message
QCS[3:0] Bits 3-0	Queue Chip Select— QCS indicates which chip select pin is being used for the SPI transfer.	0000 = SPI_CS[0] … 1001 = SPI_CS[9]

The CDPS2 register contains the control pointer data used for the current SPI2 transfer. The data for this register comes from a Control Data Pointer register in RAM. When SPI2 is active from a trigger, the data stored in the selected Control Queue's Control Data Pointer register is loaded into this register. This is a read-only register.

CDPS2 SPI2 Control Data Pointer Status Register **Addr** $2485_1026

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QW STR7	QW STR6	QW STR5	QW STR4	QW STR3	QW STR2	QW STR1	QW STR0	QR STR7	QR STR6	QR STR5	QR STR4	QR STR3	QR STR2	QR STR1	QR STR0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-26. CDPS2 Description

Name	Description	Settings
QWSTR[7:0] Bits 15-8	**Starting Location of Write Transfer Data—** QWSTR initially contains the starting location in RAM for the data to be written out of the SPI. The QWSTR becomes a real-time pointer that is updated as processing of the transfer progresses.	QWSTR = (DStart - $480)/2 (See description in SPICDP register on page 49-30)
QRSTR[7:0] Bits 7-0	**Starting Location of Read Transfer Data—** QRSTR initially contains the starting location in RAM for the data to be stored when reading from the SPI. The QRSTR becomes a real-time pointer that is updated as processing of the transfer progresses.	QRSTR = (DStart - $480)/2 (See description in SPICDP register on page 49-30)

The CDMS2 register contains the control mode data used for the current SPI2 transfer. The data for this register comes from a Control Data Mode register in RAM. When SPI2 is active from a trigger, the data stored in the selected Control Queue's Control Data Mode register is loaded into this register. This is a read-only register. The QBRST bit is only valid for Control Queues 0 through 3, else it reads as 0.

CDMS2 SPI2 Control Data Mode Status Register **Addr** $2485_1028

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QBRST	QRC5	QRC4	QRC3	QRC2	QRC1	QRC0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-27. CDMS2 Description

Name	Description	Settings
QBRST Bit 15	Queue Burst Mode Enable (Available only for Control Queues 0 through 3)— The QBRST bit is only valid for Control Queues 0 through 3, else it reads as 0.	0 = Burst mode disabled 1 = Burst mode enabled
QRC[5:0] Bits 14-9	Queue Multiple Message Repeat Count— QRC initially specifies the number of messages to be transferred. If burst mode is enabled (QBRST bit is set), then this indicates the number of remaining messages to be transferred.	000000 = 1 message to be transferred … 111111 = 64 messages to be transferred
QBL[4:0] Bits 8-4	Queue Message Data Byte Length—QBL indicates the number of whole bytes to be transferred per message.	00000 = 1 byte message … 11111 = 32 byte message
QCS[3:0] Bits 3-0	Queue Chip Select— QCS indicates which chip select pin is being used for the SPI transfer.	0000 = SPI_CS[0] … 1001 = SPI_CS[9]

This register contains the SPI configuration A data specifically for chip select 0, MQSPI_CS0.

CSCFG0A SPI Chip Select 0 Configuration A Register **Addr** $2485_102A

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
					CK DIV6	CK DIV5	CK DIV4	CK DIV3	CK DIV2	CK DIV1	CK DIV0		M_L_ SEL	CSPL	DOPH	DIPH
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-28. CSCFG0A Description

Name	Description	Settings
Bits 15-12	Reserved	N/A
CKDIV[6:0] Bits 11-5	**Clock Divide—**The CKDIV[6:0] contains a divide value for determining the clock frequency to use for this chip select. Refer to Table 49-29 for the complete listing of possible SPI clock frequencies for a system clock of 16.8 MHz. **Note:** Even divisors result in a 50% duty cycle. Odd divisors result in a clock waveform which is active low one PAT_REF clock period longer than the active high time.	Refer to Table 49-29
Bit 4	Reserved	N/A
M_L_SEL Bit 3	**Most Significant and Least Significant Select Bit—** This bit will determine if the first serial bit transmitted or received to or from the SPI is an MSB/LSB bit. The M_L_SEL bit has obvious effects on the serial data being sent, but subtle effects on how the data is taken out of RAM when sending an odd number of bytes. The following example shows how the data is fetched. TRANSMIT: Below is an example of RAM data in the MQSPI. This data when accessed for transfer is sent to a register before being loaded into a shift register and transmitted serially. Using this data as an example if a single 3 byte message were to be transferred in MSB first mode. The serial out data would be C8B944; if the LSB mode were selected 9D1325. Where the B9 byte is sent out LSB first which gives 9D, then the C8 byte is sent out LSB first which gives 13 and finally the A4 byte is sent out LSB first which gives 25. Notice that for the MSB mode the last byte sent out is 44 while for the LSB mode the last byte sent out is A4(LSB first which is 25). RAM MSB LSB C8 B9 44 A4	0 = MSB is transferred first 1= LSB is transferred first

Table 49-28. CSCFG0A Description (Continued)

Name	Description	Settings
M_L_SEL Bit 3 (Continued)	RECEIVE: The underlying assumption for the LSB mode is that the MQSPI is recognized as an MSB machine and the device being interfaced to is an LSB machine. Thus when the MQSPI receives LSB data it will pack the data into its internal RAM such that the lsb bit remains the lsb bit; this also means the lsb byte remains the lsb byte and the msb byte remains the msb byte within the MQSPI RAM. Note, in lsb mode this means that for an odd number of bytes the MQSPI packs the lsb byte into its RAM's lsb location! For an odd number of bytes in msb mode the MQSPI packs the msb byte into its RAM's msb location. For example: Assume the data in the target RAM (in this case the testbench) is C8B9 (first word) and 44A4 (second word). If in the LSB mode; for a three byte single message, data sent to the MQSPI would be 9D1325. Where 9D is B9 reordered; 13 is C8 reordered and 25 is A4 reordered. The way the data would be packed in the MQSPI RAM is as follows. The first RAM word least significant byte would get B9; the first RAM word most significant byte would get C8; the second RAM word least significant byte would get A4. Likewise, if the MQSPI is in the MSB mode; for a three byte single message, data sent to the MQSPI would be C8B944. The way the data would be packed in the MQSPI RAM is as follows. The first RAM word most significant byte would get C8; the first RAM word least significant byte would get B9; the second RAM word most significant byte would get 44.	0 = MSB is transferred first. 1= LSB is transferred first.
CSPL Bit 2	**Chip Select Polarity—** Each device can be configured for an active-high or an active-low chip select. The DOPH and DIPH bits are provided in lieu of the CPHA (clock phase selection) bit normally used for SPI peripherals. See also Figure 49-11 on page 49-42 and Figure on page 49-43	0 = Chip select will go high to start a transfer and low to end a transfer (active-high chip select) 1 = Chip select will go low to start a transfer and high to end a transfer (active-low chip select)
DOPH Bit 1	**Data Out Phase** Each device can be configured to clock data out on the falling or rising edge of the SPI clock. The DOPH and DIPH bits are provided in lieu of the CPHA (clock phase selection) bit normally used for SPI peripherals. See also Figure 49-11 on page 49-42 and Figure on page 49-43	0 = Data out is changed on the falling edge of the SPI clock (captured on the rising edge at the device) 1 = Data out is changed on the rising edge of the SPI clock (captured on the falling edge at the device)
DIPH Bit 0	**Data In Phase—** Each device can be configured to clock data in on the falling or rising edge of the SPI clock. See also Figure 49-11 on page 49-42 and Figure on page 49-43	0 = Data in is captured on the rising edge of the SPI clock (changed on the falling edge at the device) 1 = Data in is captured on the falling edge of the SPI clock (changed on the rising edge at the device)

Table 49-29. Clock Divider Table – 16.8 MHz System Clock

Divisor	SPI Clock (kHz)	Divisor	SPI Clock (kHz)	Divisor	SPI Clock (kHz)	Divisor	SPI Clock (kHz)
0	16800.00	32	509.09	64	258.46	96	173.20
1	8400.00	33	494.12	65	254.55	97	171.43
2	5600.00	34	480.00	66	250.75	98	169.70
3	4200.00	35	466.67	67	247.06	99	168.00
4	3360.00	36	454.05	68	243.48	100	166.34
5	2800.00	37	442.11	69	240.00	101	164.71
6	2400.00	38	430.77	70	236.62	102	163.11
7	2100.00	39	420.00	71	233.33	103	161.54
8	1866.67	40	409.76	72	230.14	104	160.00
9	1680.00	41	400.00	73	227.03	105	158.49
10	1527.27	42	390.70	74	224.00	106	157.01
11	1400.00	43	381.82	75	221.05	107	155.56
12	1292.31	44	373.33	76	218.18	108	154.13
13	1200.00	45	365.22	77	215.38	109	152.73
14	1120.00	46	357.45	78	212.66	110	151.35
15	1050.00	47	350.00	79	210.00	111	150.00
16	988.24	48	342.86	80	207.41	112	148.67
17	933.33	49	336.00	81	204.88	113	147.37
18	884.21	50	329.41	82	202.41	114	146.09
19	840.00	51	323.08	83	200.00	115	144.83
20	800.00	52	316.98	84	197.65	116	143.59
21	763.64	53	311.11	85	195.35	117	142.37
22	730.43	54	305.45	86	193.10	118	141.18
23	700.00	55	300.00	87	190.91	119	140.00
24	672.00	56	294.74	88	188.76	120	138.84
25	646.15	57	289.66	89	186.67	121	137.70
26	622.22	58	284.75	90	184.62	122	136.59
27	600.00	59	280.00	91	182.61	123	135.48
28	579.31	60	275.41	92	180.65	124	134.40

Table 49-29. Clock Divider Table – 16.8 MHz System Clock (Continued)

29	560.00	61	270.97	93	178.72	125	133.33
30	541.94	62	266.67	94	176.84	126	132.28
31	525.00	63	262.50	95	175.00	127	131.25

This register contains the SPI configuration B data specifically for chip select 0, MQSPI_CS0. The Delay After Transfer (DAT) and Delay Before Clock (DBC) are programmable via 7 bit words located in the CSCFG0-9B Registers. The CSCFG Registers offer programmability based on Chip Select. This programmability is available for both SPI1 and SPI2. The DAT_CNT is set in bits 7 through 13, the DBC_CNT is set in bits 0 through 6.The value of the DAT_CNT or DBC_CNT represents the programmed delay. There are 128 programmable selections; one of which is no delay. If either the DAT_CNT or DBC_CNT is set to zero; no additional delays are added, this would be for devices that do not want extra delay in the transfer. If either the DAT_CNT or DBC_CNT is set to 127, 127 clock delays will be set. DBC is defined as the additional delay added after the chip select has been asserted to the first SPI_CLK edge. DAT is defined as the additional delay of the deasserted chip select between transfers.

CSCFG0B SPI Chip Select 0 Configuration B Register **Addr** $2485_102C

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
			DAT_CNT_6	DAT_CNT_5	DAT_CNT_4	DAT_CNT_3	DAT_CNT_2	DAT_CNT_1	DAT_CNT_0	DBC_CNT_6	DBC_CNT_5	DBC_CNT_4	DBC_CNT_3	DBC_CNT_2	DBC_CNT_1	DBC_CNT_0
TYPE	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-30. CSCFG0B Description

Name	Description	Settings
Bits 15-14	Reserved	N/A
DAT_CNT[6:0] Bits 13-7	**Delay After Transfer—** Slow devices on the SPI bus may require additional time after the chip select becomes inactive at the end of a transfer before the same chip select or different chip select becomes active again for a transfer immediately following. To accommodate these devices, setting the DAT, delays the chip select inactive region between transfers by the programmable DAT_CNT in the table below. For example, if the SPI clock frequency is set to 16.8Mhz, then setting the DAT_CNT to 4 will extend the chip select inactive time between transfers to 4 X 59.525ns(16.8Mhz period) = 238.1ns	0= Normal Mode 1 <= DAT_CNT <= 127 = Extend minimum chip select inactive time after a transfer by 'DAT_CNT' serial clock periods
DBC_CNT[6:0] Bits 6-0	**Delay Before First Clock Edge—** Slow devices on the SPI bus may require additional time before the first SPI clock edge after the chip select becomes active. To accommodate these devices, setting the DBC delays the first edge of the SPI clock by the programmable DBC_CNT in the table below. For example, if the SPI clock frequency is set to 16.8 Mhz, then setting the DBC_CNT to 4 will extend the chip select active time before the first SPI_CLK to 4 X 59.525ns(16.8Mhz period) = 238.1ns.	0 = Normal Mode 1 <= DAT_DBC <= 127 = Delay first clock edge of the SPI transfer by 'count' serial clock period

Table 49-31. Delay After Transfer Table

DAT_CNT	DAT Delay	DAT_CNT	DAT Delay	DAT_CNT	DAT Delay	DAT_CNT	DAT Delay
0	No Delay	32	32 SPI_CLK's	64	64 SPI_CLK's	96	96 SPI_CLK's
1	1 SPI_CLK	33	33 SPI_CLK's	65	65 SPI_CLK's	97	97 SPI_CLK's
2	2 SPI_CLK's	34	34 SPI_CLK's	66	66 SPI_CLK's	98	98 SPI_CLK's
3	3 SPI_CLK's	35	35 SPI_CLK's	67	67 SPI_CLK's	99	99 SPI_CLK's
4	4 SPI_CLK's	36	36 SPI_CLK's	68	68 SPI_CLK's	100	100 SPI_CLK's
5	5 SPI_CLK's	37	37 SPI_CLK's	69	69 SPI_CLK's	101	101 SPI_CLK's
6	6 SPI_CLK's	38	38 SPI_CLK's	70	70 SPI_CLK's	102	102 SPI_CLK's
7	7 SPI_CLK's	39	39 SPI_CLK's	71	71 SPI_CLK's	103	103 SPI_CLK's
8	8 SPI_CLK's	40	40 SPI_CLK's	72	72 SPI_CLK's	104	104 SPI_CLK's
9	9 SPI_CLK's	41	41 SPI_CLK's	73	73 SPI_CLK's	105	105 SPI_CLK's
10	10 SPI_CLK's	42	42 SPI_CLK's	74	74 SPI_CLK's	106	106 SPI_CLK's
11	11 SPI_CLK's	43	43 SPI_CLK's	75	75 SPI_CLK's	107	107 SPI_CLK's
12	12 SPI_CLK's	44	44 SPI_CLK's	76	76 SPI_CLK's	108	108 SPI_CLK's
13	13 SPI_CLK's	45	45 SPI_CLK's	77	77 SPI_CLK's	109	109 SPI_CLK's
14	14 SPI_CLK's	46	46 SPI_CLK's	78	78 SPI_CLK's	110	110 SPI_CLK's
15	15 SPI_CLK's	47	47 SPI_CLK's	79	79 SPI_CLK's	111	111 SPI_CLK's
16	16 SPI_CLK's	48	48 SPI_CLK's	80	80 SPI_CLK's	112	112 SPI_CLK's
17	17 SPI_CLK's	49	49 SPI_CLK's	81	81 SPI_CLK's	113	113 SPI_CLK's
18	18 SPI_CLK's	50	50 SPI_CLK's	82	82 SPI_CLK's	114	114 SPI_CLK's

Table 49-31. Delay After Transfer Table (Continued)

19	19 SPI_CLK's	51	51 SPI_CLK's	83	83 SPI_CLK's	115	115 SPI_CLK's
20	20 SPI_CLK's	52	52 SPI_CLK's	84	84 SPI_CLK's	116	116 SPI_CLK's
21	21 SPI_CLK's	53	53 SPI_CLK's	85	85 SPI_CLK's	117	117 SPI_CLK's
22	22 SPI_CLK's	54	54 SPI_CLK's	86	86 SPI_CLK's	118	118 SPI_CLK's
23	23 SPI_CLK's	55	55 SPI_CLK's	87	87 SPI_CLK's	119	119 SPI_CLK's
24	24 SPI_CLK's	56	56 SPI_CLK's	88	88 SPI_CLK's	120	120 SPI_CLK's
25	25 SPI_CLK's	57	57 SPI_CLK's	89	89 SPI_CLK's	121	121 SPI_CLK's
26	26 SPI_CLK's	58	58 SPI_CLK's	90	90 SPI_CLK's	122	122 SPI_CLK's
27	27 SPI_CLK's	59	59 SPI_CLK's	91	91 SPI_CLK's	123	123 SPI_CLK's
28	28 SPI_CLK's	60	60 SPI_CLK's	92	92 SPI_CLK's	124	124 SPI_CLK's
29	29 SPI_CLK's	61	61 SPI_CLK's	93	93 SPI_CLK's	125	125 SPI_CLK's
30	30 SPI_CLK's	62	62 SPI_CLK's	94	94 SPI_CLK's	126	126 SPI_CLK's
31	31 SPI_CLK's	63	63 SPI_CLK's	95	95 SPI_CLK's	127	127 SPI_CLK's

Table 49-32. Delay Before Transfer Table

DBC_CNT	DBC Delay	DBC_CNT	DBC Delay	DBC_CNT	DBC Delay	DBC_CNT	DBC Delay
0	No Delay	32	32 SPI_CLK's	64	64 SPI_CLK's	96	96 SPI_CLK's
1	1 SPI_CLK	33	33 SPI_CLK's	65	65 SPI_CLK's	97	97 SPI_CLK's
2	2 SPI_CLK's	34	34 SPI_CLK's	66	66 SPI_CLK's	98	98 SPI_CLK's
3	3 SPI_CLK's	35	35 SPI_CLK's	67	67 SPI_CLK's	99	99 SPI_CLK's
4	4 SPI_CLK's	36	36 SPI_CLK's	68	68 SPI_CLK's	100	100 SPI_CLK's
5	5 SPI_CLK's	37	37 SPI_CLK's	69	69 SPI_CLK's	101	101 SPI_CLK's
6	6 SPI_CLK's	38	38 SPI_CLK's	70	70 SPI_CLK's	102	102 SPI_CLK's
7	7 SPI_CLK's	39	39 SPI_CLK's	71	71 SPI_CLK's	103	103 SPI_CLK's
8	8 SPI_CLK's	40	40 SPI_CLK's	72	72 SPI_CLK's	104	104 SPI_CLK's
9	9 SPI_CLK's	41	41 SPI_CLK's	73	73 SPI_CLK's	105	105 SPI_CLK's
10	10 SPI_CLK's	42	42 SPI_CLK's	74	74 SPI_CLK's	106	106 SPI_CLK's
11	11 SPI_CLK's	43	43 SPI_CLK's	75	75 SPI_CLK's	107	107 SPI_CLK's
12	12 SPI_CLK's	44	44 SPI_CLK's	76	76 SPI_CLK's	108	108 SPI_CLK's
13	13 SPI_CLK's	45	45 SPI_CLK's	77	77 SPI_CLK's	109	109 SPI_CLK's
14	14 SPI_CLK's	46	46 SPI_CLK's	78	78 SPI_CLK's	110	110 SPI_CLK's
15	15 SPI_CLK's	47	47 SPI_CLK's	79	79 SPI_CLK's	111	111 SPI_CLK's
16	16 SPI_CLK's	48	48 SPI_CLK's	80	80 SPI_CLK's	112	112 SPI_CLK's
17	17 SPI_CLK's	49	49 SPI_CLK's	81	81 SPI_CLK's	113	113 SPI_CLK's
18	18 SPI_CLK's	50	50 SPI_CLK's	82	82 SPI_CLK's	114	114 SPI_CLK's

Table 49-32. Delay Before Transfer Table (Continued)

19	19 SPI_CLK's	51	51 SPI_CLK's	83	83 SPI_CLK's	115	115 SPI_CLK's
20	20 SPI_CLK's	52	52 SPI_CLK's	84	84 SPI_CLK's	116	116 SPI_CLK's
21	21 SPI_CLK's	53	53 SPI_CLK's	85	85 SPI_CLK's	117	117 SPI_CLK's
22	22 SPI_CLK's	54	54 SPI_CLK's	86	86 SPI_CLK's	118	118 SPI_CLK's
23	23 SPI_CLK's	55	55 SPI_CLK's	87	87 SPI_CLK's	119	119 SPI_CLK's
24	24 SPI_CLK's	56	56 SPI_CLK's	88	88 SPI_CLK's	120	120 SPI_CLK's
25	25 SPI_CLK's	57	57 SPI_CLK's	89	89 SPI_CLK's	121	121 SPI_CLK's
26	26 SPI_CLK's	58	58 SPI_CLK's	90	90 SPI_CLK's	122	122 SPI_CLK's
27	27 SPI_CLK's	59	59 SPI_CLK's	91	91 SPI_CLK's	123	123 SPI_CLK's
28	28 SPI_CLK's	60	60 SPI_CLK's	92	92 SPI_CLK's	124	124 SPI_CLK's
29	29 SPI_CLK's	61	61 SPI_CLK's	93	93 SPI_CLK's	125	125 SPI_CLK's
30	30 SPI_CLK's	62	62 SPI_CLK's	94	94 SPI_CLK's	126	126 SPI_CLK's
31	31 SPI_CLK's	63	63 SPI_CLK's	95	95 SPI_CLK's	127	127 SPI_CLK's

This register contains the SPI configuration A data specifically for chip select 1, MQSPI_CS1. Configuration of this register is identical to the CSCFG0A register. Refer to the CSCFG0A register definition, for definition of this register.

CSCFG1A — SPI Chip Select 1 Configuration A Register — **Addr** $2485_102E

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
					CK DIV6	CK DIV5	CK DIV4	CK DIV3	CK DIV2	CK DIV1	CK DIV0		M_L_ SEL	CSPL	DOPH	DIPH
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register contains the SPI configuration B data specifically for chip select 1, MQSPI_CS1. Configuration of this register is identical to the CSCFG0B register. Refer to the CSCFG0B register definition, for definition of this register.

CSCFG1B — SPI Chip Select 1 Configuration B Register — **Addr** $2485_1030

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
			DAT_ CNT_ 6	DAT_ CNT_ 5	DAT_ CNT_ 4	DAT_ CNT_ 3	DAT_ CNT_ 2	DAT_ CNT_ 1	DAT_ CNT_ 0	DBC_ CNT_ 6	DBC_ CNT_ 5	DBC_ CNT_ 4	DBC_ CNT_ 3	DBC_ CNT_ 2	DBC_ CNT_ 1	DBC_ CNT_ 0
TYPE	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register contains the SPI configuration A data specifically for chip select 2, MQSPI_CS2. Configuration of this register is identical to the CSCFG0A register. Refer to the CSCFG0A register definition, for definition of this register.

CSCFG2A — SPI Chip Select 2 Configuration A Register — **Addr** $2485_1032

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
					CK DIV6	CK DIV5	CK DIV4	CK DIV3	CK DIV2	CK DIV1	CK DIV0		M_L_ SEL	CSPL	DOPH	DIPH
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register contains the SPI configuration B data specifically for chip select 2, MQSPI_CS2. Configuration of this register is identical to the CSCFG0B register. Refer to the CSCFG0B register definition, for definition of this register.

CSCFG2B — SPI Chip Select 2 Configuration B Register — **Addr** $2485_1034

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
			DAT_ CNT_ 6	DAT_ CNT_ 5	DAT_ CNT_ 4	DAT_ CNT_ 3	DAT_ CNT_ 2	DAT_ CNT_ 1	DAT_ CNT_ 0	DBC_ CNT_ 6	DBC_ CNT_ 5	DBC_ CNT_ 4	DBC_ CNT_ 3	DBC_ CNT_ 2	DBC_ CNT_ 1	DBC_ CNT_ 0
TYPE	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register contains the SPI configuration A data specifically for chip select 3, MQSPI_CS3. Configuration of this register is identical to the register. Refer to the CSCFG0A register definition, for definition of this register.

CSCFG3A SPI Chip Select 3 Configuration A Register **Addr** $2485_1036

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
					CK DIV6	CK DIV5	CK DIV4	CK DIV3	CK DIV2	CK DIV1	CK DIV0		M_L_ SEL	CSPL	DOPH	DIPH
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register contains the SPI configuration B data specifically for chip select 3, MQSPI_CS3. Configuration of this register is identical to the CSCFG0B register. Refer to the CSCFG0B register definition, for definition of this register.

CSCFG3B SPI Chip Select 3 Configuration B Register **Addr** $2485_1038

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
			DAT_ CNT_ 6	DAT_ CNT_ 5	DAT_ CNT_ 4	DAT_ CNT_ 3	DAT_ CNT_ 2	DAT_ CNT_ 1	DAT_ CNT_ 0	DBC_ CNT_ 6	DBC_ CNT_ 5	DBC_ CNT_ 4	DBC_ CNT_ 3	DBC_ CNT_ 2	DBC_ CNT_ 1	DBC_ CNT_ 0
TYPE	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register contains the SPI configuration A data specifically for chip select 4, MQSPI_CS4. Configuration of this register is identical to the CSCFG0A register. Refer to the CSCFG0A register definition, for definition of this register.

CSCFG4A SPI Chip Select 4 Configuration A Register **Addr** $2485_103A

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
					CK DIV6	CK DIV5	CK DIV4	CK DIV3	CK DIV2	CK DIV1	CK DIV0		M_L_ SEL	CSPL	DOPH	DIPH
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register contains the SPI configuration B data specifically for chip select 4, MQSPI_CS4. Configuration of this register is identical to the CSCFG0B register. Refer to the CSCFG0B register definition, for definition of this register.

CSCFG4B SPI Chip Select 4 Configuration B Register **Addr** $2485_103C

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
			DAT_ CNT_ 6	DAT_ CNT_ 5	DAT_ CNT_ 4	DAT_ CNT_ 3	DAT_ CNT_ 2	DAT_ CNT_ 1	DAT_ CNT_ 0	DBC_ CNT_ 6	DBC_ CNT_ 5	DBC_ CNT_ 4	DBC_ CNT_ 3	DBC_ CNT_ 2	DBC_ CNT_ 1	DBC_ CNT_ 0
TYPE	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register contains the SPI configuration A data specifically for chip select 5, MQSPI_CS5. Configuration of this register is identical to the CSCFG0A register. Refer to the CSCFG0A register definition, for definition of this register.

CSCFG5A — SPI Chip Select 5 Configuration A Register — **Addr** $2485_103E

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
					CK DIV6	CK DIV5	CK DIV4	CK DIV3	CK DIV2	CK DIV1	CK DIV0		M_L_ SEL	CSPL	DOPH	DIPH
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register contains the SPI configuration B data specifically for chip select 5, MQSPI_CS5. Configuration of this register is identical to the CSCFG0B register. Refer to the CSCFG0B register definition, for definition of this register.

CSCFG5B — SPI Chip Select 5 Configuration B Register — **Addr** $2485_1040

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
			DAT_ CNT_ 6	DAT_ CNT_ 5	DAT_ CNT_ 4	DAT_ CNT_ 3	DAT_ CNT_ 2	DAT_ CNT_ 1	DAT_ CNT_ 0	DBC_ CNT_ 6	DBC_ CNT_ 5	DBC_ CNT_ 4	DBC_ CNT_ 3	DBC_ CNT_ 2	DBC_ CNT_ 1	DBC_ CNT_ 0
TYPE	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register contains the SPI configuration A data specifically for chip select 6, MQSPI_CS6. Configuration of this register is identical to the CSCFG0A register. Refer to the CSCFG0A register definition, for definition of this register.

CSCFG6A — SPI Chip Select 6 Configuration A Register — **Addr** $2485_1042

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
					CK DIV6	CK DIV5	CK DIV4	CK DIV3	CK DIV2	CK DIV1	CK DIV0		M_L_ SEL	CSPL	DOPH	DIPH
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register contains the SPI configuration B data specifically for chip select 6, MQSPI_CS6. Configuration of this register is identical to the CSCFG0B register. Refer to the CSCFG0B register definition, for definition of this register.

CSCFG6B — SPI Chip Select 6 Configuration B Register — **Addr** $2485_1044

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
			DAT_ CNT_ 6	DAT_ CNT_ 5	DAT_ CNT_ 4	DAT_ CNT_ 3	DAT_ CNT_ 2	DAT_ CNT_ 1	DAT_ CNT_ 0	DBC_ CNT_ 6	DBC_ CNT_ 5	DBC_ CNT_ 4	DBC_ CNT_ 3	DBC_ CNT_ 2	DBC_ CNT_ 1	DBC_ CNT_ 0
TYPE	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register contains the SPI configuration data A specifically for chip select 7, MQSPI_CS7. Configuration of this register is identical to the CSCFG0A register. Refer to the CSCFG0A register definition, for definition of this register.

CSCFG7A SPI Chip Select 7 Configuration A Register **Addr** $2485_1046

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
					CK DIV6	CK DIV5	CK DIV4	CK DIV3	CK DIV2	CK DIV1	CK DIV0		M_L_ SEL	CSPL	DOPH	DIPH
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register contains the SPI configuration B data specifically for chip select 7, MQSPI_CS7. Configuration of this register is identical to the CSCFG0B register. Refer to the CSCFG0B register definitions definition of this register.

CSCFG7B SPI Chip Select 7 Configuration B Register **Addr** $2485_1048

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
			DAT_ CNT_ 6	DAT_ CNT_ 5	DAT_ CNT_ 4	DAT_ CNT_ 3	DAT_ CNT_ 2	DAT_ CNT_ 1	DAT_ CNT_ 0	DBC_ CNT_ 6	DBC_ CNT_ 5	DBC_ CNT_ 4	DBC_ CNT_ 3	DBC_ CNT_ 2	DBC_ CNT_ 1	DBC_ CNT_ 0
TYPE	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register contains the SPI configuration data A specifically for chip select 8, MQSPI_CS8. Configuration of this register is identical to the CSCFG0A register. Refer to the CSCFG0A register definition, for definition of this register.

CSCFG8A SPI Chip Select 8 Configuration A Register **Addr** $2485_104A

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
					CK DIV6	CK DIV5	CK DIV4	CK DIV3	CK DIV2	CK DIV1	CK DIV0		M_L_ SEL	CSPL	DOPH	DIPH
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register contains the SPI configuration B data specifically for chip select 8, MQSPI_CS8. Configuration of this register is identical to the CSCFG0B register. Refer to the CSCFG0B register definition, for definition of this register.

CSCFG8B SPI Chip Select 8 Configuration B Register **Addr** $2485_104C

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
			DAT_ CNT_ 6	DAT_ CNT_ 5	DAT_ CNT_ 4	DAT_ CNT_ 3	DAT_ CNT_ 2	DAT_ CNT_ 1	DAT_ CNT_ 0	DBC_ CNT_ 6	DBC_ CNT_ 5	DBC_ CNT_ 4	DBC_ CNT_ 3	DBC_ CNT_ 2	DBC_ CNT_ 1	DBC_ CNT_ 0
TYPE	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register contains the SPI configuration data A specifically for chip select 9, MQSPI_CS9. Configuration of this register is identical to the CSCFG0A register. Refer to the CSCFG0A register definition, for definition of this register.

CSCFG9A — SPI Chip Select 9 Configuration A Register — **Addr** $2485_104E

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
					CK DIV6	CK DIV5	CK DIV4	CK DIV3	CK DIV2	CK DIV1	CK DIV0		M_L_ SEL	CSPL	DOPH	DIPH
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register contains the SPI configuration B data specifically for chip select 9, MQSPI_CS9. Configuration of this register is identical to the CSCFG0B register. Refer to the CSCFG0B register definition, for definition of this register.

CSCFG9B — SPI Chip Select 9 Configuration B Register — **Addr** $2485_1050

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
			DAT_ CNT_ 6	DAT_ CNT_ 5	DAT_ CNT_ 4	DAT_ CNT_ 3	DAT_ CNT_ 2	DAT_ CNT_ 1	DAT_ CNT_ 0	DBC_ CNT_ 6	DBC_ CNT_ 5	DBC_ CNT_ 4	DBC_ CNT_ 3	DBC_ CNT_ 2	DBC_ CNT_ 1	DBC_ CNT_ 0
TYPE	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register, in conjunction with the QST1B register, contains queue status data for SPI1. The QST1A register contains information on the number of messages and bytes written. This register is read-only.

QST1A — SPI1 Queue Status Register A — **Addr** $2485_1052

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	MW5	MW4	MW3	MW2	MW1	MW0	BW4	BW3	BW2	BW1	BW0	AT4	AT3	AT2	AT1	AT0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-33. QST1A Description

Name	Description	Settings
MW[5:0] Bit 15-10	Messages Written— MW indicates the number of messages written during the current SPI1 transfer. This information would normally only be used for multiple message transfers, but not burst mode transfers.	000000 = 1 message transferred … 111111 = 64 messages transferred
BW[4:0] Bits 9-5	**Bytes Written—** BW indicates the number of bytes written during the current SPI1 transfer.	00000 = 1 byte written … 11111 = 32 bytes written
AT[4:0] Bits 4-0	**Active Trigger—** AT indicates the current active trigger of the 32 available triggers during the current SPI1 transfer.	00000 = Trigger 0 … 11111 = Trigger 31

This register, in conjunction with the QST1A register, contains queue status data for SPI1. The QST1B register contains high priority and low priority FIFO queue pointers for SPI1.

QST1B SPI1 Queue Status Register B **Addr** $2485_1**054**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
			HALT ACK	APQ	HP BOQP 2	HP BOQP 1	HP BOQP 0	HP EOQP 2	HP EOQP 1	HP EOQP 0	LP BOQP 2	LP BOQP 1	LP BOQP 0	LP EOQP 2	LP EOQP 1	LP EOQP 0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-34. QST1B Description

Name	Description	Settings
Bits 15-14	Reserved	N/A
HALT_ACK Bit 13	Halt Acknowledge - Indicates the SPI has halted operation due to entry into Wait or Doze low power mode after completing transfers in progress. It will be cleared when the chip exits the low power mode. **Note:** HALT_ACK is only visible in certain test modes since the CPU is inactive during the low power modes when this bit is set.	0 - SPI is operating normally 1 - SPI is inactive due to Wait or Doze Low Power modes
APQ Bit 12	Active Priority Queue— The APQ identifies whether the active triggered queue is from the high or low priority FIFO1 queue.	0 = Active triggered control queue is from the low priority FIFO1 queue 1 = Active triggered control queue is from the high priority FIFO1 queue
HPBOQP[2:0] Bits 11-9	**Beginning of High Priority FIFO Queue—** This value points to the next High Priority FIFO1 Queue to be read.	000 ... 111
HPEOQP[2:0] Bits 8-6	End of High Priority FIFO Queue— This value points to the next High Priority FIFO1 Queue to be written to.	000 ... 111
LPBOQP[2:0] Bits 5-3	Beginning of Low Priority FIFO Queue— This value points to the next Low Priority FIFO1 Queue to be read.	000 ... 111
LPEOQP[2:0] Bits 2-0	End of Low Priority FIFO Queue— This value points to the next Low Priority FIFO1 Queue to be written to.	000 ... 111

This register, in conjunction with the QST2B register, contains queue status data for SPI2. The QST2A register contains information on the number of messages and bytes written.

QST2A SPI2 Queue Status Register A **Addr** $2485_1056

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	MW5	MW4	MW3	MW2	MW1	MW0	BW4	BW3	BW2	BW1	BW0	AT4	AT3	AT2	AT1	AT0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-35. QST2A Description

Name	Description	Settings
MW[5:0] Bit 15-10	Messages Written— MW indicates the number of messages written during the current SPI2 transfer. This information would normally only be used for multiple message transfers, but not burst mode transfers.	000000 = 1 message transferred … 111111 = 64 messages transferred
BW[4:0] Bits 9-5	**Bytes Written—** BW indicates the number of bytes written during the current SPI2 transfer.	00000 = 1 byte written … 11111 = 32 bytes written
AT[4:0] Bits 4-0	**Active Trigger—** AT indicates the current active trigger of the 32 available triggers during the current SPI2 transfer. 00000 = Trigger 0 … 11111 = Trigger 31	00000 = Trigger 0 … 11111 = Trigger 31

This register, in conjunction with the QST2A register, contains queue status data for SPI2. The QST2B register contains high priority and low priority FIFO queue pointers for SPI2.

QST2B SPI2 Queue Status Register B **Addr** $2485_1058

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
			HALT ACK	APQ	HP BOQP 2	HP BOQP 1	HP BOQP 0	HP EOQP 2	HP EOQP 1	HP EOQP 0	LP BOQP 2	LP BOQP 1	LP BOQP 0	LP EOQP 2	LP EOQP 1	LP EOQP 0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-36. QST2B Description

Name	Description	Settings
Bits 51-13	Reserved	N/A

Table 49-36. QST2B Description

Name	Description	Settings
HALT_ACK Bit 13	Halt Acknowledge - Indicates the SPI has halted operation due to entry into Wait or Doze low power mode after completing transfers in progress. It will be cleared when the chip exits the low power mode. **Note:** HALT_ACK is only visible in certain test modes since the CPU is inactive during the low power modes when this bit is set.	0 - SPI is operating normally 1 - SPI is inactive due to Wait or Doze Low Power modes
APQ Bit 12	Active Priority Queue— The APQ identifies whether the active triggered queue is from the high or low priority FIFO2 queue.	0 = Active triggered control queue is from the low priority FIFO2 queue 1 = Active triggered control queue is from the high priority FIFO2 queue
HPBOQP[2:0] Bits 11-9	**Beginning of High Priority FIFO Queue—** This value points to the next High Priority FIFO2 Queue to be read.	000 ... 111
HPEOQP[2:0] Bits 8-6	End of High Priority FIFO Queue— This value points to the next High Priority FIFO2 Queue to be written to.	000 ... 111
LPBOQP[2:0] Bits 5-3	Beginning of Low Priority FIFO Queue— This value points to the next Low Priority FIFO2 Queue to be read.	000 ... 111
LPEOQP[2:0] Bits 2-0	**End of Low Priority FIFO Queue—** This value points to the next Low Priority FIFO2 Queue to be written to.	000 ... 111

This register contains the status of the SPI1 and SPI2.

SST SPI Status Register **Addr** $2485_105A

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
			LT1 4	LT1 3	LT1 2	LT1 1	LT1 0	LT2 4	LT2 3	LT2 2	LT2 1	LT2 0	SPIB1	SPIB2	LPQ1	LPQ2
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-37. SST Description

Name	Description	Settings
Bits 15-14	Reserved	N/A
LT1[4:0] Bits 13-9	**Last Trigger on SPI1—** LT1 identifies the last triggered control queue on SPI1.	00000 = Last trigger on SPI1 is Trigger 0 … 11111 = Last trigger on SPI1 is Trigger 31
LT2[4:0] Bits 8-4	**Last Trigger on SPI2—** LT2 identifies the last triggered control queue on SPI2.	00000 = Last trigger on SPI2 is Trigger 0 … 11111 = Last trigger on SPI2 is Trigger 31
SPIB1 Bit 3	**SPI1 Busy—** SPIB1 indicates that SPI1 is currently in use.	0 = SPI1 not in use 1 = SPI1 in use.
SPIB2 Bit 2	**SPI2 Busy —**SPIB2 indicates that SPI2 is currently in use.	0 = SPI2 not in use 1 = SPI2 in use.
LPQ1 Bit 1	**Last Priority Queue for SPI1—**LPQ1 indicates whether the last priority FIFO queue was the low or high priority FIFO queue for SPI1.	0 = Low priority FIFO queue was used last for SPI1 1 = High priority FIFO queue was used last for SPI1
LPQ2 Bit 0	**Last Priority Queue for SPI2—**LPQ2 indicates whether the last priority FIFO queue was the low or high priority FIFO queue for SPI2.	0 = Low priority FIFO queue was used last for SPI2 1 = High priority FIFO queue was used last for SPI2

This register, in conjunction with the SHPFQ1B and SHPFQ1C registers, contains the contents of the first 3 FIFO queue buffers for the high priority FIFO queue for SPI1.

SHPFQ1A SPI1 High Priority FIFO Queue A Register **Addr** $2485_105C

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
		FQ2 4	FQ2 3	FQ2 2	FQ2 1	FQ2 0	FQ1 4	FQ1 3	FQ1 2	FQ1 1	FQ1 0	FQ0 4	FQ0 3	FQ0 2	FQ0 1	FQ0 0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-38. SHPFQ1A Description

Name	Description	Settings
Bit 15	Reserved	N/A
FQ2[4:0] Bits 14-10	Contents of High Priority FIFO1 Queue 2	N/A
FQ1[4:0] Bits 9-5	Contents of High Priority FIFO1 Queue 1	N/A
FQ0[4:0] Bits 4-0	Contents of High Priority FIFO1 Queue 0	N/A

This register, in conjunction with the SHPFQ1A and SHPFQ1C registers, contains the contents of the second 3 FIFO queue buffers for the high priority FIFO queue for SPI1.

SHPFQ1B SPI1 High Priority FIFO Queue B Register **Addr** $2485_105E

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
		FQ54	FQ53	FQ52	FQ51	FQ50	FQ44	FQ43	FQ42	FQ41	FQ40	FQ34	FQ33	FQ32	FQ31	FQ30
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-39. SHPFQ1B Description

Name	Description	Settings
Bit 15	Reserved	N/A
FQ5[4:0] Bits 14-10	Contents of High Priority FIFO1 Queue 5	N/A
FQ4[4:0] Bits 9-5	Contents of High Priority FIFO1 Queue 4	N/A
FQ3[4:0] Bits 4-0	Contents of High Priority FIFO1 Queue 3	N/A

his register, in conjunction with the SHPFQ1A and SHPFQ1B registers, contains the contents of the last 2 FIFO queue buffers for the high priority FIFO queue for SPI1.

SHPFQ1C SPI1 High Priority FIFO Queue C Register **Addr** $2485_1060

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
							FQ74	FQ73	FQ72	FQ71	FQ70	FQ64	FQ63	FQ62	FQ61	FQ60
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-40. SHPFQ1C Description

Name	Description	Settings
Bits 15-10	Reserved	N/A
FQ7[4:0] Bits 9-5	Contents of High Priority FIFO1 Queue 7	N/A
FQ6[4:0] Bits 4-0	Contents of High Priority FIFO1 Queue 6	N/A

This register, in conjunction with the SLPFQ1B and SLPFQ1C registers, contains the contents of the first 3 FIFO queue buffers for the low priority FIFO queue for SPI1.

SLPFQ1A SPI1 Low Priority FIFO Queue A Register **Addr** $2485_1062

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
		FQ2 4	FQ2 3	FQ2 2	FQ2 1	FQ2 0	FQ1 4	FQ1 3	FQ1 2	FQ1 1	FQ1 0	FQ0 4	FQ0 3	FQ0 2	FQ0 1	FQ0 0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-41. SLPFQ1A Description

Name	Description	Settings
Bit 15	Reserved	N/A
FQ2[4:0] Bits 14-10	Contents of Low Priority FIFO1 Queue 2	N/A
FQ1[4:0] Bits 9-5	Contents of Low Priority FIFO1 Queue 1	N/A
FQ0[4:0] Bits 4-0	Contents of Low Priority FIFO1 Queue 0	N/A

This register, in conjunction with the SLPFQ1A and SLPFQ1C registers, contains the contents of the second 3 FIFO queue buffers for the low priority FIFO queue for SPI1.

SLPFQ1B SPI1 Low Priority FIFO Queue B Register **Addr** $2485_1**064**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
		FQ54	FQ53	FQ52	FQ51	FQ50	FQ44	FQ43	FQ42	FQ41	FQ40	FQ34	FQ33	FQ32	FQ31	FQ30
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-42. SLPFQ1B Description

Name	Description	Settings
Bit 15	Reserved	N/A
FQ5[4:0] Bits 14-10	Contents of Low Priority FIFO1 Queue 5	N/A
FQ4[4:0] Bits 9-5	Contents of Low Priority FIFO1 Queue 4	N/A
FQ3[4:0] Bits 4-0	Contents of Low Priority FIFO1 Queue 3	N/A

This register, in conjunction with the SLPFQ1A and SLPFQ1B registers, contains the contents of the last 2 FIFO queue buffers for the low priority FIFO queue for SPI1.

SLPFQ1C SPI1 Low Priority FIFO Queue C Register **Addr** $2485_1066

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
							FQ74	FQ73	FQ72	FQ71	FQ70	FQ64	FQ63	FQ62	FQ61	FQ60
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-43. SLPFQ1C Description

Name	Description	Settings
Bits 15-10	Reserved	N/A
FQ7[4:0] Bits 9-5	Contents of Low Priority FIFO1 Queue 7	N/A
FQ6[4:0] Bits 4-0	Contents of Low Priority FIFO1 Queue 6	N/A

This register, in conjunction with the SHPFQ2B and SHPFQ2C registers, contains the contents of the first 3 FIFO queue buffers for the high priority FIFO queue for SPI2.

SHPFQ2A SPI2 High Priority FIFO Queue A Register **Addr** $2485_1068

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
		FQ2 4	FQ2 3	FQ2 2	FQ2 1	FQ2 0	FQ1 4	FQ1 3	FQ1 2	FQ1 1	FQ1 0	FQ0 4	FQ0 3	FQ0 2	FQ0 1	FQ0 0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-44. SHPFQ2A Description

Name	Description	Settings
Bit 15	Reserved	N/A
FQ2[4:0] Bits 14-10	Contents of High Priority FIFO1 Queue 2	N/A
FQ1[4:0] Bits 9-5	Contents of High Priority FIFO1 Queue 1	N/A
FQ0[4:0] Bits 4-0	Contents of High Priority FIFO1 Queue 0	N/A

This register, in conjunction with the SHPFQ2A and SHPFQ2C registers, contains the contents of the second 3 FIFO queue buffers for the high priority FIFO queue for SPI2.

SHPFQ2B SPI2 High Priority FIFO Queue B Register **Addr** $2485_106A

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
		FQ54	FQ53	FQ52	FQ51	FQ50	FQ44	FQ43	FQ42	FQ41	FQ40	FQ34	FQ33	FQ32	FQ31	FQ30
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-45. SHPFQ2B Description

Name	Description	Settings
Bit 15	Reserved	N/A
FQ5[4:0] Bits 14-10	Contents of High Priority FIFO2 Queue 5	N/A
FQ4[4:0] Bits 9-5	Contents of High Priority FIFO2 Queue 4	N/A
FQ3[4:0] Bits 4-0	Contents of High Priority FIFO2 Queue 3	N/A

This register, in conjunction with the SHPFQ2A and SHPFQ2B registers, contains the contents of the last 2 FIFO queue buffers for the high priority FIFO queue for SPI2.

SHPFQ2C SPI2 High Priority FIFO Queue C Register **Addr** $2485_106C

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
							FQ74	FQ73	FQ72	FQ71	FQ70	FQ64	FQ63	FQ62	FQ61	FQ60
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-46. SHPFQ1C Description

Name	Description	Settings
Bits 15-10	Reserved	N/A
FQ7[4:0] Bits 9-5	Contents of High Priority FIFO2 Queue 7	N/A
FQ6[4:0] Bits 4-0	Contents of High Priority FIFO2 Queue 6	N/A

This register, in conjunction with the SLPFQ2B and SLPFQ2C registers, contains the contents of the first 3 FIFO queue buffers for the low priority FIFO queue for SPI2.

SLPFQ2A SPI2 Low Priority FIFO Queue A Register **Addr** $2485_106E

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
		FQ2 4	FQ2 3	FQ2 2	FQ2 1	FQ2 0	FQ1 4	FQ1 3	FQ1 2	FQ1 1	FQ1 0	FQ0 4	FQ0 3	FQ0 2	FQ0 1	FQ0 0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-47. SLPFQ2A Description

Name	Description	Settings
Bit 15	Reserved	N/A
FQ2[4:0] Bits 14-10	Contents of Low Priority FIFO2 Queue 2	N/A
FQ1[4:0] Bits 9-5	Contents of Low Priority FIFO2 Queue 1	N/A
FQ0[4:0] Bits 4-0	Contents of Low Priority FIFO2 Queue 0	N/A

This register, in conjunction with the SLPFQ2A and SLPFQ2C registers, contains the contents of the second 3 FIFO queue buffers for the low priority FIFO queue for SPI2.

SLPFQ2B SPI2 Low Priority FIFO Queue B Register **Addr** $2485_1070

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
		FQ54	FQ53	FQ52	FQ51	FQ50	FQ44	FQ43	FQ42	FQ41	FQ40	FQ34	FQ33	FQ32	FQ31	FQ30
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-48. SLPFQ2B Description

Name	Description	Settings
Bit 15	Reserved	N/A
FQ5[4:0] Bits 14-10	Contents of Low Priority FIFO2 Queue 2	N/A
FQ4[4:0] Bits 9-5	Contents of Low Priority FIFO2 Queue 1	N/A
FQ3[4:0] Bits 4-0	Contents of Low Priority FIFO2 Queue 0	N/A

This register, in conjunction with the SLPFQ2A and SLPFQ2B registers, contains the contents of the last 2 FIFO queue buffers for the low priority FIFO queue for SPI2.

SLPFQ2C SPI2 Low Priority FIFO Queue C Register **Addr** $2485_1072

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
							FQ74	FQ73	FQ72	FQ71	FQ70	FQ64	FQ63	FQ62	FQ61	FQ60
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-49. SLPFQ2C Description

Name	Description	Settings
Bits 15-10	Reserved	N/A
FQ7[4:0] Bits 9-5	Contents of Low Priority FIFO2 Queue 7	N/A
FQ6[4:0] Bits 4-0	Contents of Low Priority FIFO2 Queue 6	N/A

This register, in conjunction with the BCDM0S1 register, contains the contents of the current or last state of the Trigger 0 Burst Command Data Pointer Control when SPI1 is used. This register is primarily used for test purposes and only for burst mode operation. This register is updated as each message in the queue is individually triggered. The contents of the BCDP0S1 register indicates active information of an active burst mode operation when the Burst Mode Active (BMA) bit in the BCDM0S1 BMAS register is set.

BCDP0S1 SPI1 Burst Cmd Data Pointer Status Register 0 **Addr** $2485_1**074**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QWSTR 7	QWSTR 6	QWSTR 5	QWSTR 4	QWSTR 3	QWSTR 2	QWSTR 1	QWSTR 0	QRSTR 7	QRSTR 6	QRSTR 5	QRSTR 4	QRSTR 3	QRSTR 2	QRSTR 1	QRSTR 0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-50. BCDP0S1 Description

Name	Description	Settings
QWSTR[7:0] Bits 15-8	**Starting Location of the Next Write Transfer Data—** QWSTR contains the starting location in RAM for the **next** message to be written out of the SPI. Note: Because the internal RAM address differs from the external address, an adjustment must be made to the address before it is stored in QWSTR. The adjustment is: QWSTR = (DStart - $480)/2, where: DStart = Offset Starting address of data to be transferred (written) out of the SPI, $480 = Offset Address of the beginning of the SPI I/O Data RAM.	QWSTR = (DStart - $480)/2
QRSTR[7:0] Bits 7-0	**Starting Location of Next Read Transfer Data—** QRSTR contains the starting location in RAM for the **next** message to be read and stored from the SPI. Note: Because the internal RAM address differs from the external address, an adjustment must be made to the address before it is stored in QRSTR. The adjustment is: QRSTR = (DStart - $480)/2, where: DStart = Offset Starting address of data to be stored from the SPI read transfer, $480 = Offset Address of the beginning of the SPI I/O Data RAM.	QRSTR = (DStart - $480)/2

This register, in conjunction with the BCDP0S1 register, contains the contents of the current or last state of the Trigger 0 Burst Command Data Mode Control when SPI1 is used. This register is primarily used for test purposes and only for burst mode operation. At the beginning of a SPI burst mode transfer, the contents of this register is identical to the trigger 0 Mode Control data in the Control Data RAM. This register is then updated as each message in the queue is individually triggered. The contents of the BCDM0S1 register indicates active information of an active burst mode operation when the Burst Mode Active (BMA) bit is set in the BMAS register.

BCDM0S1 SPI1 Burst Cmd Data Mode Status Register 0 **Addr** $2485_1076

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QBRST	QRC 5	QRC 4	QRC 3	QRC 2	QRC 1	QRC 0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-51. BCDM0S1 Description

Name	Description	Settings
QBRST Bits 15	**Queue Burst Mode Enable—**The QBRST indicates burst mode operation.	0 = Burst mode disabled 1 = Burst mode enabled
QRC[5:0] Bits 14-9	**Queue Multiple Message Repeat Count—**QRC indicates the number of messages remaining to be transferred.	000000 = 1 message remaining to be transferred (provided that the BMA bit is set) … 111111 = 64 messages remaining to be transferred
QBL[4:0] Bits 8-4	**Queue Message Data Byte Length—** QBL indicates the number of whole bytes to be transferred per message.	00000 = 1 byte message … 11111 = 32 byte message
QCS[3:0] Bits 3-0	**Queue Chip Select —** QCS chooses which chip select pin to use for the SPI transfer.	0000 = SPI_CS[0] … 1001 = SPI_CS[9]

This register, in conjunction with the BCDM1S1 register, contains the contents of the current or last state of the Trigger 1 Burst Command Data Pointer Control when SPI1 is used. This register is identical in operation as BCDP0S1 as is defined in its section.

BCDP1S1 SPI1 Burst Cmd Data Pointer Status Register 1 **Addr** $2485_1078

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QWSTR7	QWSTR6	QWSTR5	QWSTR4	QWSTR3	QWSTR2	QWSTR1	QWSTR0	QRSTR7	QRSTR6	QRSTR5	QRSTR4	QRSTR3	QRSTR2	QRSTR1	QRSTR0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register, in conjunction with the BCDP0S1 register, contains the contents of the current or last state of the Trigger 1 Burst Command Data Mode Control when SPI1 is used. This register is identical in operation as BCDM0S1 as is defined in its section.

BCDM1S1 SPI1 Burst Cmd Data Mode Status Register 1 **Addr** $2485_107A

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QBURST	QRC5	QRC4	QRC3	QRC2	QRC1	QRC0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register, in conjunction with the BCDM2S1 register, contains the contents of the current or last state of the Trigger 2 Burst Command Data Pointer Control when SPI1 is used. This register is identical in operation as BCDP0S1 as is defined in its section.

BCDP2S1 SPI1 Burst Cmd Data Pointer Status Register 2 **Addr** $2485_107C

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QWSTR7	QWSTR6	QWSTR5	QWSTR4	QWSTR3	QWSTR2	QWSTR1	QWSTR0	QRSTR7	QRSTR6	QRSTR5	QRSTR4	QRSTR3	QRSTR2	QRSTR1	QRSTR0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register, in conjunction with the BCDP2S1 register, contains the contents of the current or last state of the Trigger 2 Burst Command Data Mode Control when SPI1 is used. This register is identical in operation as BCDM0S1 as is defined in its section.

BCDM2S1 SPI1 Burst Cmd Data Mode Status Register 2 **Addr** $2485_107E

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QBURST	QRC5	QRC4	QRC3	QRC2	QRC1	QRC0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register, in conjunction with the BCDM3S1 register, contains the contents of the current or last state of the Trigger 3 Burst Command Data Pointer Control when SPI1 is used. This register is identical in operation as BCDP0S1 as is defined in its section.

BCDP3S1 SPI1 Burst Cmd Data Pointer Status Register 3 **Addr** $2485_1080

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QWSTR7	QWSTR6	QWSTR5	QWSTR4	QWSTR3	QWSTR2	QWSTR1	QWSTR0	QRSTR7	QRSTR6	QRSTR5	QRSTR4	QRSTR3	QRSTR2	QRSTR1	QRSTR0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register, in conjunction with the BCDP3S1 register, contains the contents of the current or last state of the Trigger 3 Burst Command Data Mode Control when SPI1 is used. This register is identical in operation as BCDM0S1 as is defined in its section.

BCDM3S1 SPI1 Burst Cmd Data Mode Status Register 3 **Addr** $2485_1082

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QBURST	QRC5	QRC4	QRC3	QRC2	QRC1	QRC0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register, in conjunction with the BCDM0S2 register, contains the contents of the current or last state of the Trigger 0 Burst Command Data Pointer Control when SPI2 is used. This register is identical in operation as BCDP0S1 as is defined in its section.

BCDP0S2 SPI2 Burst Cmd Data Pointer Status Register 0 **Addr** $2485_1084

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QWSTR7	QWSTR6	QWSTR5	QWSTR4	QWSTR3	QWSTR2	QWSTR1	QWSTR0	QRSTR7	QRSTR6	QRSTR5	QRSTR4	QRSTR3	QRSTR2	QRSTR1	QRSTR0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register, in conjunction with the BCDP0S2 register, contains the contents of the current or last state of the Trigger 0 Burst Command Data Mode Control when SPI2 is used. This register is identical in operation as BCDM0S1 as is defined in its section.

BCDM0S2 SPI2 Burst Cmd Data Mode Status Register 0 **Addr** $2485_1086

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QBURST	QRC5	QRC4	QRC3	QRC2	QRC1	QRC0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register, in conjunction with the BCDM1S2 register, contains the contents of the current or last state of the Trigger 1 Burst Command Data Pointer Control when SPI2 is used. This register is identical in operation as BCDP0S1 as is defined in its section.

BCDP1S2 SPI2 Burst Cmd Data Pointer Status Register 1 **Addr** $2485_1088

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QW STR 7	QW STR 6	QW STR 5	QW STR 4	QW STR 3	QW STR 2	QW STR 1	QW STR 0	QR STR 7	QR STR 6	QR STR 5	QR STR 4	QR STR 3	QR STR 2	QR STR 1	QR STR 0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register, in conjunction with the BCDP0S2 register, contains the contents of the current or last state of the Trigger 0 Burst Command Data Mode Control when SPI2 is used. This register is identical in operation as BCDM0S1 as is defined in its section.

BCDM1S2 SPI2 Burst Cmd Data Mode Status Register 1 **Addr** $2485_108A

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QBURST	QRC 5	QRC 4	QRC 3	QRC 2	QRC 1	QRC 0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register, in conjunction with the BCDM2S2 register, contains the contents of the current or last state of the Trigger 2 Burst Command Data Pointer Control when SPI2 is used. This register is identical in operation as BCDP0S1 as is defined in its section.

BCDP2S2 SPI2 Burst Cmd Data Pointer Status Register 2 **Addr** $2485_108C

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QW STR 7	QW STR 6	QW STR 5	QW STR 4	QW STR 3	QW STR 2	QW STR 1	QW STR 0	QR STR 7	QR STR 6	QR STR 5	QR STR 4	QR STR 3	QR STR 2	QR STR 1	QR STR 0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register, in conjunction with the BCDP2S2 register, contains the contents of the current or last state of the Trigger 2 Burst Command Data Mode Control when SPI2 is used. This register is identical in operation as BCDM0S1 as is defined in its section.

BCDM2S2 SPI2 Burst Cmd Data Mode Status Register 2 **Addr** $2485_108E

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QBURST	QRC 5	QRC 4	QRC 3	QRC 2	QRC 1	QRC 0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register, in conjunction with the BCDM3S2 register, contains the contents of the current or last state of the Trigger 3 Burst Command Data Pointer Control when SPI2 is used. This register is identical in operation as BCDP0S1 as is defined in its section.

BCDP3S2 SPI2 Burst Cmd Data Pointer Status Register 3 **Addr** $2485_1**090**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QW STR 7	QW STR 6	QW STR 5	QW STR 4	QW STR 3	QW STR 2	QW STR 1	QW STR 0	QR STR 7	QR STR 6	QR STR 5	QR STR 4	QR STR 3	QR STR 2	QR STR 1	QR STR 0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register, in conjunction with the BCDP3S2 register, contains the contents of the current or last state of the Trigger 3 Burst Command Data Mode Control when SPI2 is used. This register is identical in operation as BCDM0S1 as is defined in its section.

BCDM3S2 SPI2 Burst Cmd Data Mode Status Register 3 **Addr** $2485_1**092**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QBURST	QRC 5	QRC 4	QRC 3	QRC 2	QRC 1	QRC 0	QBL4	QBL3	QBL2	QBL1	QBL0	QCS3	QCS2	QCS1	QCS0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register defines whether burst triggers 0 through 3, for SPI1 or SPI2, are active.

BMAS Burst Mode Active Status Register **Addr** $2485_1**094**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									BMA_2_0	BMA_2_1	BMA_2_2	BMA_2_3	BMA_1_0	BMA_1_1	BMA_1_2	BMA_1_3
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49-52. BMAS Description

Name	Description	Settings
Bits 15-8	Reserved	N/A
BMA Bits 7-0	Burst Mode Active— The BMA bit indicates whether the control queue is actively using the burst mode operation. There is a status bit for each of Queues 0-3 for both SPI's. Burst mode may only be active for one SPI at any given time.	0 = Burst mode is not in use. 1 = Burst mode is in use.

49.3.14.4 Interrupt Handling

Two interrupts are provided by the MQSPI, one for each SPI. There is one register for enabling interrupts and one register for determining the source of the interrupt. Interrupts are enabled using the SPI Mask register (SPIM). Pending interrupts are read from the SPI Flag register (SFLG). Although there are two interrupts provided by the MQSPI, there is only one mask and one flag register that are shared between the two SPIs.

A SPI transferred finish interrupt can be programmed for any of the 32 triggers. They are enabled using the SPI Transfer Finished Enable registers (STFE_H and STFE_L). In addition, the SPI Transfer Finished (STF) mask bit must be enabled in the SPIM register. When a SPI transferred finished interrupt occurs, the SPI Transfer Finished Flag registers (STFF_H and STFF_L) must be polled to determine the source.

The sequence of events to process an interrupt is as follows:

1. The CP software reads the interrupt status register in the ITC module to determine the module that is sourcing the interrupt.
2. When the source is either SPI1 or SPI2, the CP software reads the SFLG register in the MQSPI module. The flags that are set are cleared by writing a ‘1’ to the corresponding bit in the SFLG register.
3. If the SFLG register indicates a transfer finished flag interrupt, the CP software reads the STFF_H and STFF_L registers to determine which SPI trigger(s) had completed its transfer. The trigger(s) that indicate a completed transfer are cleared by writing a 1 to the corresponding bit in the STFF_H and/or STFF_L registers.

When the CP software clears the interrupt flags in the SFLG and STFF_H/STFF_L registers, the interrupt line (INT or FINT) is deactivated provided that there are no more pending interrupts.

Chapter 50
Layer 1 Timer (L1T)

Revision History Table

Revision	Date	Author	Changes
1.0	10/8/99		Initial Release.
1.1	4/10/2000	Steve Olsen	Clarified the Master Enable bit function. Removed EER section and registers. Corrected reversal of PME and MCCE bits in the ES2 register. Updated the QBCO Handler section. Corrected reset values of RQBC, MQBC, TB_PRE, TRQBC, and TRAFNM_L registers. Updated the Pin Control Unit section. Corrected the bit ordering of the GTS1 register.
1.2	9/25/2000	Scott King	Mods for reuse in Neptune: -tri-state TOUT control not used -omitted AAM references
1.8	1/18/2001	Steve Olsen	Clarifications for Neptune: -added notes to appendix -corrected PTDLV description -added notes concerning FFI -updated port description -changed MCU base address
1.9	02/08/2001	Dan Chlopek Steven Olsen	Changes and updates related to DDTS defects: DSPhl10433, DSPhl10437, DSPhl10438, DSPhl10439, DSPhl10440, DSPhl10441, DSPhl10442, DSPhl10443, DSPhl10444, DSPhl10445, TLSbo08543, TLSbo08544, DSPhl10462
1.91	02/19/01	Kathy Flories Shannon Osgood	Spell checked. Checked/Updated address variables 2/22.

Revision History Table

Revision	Date	Author	Changes
1.92	03/09/2001	Dan Chlopek	-Updated DSP MQBS register reset value (DDTS DSPhl10943) -Updated the duty cycle for RCLOCK and CCLOCK on page 50-7 (DDTS DSPhl10642)
1.93	03/15/2001	Dan Chlopek	Changes and updates related to DDTS defects: DSPhl11112, DSPhl11115, DSPhl11071, DSPhl11109, DSPhl11110
2.0	03/28/2001	Dan Chlopek	-Updated description of DSP registers as per DDTS DSPhl11253 -Updated descriptions of TRFC and TCFC registers as per DDTS DSPhl11203 -Changed ckih input description to reference PAT_REF clock and added footnote on use of CKIH in this chapter as per DDTS DSPhl11355 -Ran through spell check
2.1	04/17/2001	Dan Chlopek	-Updated descriptions of the PCFG, PD, PS, and PTC registers -Updated DSP side QBCO description to clarify the delay
2.2	08/15/01	Shannon Osgood	Verified pin names against stub
2.3	12/17/01	Shannon Osgood	Update per DDTS# DSPhl12787
2.4	02/11/02	Shannon Osgood	Update per DDTS# DSPhl12785/DSPhl12786
2.5	02/25/02	Chris Daniels	Updated for Neptune LTS
2.5	05/06/02	Chris Daniels	Added DSP_EG_MD register
2.6	07/19/02	Chris Daniels	Corrected Registers and TBD
2.7	05/07/03		Updated for LTE specification release.

The Layer 1 (L1) Timer supports and executes all GSM, IS-136 TDMA, iDEN, and Iridium protocol related timing, off-loading Layer 1 scheduling tasks from the CPU. Key features include:

- 9-bit programmable prescaler.
- 16-bit programmable modulus Reference Quarter Bit Count (RQBC) counter.
- 32-bit programmable modulus Reference Absolute Frame Number (RAFN) counter.
- 16-bit programmable Quarter Bit Count Offset (QBCO).
- Synchronous, asynchronous, and event code synchronization of the QBCO.
- Multiple methods for updating timebase offset (QBCO)
- Timebase visible to DSP
- 192 entry Event Table
- Two relocatable Frame Tables
- Three relocatable Macro Tables
- One relocatable Parameter Table
- 16 general purpose MCU interrupts
- RX, TX and 15 general purpose vectored interrupts to DSP
- Frame and/or Macro Table event codes for:
  - Initiating Macro Tables.
  - Triggering MQSPI transfers.
  - Repeating (looping) event codes.
  - Programmable delays.
  - Toggling output pins.
  - Generating interrupts to the DSP or MCU
  - DSP Wakeup from STOP
- “Catch Up” mode for Frame Table event execution
- Frame/Macro Table error detection.
- Periodic Interrupt Timer (PIT).

All system events are synchronized to the system clock. The L1 timer measures time in fractional symbol counts, which are referred to as Quarter Bit Counts (GSM term). The QBCS are obtained by dividing the system clock by a programmable prescaler. The number of QBCS per frame varies with each of the different radio protocols, i.e. GSM, TDMA, iDEN and Iridium.

The core of the L1 timer is the dynamically programmable 192 entry Event Table. Each entry in the Event Table RAM block contains two fields: an 8-bit Event Code (EC) and a 16-bit Event Quarter Bit Count (EQBC). The EC specifies what action (event) the L1 timer is to execute. The EQBC field specifies the event's execution time relative to the start of the channel frame.

The Event Table can support two frame tables, three macro tables, and a parameter table. These tables provide flexibility in scheduling and executing events. The Frame table locations are dynamically configurable via the Frame Table Configuration Registers. Each Frame Table Configuration Register consists of a START pointer to define the first table entry, and an END pointer to define the last table entry. The NEXT FRAME TABLE bits are stored in the Next Frame Table Configuration Register. Events can be scheduled to execute in a single frame or over multiple frames. Frame tables can operate in two modes. In alternating mode, table completion causes the active and inactive tables to be switched. In repeat mode, the active table is repeated.

Macros tables can be used to define a fixed sequence of events for an RX or TX burst or for use as a signal strength measurement. Macro tables are dynamically configurable with START and END pointers defining macro tables A, B and C. The ability to use up to three macro tables minimizes the software interaction required with the L1 timer. The timer is designed so that under normal conditions, macro tables and parameter tables do not need to be reprogrammed once they have been set up. Calling a macro loads a parameter value from the parameter table.

The parameter table is defined by START and END pointers. Parameter values can be used as either delay values or loop number values. By calling a macro table with different parameter values used as delay values, the user can vary the pulse width of a signal. Macros can also use programmable delay values from delay registers to provide a variable offset from a reference event. Macro tables carry out events in sequential order. If a sequence of events is to be repeated, the sequence of events must be repeated using more event codes or the macro would be have to be called multiple times. To allow a sequence of events to be repeated in a clear and efficient manner and with much flexibility without rebuilding a macro table, a loop structure can be implemented for each of the macros. These loop count values are defined with either value from the parameter table or loop count registers.

The L1 Timer drives 16 chip external pins for Layer 1 device control. Thirty-two triggers are provided for Multiple Queued Serial Port Interface (MQSPI) control. The DSP core is capable of overriding timer control pins via the DTX_SELECT port. The Deep Sleep Module (DSM) can shut-down the timer via the DIS_L1T pin.

The L1 Timer supports various interrupts for both the MCU and the DSP. On the MCU side, the interrupts are:

RFI - Reference Frame Interrupt

RFCI - Reference Frame Counter Interrupt

CI - Channel Interrupt. Combination of Channel Frame Interrupt (CFI) and Channel Frame Counter Interrupt (CFCI)

FFI[1] - Combination of RFCI and RFI

MCU_INT0 - Combination of Programmable Interrupt Timer Interrupt (PITI), Programmable Wakeup Timer Interrupt (PWTI), and Timer Error Interrupt (TERI)

MCU_INT1 - Combination of interrupts generated by the MCU Interrupt Instructions

On the DSP side, the vectored interrupts are

L1RX - Layer 1 Receive Interrupt - tied to transitions of the PD0 pin

L1TX - Layer 1 Transmit Interrupt - tied to transitions of the PD1 pin

DVI0-DVI14 - DSP Vectored Interrupt instructions

DSPWAKE[2] - Wakeup interrupt to DSP

1. FFI output is not connected in the Neptune IC.
2. DSPWAKE output pulse width is 2 CKIH cycles.

50.1 Port Definitions

The interface for the Layer 1 Timer is shown in Figure 50-1. Port definitions for the Layer 1 Timer are provided in Table 50-1.

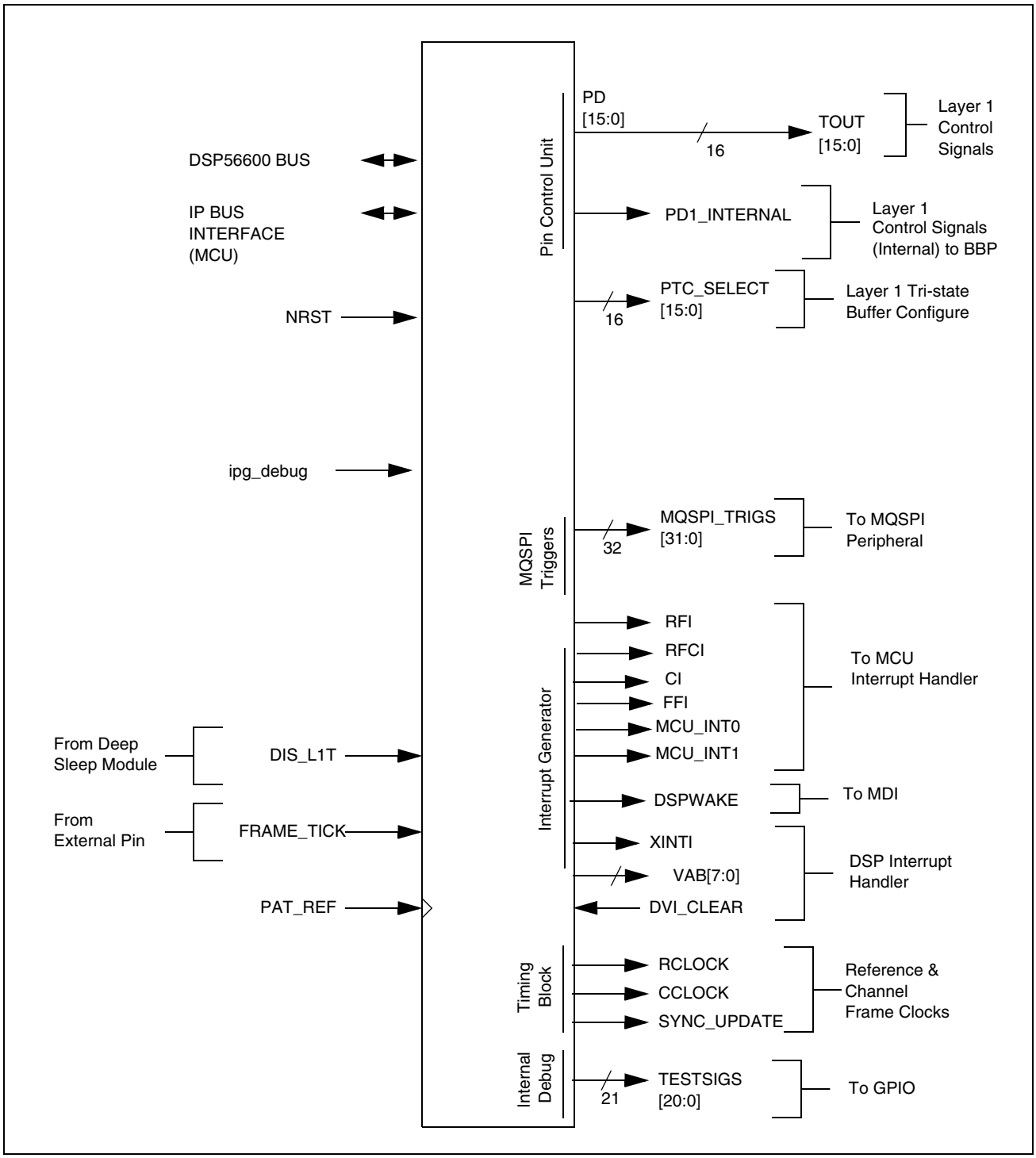


Figure 50-1. Interface Definition

50.1.1 L1T Module Pin List

Table 50-1 is a list of the L1T pins.

Table 50-1. L1T Module Pin List

Pin Name	Direction	Description
md_z[31:0]	Inout	Module to PIG data bus (MIG)
bcstop	Input	DSP entering low power mode
ckih[1]	Input	PAT_REF clock
ckih_clk_b	Input	Inverse PAT_REF clock
l1t_dis	Input	Timer disable from DSM
l1t_frame_tick	Input	external frame tick
gclkw	Input	Raw S-PMBIF clock
gclkw_b	Input	Inverse raw S-PMBIF clock
gdb_wr[15:0]	Input	core gdb write data path
ips_module_en	Input	Peripheral bus Layer 1 Timer module select
mcab_rd[6:0]	Input	core peripheral address bus
mcab_wr[6:0]	Input	core peripheral address bus
mcdis0	Input	T0 clock disable
mcdis1	Input	T1 clock disable
mcsel_rd	Input	X or Y I/O read select
mcsel_wr	Input	X or Y I/O write select
mcu_clk	Input	MCU peripheral clock
mcu_clk_b	Input	Inverse MCU peripheral clock
l1t_rst_b	Input	active-low hardware reset
ipg_debug	Input	Software Debug mode
ips_addr[11:1]	Input	IP address
ips_byte_15_8	Input	Module byte write enables
ips_byte_7_0	Input	Module byte write enables
ips_rwb	Input	Module read/write
ips_bus_clk	Input	IP bus clock
pires	Input	DSP software reset
pivack	Input	Peripheral module interrupt acknowledge
pivrd	Input	Vector read strobe
pivsse	Input	Non-maskable interrupt vector source select
res	Input	DSP hardware reset
scan_en	Input	scan enable
scan_mode	Input	scan_mode
l1t_cclock	Output	50% duty cycle channel frame clock
l1t_ci	Output	Channel Interrupt

Table 50-1. L1T Module Pin List (Continued)

Pin Name	Direction	Description
l1t_dspwake	Output	DSP wakeup interrupt
l1t_ffi[2]	Output	(RFI & RFCI), SATCAP addition DB, 4/26/99
gdb_rd[15:0]	Output	core gdb read data path
l1t_on	Output	Timer enabled
l1t_mcu_int0	Output	MCU Interrupt 0
l1t_mcu_int1	Output	MCU Interrupt 1
ips_rdata[15:0]	Output	Module to IP data bus
mirq_b	Output	Peripheral module non maskable interrupt request
l1t_mqspi_trig[31:0]	Output	L1 Timer Trigger outputs
ips_xfr_err	Output	IP transfer error
ips_xfr_wait	Output	IP transfer wait
l1t_pd1_internal	Output	Internally-routed, always-active-high version of PD1
l1t_pd[15:0]	Output	Pin drivers
l1t_ptc_select[15:0]	Output	Tristate driver enables for pin drivers
l1t_rclock	Output	50% duty cycle reference frame clock
l1t_rfci	Output	Reference Frame Counter Interrupt
l1t_rfi	Output	Reference Frame Interrupt
l1t_sync_update	Output	Synchronous update enable
testsigs[18:0]	O	Internal signals used for debug. b0: fload[3] b1: aload[4] b2: bload[5] b3: cload[6] b4: pload b5: ege[7] b6: qbc_clock[8] b7: out_of_sync[9] b8: fatal[10] b9: mqbc[6][11] b10: mqbc[7] b11: mqbc[8] b12: mqbc[9] b13: mqbc[10] b14: mqbc[11] b15: mqbc[12] b16: mqbc[13] b17: mqbc[14] b18: mqbc[15]
vab_rd[7:1]	Output	interrupt vector bus

1. ckih is a remnant of Patriot. Whenever ckih is mentioned in this chapter, it refers to the 13MHz PAT_REF clock coming into this module
2. the ffi output is not connected in the neptune ic.
3. fload - indicates frame table event is being pre-fetched from RAM.

4. aload - indicates macro A table event is being pre-fetched from RAM.
5. bload - indicates macro B table event is being pre-fetched from RAM.
6. cload - indicates macro C table event is being pre-fetched from RAM.
7. ege - Event Generator Enable, active when event generator is enabled.
8. qbc_clock - Timing Block output, main time base.
9. out_of_sync - internal timer error, should never occur.
10. fatal - indicates Frame Table boundary or prefetch error occurred.
11. mqbc[15:6] - Timing Block output: mqbc_count[15:6].

50.2 MCU Memory Map

The L1 Timer memory map is comprised of registers and RAM. L1 Timer registers are divided into:

- Global control and status registers
- Timing Block counter data storage registers
- Interrupt mask registers
- Event Table Boundary, Frame, and Macro Pointers
- Pin Control Unit configuration and status registers.
- Error mask and error status registers

Event Table RAM is used to store Event Quarter Bit Counts, Event Codes, Quarter Bit Count Delays, and Frame Delays. The memory map is shown in Table 50-2. Memory is organized in 16-bit words and accessed only on word boundaries ($00, $02,...).

NOTE:

The Frame, Macro and Parameter Tables definitions in Table 50-2 are for the default tables (32 entries for each Frame, Macro and Parameter Table). Each of the tables may be defined anywhere in the 192-entry event code RAM ($000 - $17E) and 192-entry event QBC RAM ($200 - $37E) based on the corresponding table START and END pointers. The order of the table definitions does not matter and the hardware allows the tables to be defined as overlapping, though such use is generally not recommended. There is no error checking for overlapping tables, though there is error checking to indicate that a Table START or END pointer value is >= 192. A valid example configuration would be:

— Frame Table 0: Entries 0 - 15 ($000 - $01E and $200 - $21E)

— Macro Table A: Entries 16 - 31 ($020 - $03E and $220 - $23E)

— Frame Table 1: Entries 32 - 95 ($040 - $0BE and $240 - $2BE)

— Macro Table C: Entries 96 - 127 ($0C0 - $0FE and $2C0 - $2FE)

— Parameter Table: Entries 128-143 ($100 - $01E and $300 - $31E)

— Macro Table B: Entries 144-175 ($120 - $15E and $320 - $35E)

— Unused: Entries 176-191 ($160 - $17E and $360 - $37E).

A list of the Layer 1 Timer's registers is contained in Table 50-3. See Section 50.14, "MCU Memory Map Definitions", for more detailed information on register content.

Table 50-2. L1 Timer MCU Memory Map

Address Range[1]	Number of Words	Description
$000 – $07E	64	Default Frame Table Event Codes
$080 – $0BE	32	Default Macro Table A Event Codes
$0C0 – $0FE	32	Default Macro Table B Event Codes
$100 – $13E	32	Default Macro Table C Event Codes
$140 – $17E	32	Default Parameter Table - Frame Delays
$180 - $19E	64	Reserved Memory
$200 – $27E	64	Default Frame Table Event Quarter Bit Counts
$280 – $2BE	32	Default Macro Table A Event Quarter Bit Counts
$2C0 – $2FE	32	Default Macro Table B Event Quarter Bit Counts
$300 – $33E	32	Default Macro Table C Event Quarter Bit Counts
$340 – $37E	32	Default Parameter Table - Quarter Bit Count Delays and Loop Counts
$380 – $400	65	Registers
$402 – $7FE	511	Reserved memory

1. Addresses listed are byte offsets from the start of L1 Timer memory.

Table 50-3. L1 Timer MCU Register List

Address	Access	Register	Description
$380	R/W	MSTR	Master Enable Register for L1 Timer
$382	R/W	GTC0	Global Timer Control - Register 0
$384	R/W	GTC1	Global Timer Control - Register 1
$386	R/W	GTS0	Global Timer Status - Register 0
$388	R/WC	GTS1	Global Timer Status - Register 1
$38A	R/W	IGM	Interrupt Generator Mask
$38C	R/W	GPWT	Global Programmable Wakeup Timer
$38E	R/WC	ES0	Error Source 0
$390	R/W	EM0	Error Mask 0
$392	R/WC	ES1	Error Source 1
$394	R/W	EM1	Error Mask 1
$396	R/WC	ES2	Error Source 2

Table 50-3. L1 Timer MCU Register List (Continued)

Address	Access	Register	Description
$398	R/W	EM2	Error Mask 2
$39A	R/WC	MCUIS	MCU Interrupt Status Register
$39C	R/W	MCUIM	MCU Interrupt Mask Register
$39E	R/W	DVIM	DSP Vectored Interrupt Mask Register
$3A0	R/W	TB_PRE	Timing Block — Prescaler Count
$3A2	R/W	TBPREM	Timing Block — Quarter Bit Count Prescaler Modulus
$3A4	R/W	TRQBC	Timing Block — Reference Quarter Bit Count
$3A6	R/W	TRQBCM	Timing Block — Reference Quarter Bit Count Modulus
$3A8	R	TMQBC	Timing Block - Matching Quarter Bit Count
$3AA	R/W	TCFC	Timing Block — Channel Frame Counter
$3AC	R/W	TRFC	Timing Block — Reference Frame Counter
$3AE	R/W	TRAFN_H	Timing Block — Reference Absolute Frame Number MS Word
$3B0	R/W	TRAFN_L	Timing Block — Reference Absolute Frame Number LS Word
$3B2	R/W	TRAFNM_H	Timing Block — Reference Absolute Frame Number Modulus MS Word
$3B4	R/W	TRAFNM_L	Timing Block — Reference Absolute Frame Number Modulus LS Word
$3B6	R/W	TCFN_H	Timing Block — Channel Frame Number MS Word
$3B8	R/W	TCFN_L	Timing Block — Channel Frame Number LS Word
$3BA	R/W	TQBCFT	Timing Block — Quarter Bit Count Frame Tick
$3BC	R/W	TPIT	Timing Block — Periodic Interrupt Timer Count
$3BE	R/W	TPITM	Timing Block — Periodic Interrupt Timer Modulus
$3C0	R/W	QBCO	Quarter Bit Count Offset
$3C2	R/W	QBCO_UDT_F	QBCO Update Time - Frame
$3C4	R/W	QBCO_UDT_Q	QBCO Update Time - Quarter Bit Count
$3C6	R/W	QBCO_UC0	QBCO Update Control - Register 0
$3C8	R/W	QBCO_UC1	QBCO Update Control - Register 1
$3CA	R/W	EG_FT0_CFG	Event Generator — Frame Table 0 Configuration Register
$3CC	R/W	EG_FT1_CFG	Event Generator — Frame Table 1 Configuration Register
$3CE	R/W	EG_NFT_ CFG	Event Generator Next Frame Table Configuration Register
$3D0	R/W	EG_MA_CFG	Event Generator Macro A Configuration Register
$3D2	R/W	EG_MB_CFG	Event Generator Macro B Configuration Register
$3D4	R/W	EG_MC_CFG	Event Generator Macro C Configuration Register
$3D6	R/W	EG_PT_CFG	Event Generator Parameter Table Configuration Register

Table 50-3. L1 Timer MCU Register List (Continued)

Address	Access	Register	Description
$3D8	R	EG_FPTR	Event Generator — Frame Table Pointer
$3DA	R	EG_MPTR0	Event Generator Macro Table Pointer register 0
$3DC	R	EG_MPTR1	Event Generator Macro Table Pointer register 1
$3DE	R/W	EG_D0_FD	Event Generator — Variable Delay Register 0 - Frame Delay
$3E0	R/W	EG_D0_QD	Event Generator — Variable Delay Register 0 - QBC Delay
$3E2	R/W	EG_D1_FD	Event Generator — Variable Delay Register 1 - Frame Delay
$3E4	R/W	EG_D1_QD	Event Generator — Variable Delay Register 1 - QBC Delay
$3E6	R/W	EG_D2_FD	Event Generator — Variable Delay Register 2 - Frame Delay
$3E8	R/W	EG_D2_QD	Event Generator — Variable Delay Register 2 - QBC Delay
$3EA	R/W	EG_D3_FD	Event Generator — Variable Delay Register 3 - Frame Delay
$3EC	R/W	EG_D3_QD	Event Generator — Variable Delay Register 3 - QBC Delay
$3EE	R/W	EG_LP0	Event Generator — Variable Loop Count Register 0
$3F0	R/W	EG_LP1	Event Generator — Variable Loop Count Register 1
$3F2	R/W	EG_LP2	Event Generator — Variable Loop Count Register 2
$3F4	R/W	EG_LP3	Event Generator — Variable Loop Count Register 3
$3F6	R/W	PCFG	Pin Control Unit — Configuration Register
$3F8	R/W	PD	Pin Control Unit — Default Register
$3FA	R	PS	Pin Control Unit — Status Register
$3FC	R/W	PTC	Pin Control Unit — Tri-state Configuration Register
$3FE	R/W	PDIMASK	Pin Driver Internal - Mask Register
$400	R/W	EG_MD	Event Generator — Macro Disable Register

50.3 Timing Block

The Timing Block converts the system clock (19.44/16.8/13.0 MHz) from the Clock Manager into a quarter-bit timing reference for the Event Generator. An offset (**qbco**) is used for aligning handset and base station timing. End-of-frame and programmable wakeup interrupts are generated for both the exec and Layer 1.

The Timing Block is shown in Figure 50-2. Port definitions for the Timing Block are contained in Table 50-4. Functional descriptions of each Timing Block module are contained in Sections 50.3.1 - 50.3.8.

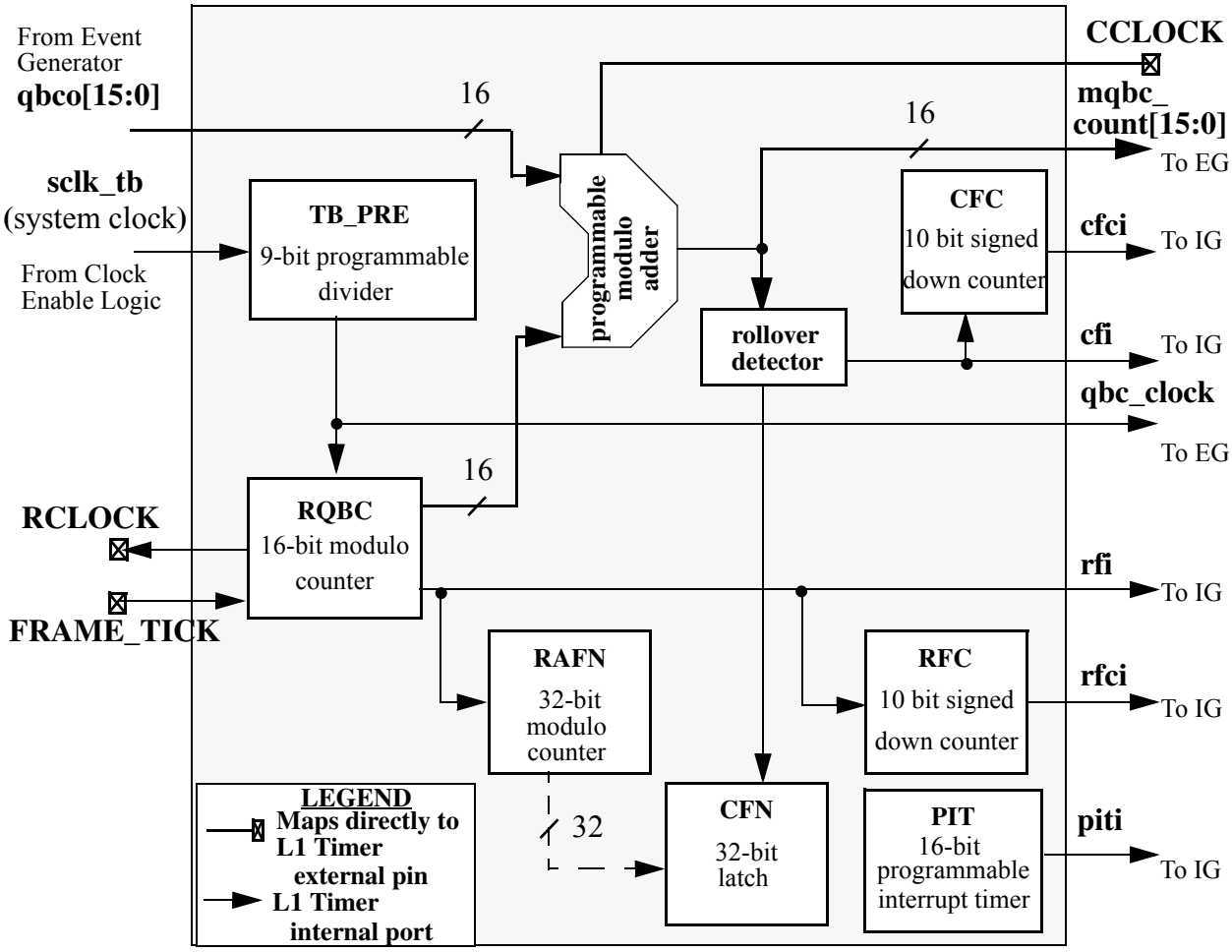


Figure 50-2. Timing Block

Timing Block. Outputs of this module go to the Layer 1 Timer's Event Generator (EG) or Interrupt Generator (IG) blocks. QBCP = Quarter Bit Count Prescaler, RQBC = Reference Quarter Bit Counter, RAFN = Reference Absolute Frame Number, CFN = Channel Frame Number, RFC = Reference Frame Counter, CFC = Channel Frame Counter, sclk_tb = Timing Block's gated system clock, qbco = Quarter Bit Count Offset, mqbc_count = Matching Quarter Bit Count, cfci = Channel Frame Counter Interrupt, cfi = Channel Frame Counter Interrupt, qbc_clock = Quarter-Bit Clock, rfi = Reference Frame Interrupt, rfci = Reference Frame Counter Interrupt.

Table 50-4. Port Definitions: Timing Block

Signal Name	I/O	To/From	Description
sclk_tb	I	Clock Enable logic	Gated 19.44/16.8/13.0 MHz system clock
FRAME_TICK	I	L1 Timer external pin	Frame tick reset used to externally reset the rqbc counter for test purposes.
RCLOCK	O	L1 Timer external pin	Reference frame clock, 50% duty cycle, synchronized to the reference frame.
qbco[15:0]	I	Event Generator	Quarter Bit Count Offset
CCLOCK	O	L1 Timer external pin	Channel frame clock, 50% duty cycle, synchronized to the channel frame.
cfci	O	Interrupt Generator	Channel Frame Counter Interrupt
cfi	O	Interrupt Generator	Channel Frame Interrupt
mqbc_count[15:0]	O	Event Generator	Matching Quarter Bit Count
rfi	O	Interrupt Generator	Reference Frame Interrupt
rfci	O	Interrupt Generator	Reference Frame Counter Interrupt
piti	O	Interrupt Generator	Periodic Interrupt Timer Interrupt
qbc_clock	O	Event Generator	Quarter Bit Clock

50.3.1 Prescaler (TB_PRE)

The prescaler is a 9-bit programmable divider which divides the L1 Timer system clock (19.44/16.8/13.0 MHz) to create the quarter-bit clock (**qbc_clock**). The prescaler count can be accessed through the TB_PRE R/W register. The prescaler modulus is set by writing to the TBPREM register through the MCU. Valid values of TB_PREM are from 11 through 511 and represent divide by 12 through 512 respectively. Values of 0 through 10 for TBPREM are not valid and shall inhibit operation of the L1 Timer. Values of 11 through 14 shall inhibit Macro Table C from operating. A Prescaler Error modulus error bit (PME) is defined as bit 6 of the Event Source 2 (ES2) register. This bit shall be set when a value of 10 or less is entered into the TBPREM register. This error condition is not maskable and puts the L1 Timer event generator (EG) into sleep mode.

50.3.2 Reference Quarter Bit Counter (RQBC)

The Reference Quarter Bit Counter block contains a 16-bit, programmable modulus counter and provides reference frame timing. The count output bus is **rqbc_count[15:0]**. The programmable modulus value (TRQBCM) register can be programmed over the range of 0 through 65535 which represents a modulo 1 through 65536. The Reference Frame Interrupt (**rfi**) asserts when the counter rolls over at the programmed modulo value. The TRQBCM register is read and write accessible by the MCU.

When the **rqbc_count** rolls over:

1. A new reference frame is started, and the Reference Frame Interrupt is asserted.
2. If the Reference Frame Counter (RFC) is enabled[1], then it is decremented. If RFC = 0, the RFCI interrupt is asserted.
3. The Reference Absolute Frame Number (RAFN) counter is incremented.

1. RFCE = 1 in the GTC1 control register.

The RCLOCK signal represents a ~50% duty cycle. This is achieved by comparing the **rqbc_count[15:0]** bus to the modulo value **trqbcm[15:0]** divided by 2.

The external FRAME_TICK input signal allows the RQBC counter to be reset. The reset capability functions according to 3 control signals: 1) enable/disable reset function; 2) select rising/falling edge reset; and 3) single shot/continuous reset mode.

The Timer Quarter Bit Counter Frame Tick (TQBCFT) register (MCU R/W accessible) holds the 3 control bits mentioned. The Frame Tick Enable (FTEN) bit enables the reset function when FTEN is a 1, and disables it when FTEN is a 0. The Frame Tick Edge (FTED) bit selects falling edge when FTED is a 1, and rising edge when FTED is a 0. The Frame Tick Mode (FTM) bit selects single shot when FTM is a 1, and continuous when FTM is a 0. When configured in single shot mode (FTM = 1) and when the frame tick is enabled (FTEN = 1), the FTEN bit is automatically cleared when the RQBC counter is reset by the FRAME_TICK signal.

50.3.3 Modulo Adder and Rollover Detector Block

Event scheduling in the mobile equipment needs to be aligned with event scheduling at the base station. To provide the required timing alignment, an offset (**qbco**) is added to the reference frame time (**rqbc_count**) to produce the Matching Quarter Bit Count (**mqbc_count**). The CCLOCK signal represents a ~50% duty cycle. This is achieved by comparing the **mqbc_count[15:0]** bus to the modulo value **trqbcm[15:0]** divided by 2 (right-shifted by 1 bit). Due to the implementation of the logic, RCLOCK lags CCLOCK (see below) by qbco+1 counts even though **rqbc_count** lags **mqbc_count** by exactly **qbco** counts. CCLOCK and RCLOCK are used only for analysis and test of the L1 Timer. The current value of the mqbc_count is read-only accessible by the MCU via the TMQBC register.

When the **mqbc_count** rolls over:

1. A new channel frame is started, and the Channel Frame Interrupt (**cfi**) is asserted.
2. If the Channel Frame Counter (CFC) is enabled[1], then it is decremented. If CFC = 0, the CFCI interrupt is asserted.
3. The Reference Absolute Frame Number (TRAFN_H and TRAFN_L) is latched into the Channel Frame Number (TCFN_H and TCFN_L) registers.

1. CFCE = 1 in the GTC1 control register.

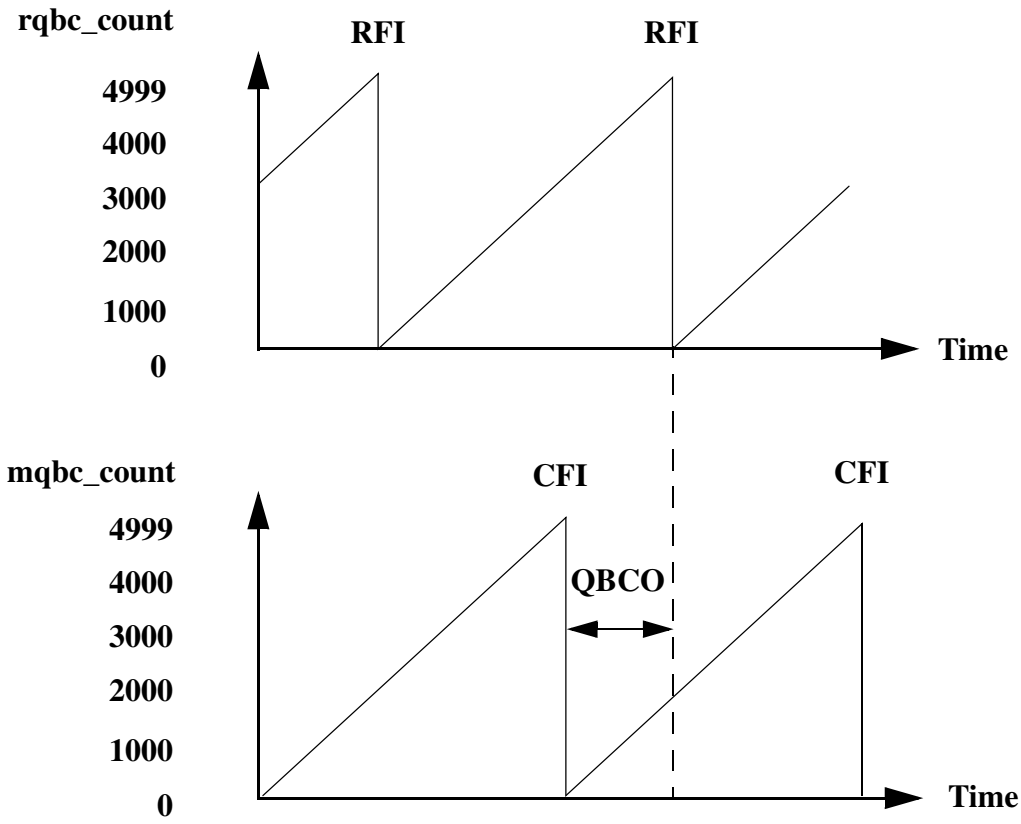


Figure 50-3. Relation Between Reference Frame Timing (top) and Channel Frame Timing (bottom) for trqbcm[15:0] = 4999.

The timing relationship between the reference frame, channel frame, and Quarter-Bit Count Offset is shown in Figure 50-3.

50.3.4 Channel Frame Counter (CFC)

The Channel Frame Counter is used to generate a Layer 1 wake-up interrupt after the processor has been asleep. The CFC is enabled by setting the CFCE bit in the Global Timer Control 1 (GTC1) register.

The CFC is a 10-bit signed down-counter. The CFC should be loaded by writing a (positive) 9-bit initial value. If the CFC is enabled, a trigger signal from the rollover detector decrements the CFC. When the counter reaches zero, the CFC will generate a Channel Frame Counter Interrupt (CFCI). After reaching zero, triggers from the rollover detector will continue to decrement the CFC (-1, -2, -3) until it is reloaded. This allows the interrupt service routine to count the number of frames the interrupt has been pending. The 10-bit contents of the CFC counter are sign-extended to 16 bits when the TCFC register is read.

50.3.5 Reference Frame Counter (RFC)

The Reference Frame Counter is used to generate an exec wake-up interrupt after the processor has been in a long sleep period. The RFC is enabled by setting the RFCE bit in the Global Timer Control 1 (GTC1) register.

The RFC is a 10-bit signed down-counter. The RFC should be loaded with a (positive) 9-bit initial value. If the RFC is enabled, a trigger signal from the Reference Quarter Bit Counter decrements the RFC. When the counter reaches zero, the RFC will generate a Reference Frame Counter Interrupt (RFCI). After

reaching zero, triggers from the RQBC will continue to decrement the RFC (-1, -2, -3) until it is reloaded. This allows the interrupt service routine to count the number of frames the interrupt has been pending. The 10-bit contents of the RFC counter are sign-extended to 16 bits when the TRFC register is read.

50.3.6 Reference Absolute Frame Number (RAFN)

For GSM a hyperframe is defined as 2715648 frames. The Reference Absolute Frame Number is a free-running 32-bit modulo counter. The RAFN is enabled by asserting the Frame Number Enable (FNE) bit in the Global Timer Control 1 (GTC1) register. If FNE is asserted, the RAFN is incremented at the start of the reference frame (**rqbc_count** = 0). The counter modulus is contained in two MCU R/W registers, TRAFNM_H (upper 16 bits) and TRAFNM_L (lower 16 bits). The current value of the rafn_count is R/W accessible by the MCU via the TRAFN_H (upper 16 bits) and TRAFN_L (lower 16 bits) registers. See Section 50.16, for more information on the use of reference and channel frame numbers.

50.3.7 Channel Frame Number (CFN)

The Channel Frame Number is a latched copy of the Reference Absolute Frame Number. The CFN is a 32-bit value and is enabled by setting the FNE bit in the GTC1 register. If FNE is asserted, the Channel Frame Number is latched one system clock cycle before the start of the channel frame (mqbc_count = 0). The current value of the CFN is R/W accessible by the MCU via the TCFN_H (upper 16 bits) and TCFN_L (lower 16 bits) registers.The relation between the reference and channel frame numbers at various values of qbco is shown in Figure 50-4, Figure 50-5 and Figure 50-6. Note that when qbco = 0, the reference and channel frames are aligned. In this case, the CFN will lag the RAFN by exactly one frame, as shown in Figure 50-4. See Section 50.16 for more information on the use of reference and channel frame numbers.

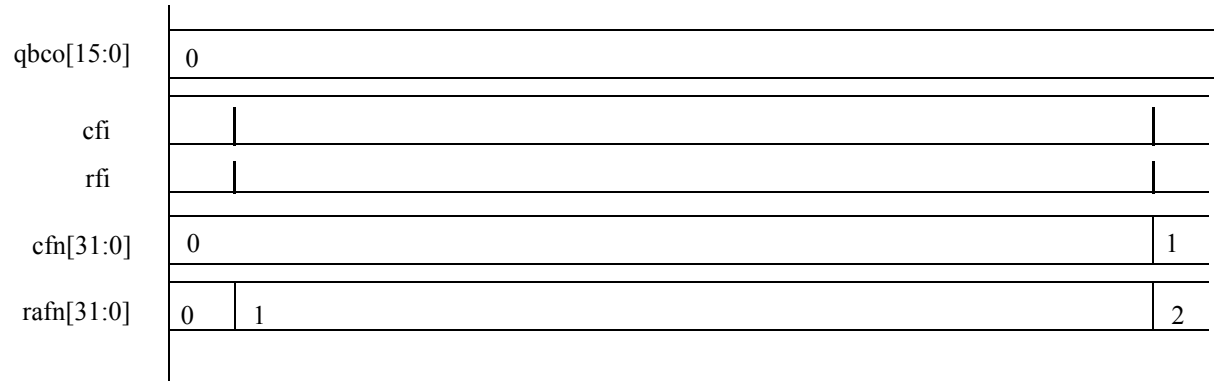


Figure 50-4. Relation between RAFN and CFN when QBCO = 0.

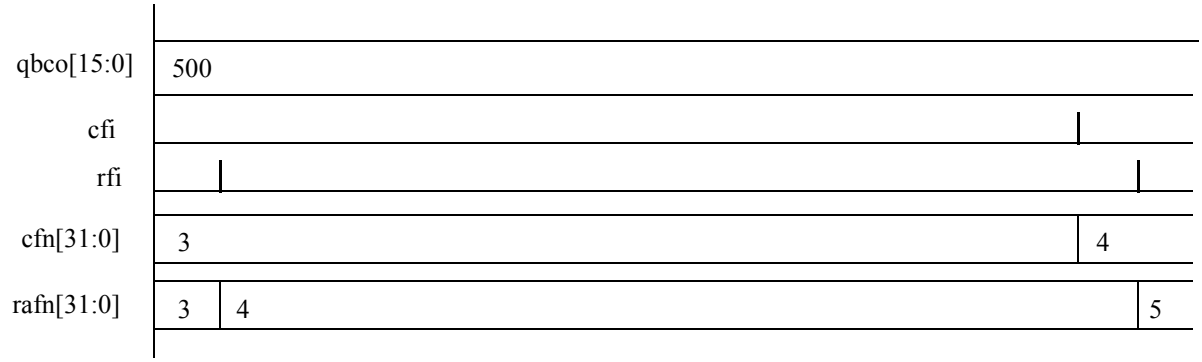


Figure 50-5. Relation between RAFN and CFN when QBCO = 500.

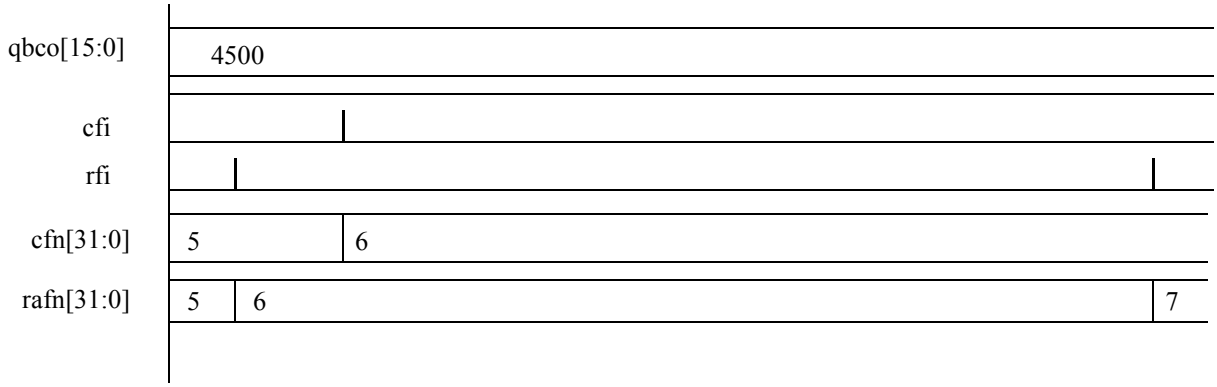


Figure 50-6. Relation between RAFN and CFN when QBCO = 4500.

50.3.8 Periodic Interrupt Timer (PIT)

The operating system used for the call processor software can require a periodic interrupt. The Periodic Interrupt Timer (PIT) generates a free-running periodic interrupt that is routed to the MCU. The PIT contains a 16-bit, programmable modulus counter which uses the Quarter Bit Clock from the Quarter Bit Clock Prescaler block. The modulus value register (TPITM) is MCU read/write accessible. The timebase for the PIT is derived from qbc_clock of the Timing Block (see Figure 50-2). The Timing Block must be enabled for the PIT to run.

A value of 0 (reset value) for TPITM disables the periodic interrupt timer. When the PIT counter rolls over, the periodic interrupt timer interrupt (**piti**) signal is asserted. If the Event Generator is enabled the **piti** signal is combined with other interrupt signals which becomes the MCU_INT0 output signal of the L1 Timer module, which is defined as an active-low level interrupt. The **piti** is masked and cleared using MCU R/W accessible registers described below.

The PIT will free run in SLEEP mode but cannot generate interrupts. To run and generate interrupts the L1 Timer must be in TIMER_ON mode. See Table 50-11 for details on operating modes.

The **piti** signal is masked by the bit, PITIE, in the Interrupt Generator Mask (IGM) register. When the PITIE bit is 1, the interrupt is enabled. When the PITIE bit is 0 (reset value), the interrupt is masked.

The status of the **piti** signal is monitored by PITI, in the Global Timer Status (GTS1) register. When the PITI bit is 0, the interrupt is not asserted. When the PITI bit is 1, the interrupt is asserted. The PITI status is cleared to 0 (and the corresponding **piti** signal is de-asserted) when 1 is written to the PITI bit.

The Timer Periodic Interrupt Timer (TPIT) register is read and write accessible by the MCU. This register provides the instantaneous count of the PIT counter. The PIT counter can be altered by writing to this register.

50.3.9 System Clock Divider

An option is provided to divide the L1 Timer system clock (PAT_REF - 19.44/16.8/13.0 MHz) by 2. This option is provided to guarantee that the L1 Timer will operate correctly when running at the maximum system clock frequency (19.44 MHz) and the lowest voltage (1.8v). When operating at a maximum system clock frequency of 19.44 MHz and a voltage of 2.2v, the system clock divider is not necessary. When the IC is functioning in a GSM mode (13.0 MHz) or an IDEN mode (16.8 MHz) the system clock divider is not required at any operating voltage.

To enable the divider the DIVEN bit of the Global Timer Control 0 (GTC0) register must be set. This should be done after setting the MST_EN bit of the Master Enable (MSTR) register. Note that a hard or a soft reset of the L1 Timer will revert the system clock divider to it's default state (disabled).

50.3.10 Debug Mode

To aid in system debug, the L1 Timer can halt all operations when DEBUG mode is entered. This feature allows the debug system to read the state of the L1 Timer at the instant DEBUG mode is activated. Without this feature the timer will likely advance to subsequent states before the debug system can respond.

By default DEBUG mode is ignored by the L1 Timer. To enable halting of the L1 Timer in DEBUG mode, the DBEN bit in the Global Timer Control 0 (GTC0) register must be set.

50.4 Event Generator

The L1 timer generates all Layer 1 control signals, starts MQSPI transfers, and sends interrupts to the call processor and DSP. Scheduling in the Layer 1 timer is performed by the Event Generator, shown in Figure 50-7. Port definitions for the Event Generator are contained in Table 50-5. Functional descriptions of each Event Generator module are contained in Sections 50.4.1 - 50.4.9.

50.4.1 Event Table

The 192-entry Event Table consists of dynamically configurable parameter table, macro tables (three - A, B and C), and frame tables (two). The tables contain lists of events (Event Codes, or ECs) and their corresponding execution times. Event scheduling is performed by sequentially stepping through table entries.

Note that a given table can execute only one event in a given quarter-bit count. Thus, it is possible to execute a frame table event, a Macro A table event, a Macro B table event and a Macro C table event in the same quarter-bit count. The Event Generator is designed to execute events from multiple tables in the same QBC without race conditions. Contention logic is provided to prevent glitching on Layer 1 Control pins when multiple tables request conflicting values (See Section 50.6, "Error Detector", for more information).

50.4.2 Frame Tables

The frame tables can be dynamically partitioned into two frame tables anywhere within the 192-entry Event Table RAM using the Frame Table Configuration Registers EG_FT0_CFG and EG_FT1_CFG. These registers consist of START pointers (**ft0_start**, **ft1_start**) to define the first table entry, and an END

pointer (**ft0_end**, **ft1_end**) to define the last table entry. The EG_NFT_CFG register consists of NEXT_FRAME_TABLE bits (**ft1_next**, **ft0_next**) which defines the next active frame table upon completion of the current frame table.

Each entry in the frame table RAM block contains two fields: a 8-bit Event Code (EC) and a 16-bit Event Quarter Bit Count (EQBC). The EQBC field specifies the event's execution time (modulo <**trqbcm** value>) relative to the start of the channel frame.

At a given time, only one of the frame tables can be active. The Layer 1 Timer architecture is designed such that the programmer should not have to overwrite entries in the active table. However, there is no hardware to prevent active table overwrites.

When the last event in the active table is executed, the next active table is selected by the NEXT_FRAME_TABLE bit. Depending on the value of this bit, the user can either alternate between frame tables or repeat the same table. For example, assume frame table 0 is executing and ft0_next = 1. When frame table 0 is finished, at the beginning of the next channel frame, frame table 1 will become the active frame table. However, if ft0_next = 0, frame table 0 would be repeated. The ft_next bit of the active frame table can be changed anytime during a channel frame up until **mqbc_count** = <**trqbcm**-1>. Changing the ft_next bit of the active frame table while **mqbc_count** = <**trqbcm**-1> or <**trqbcm**> will interfere with the Address Generation Unit's fetch sequence.

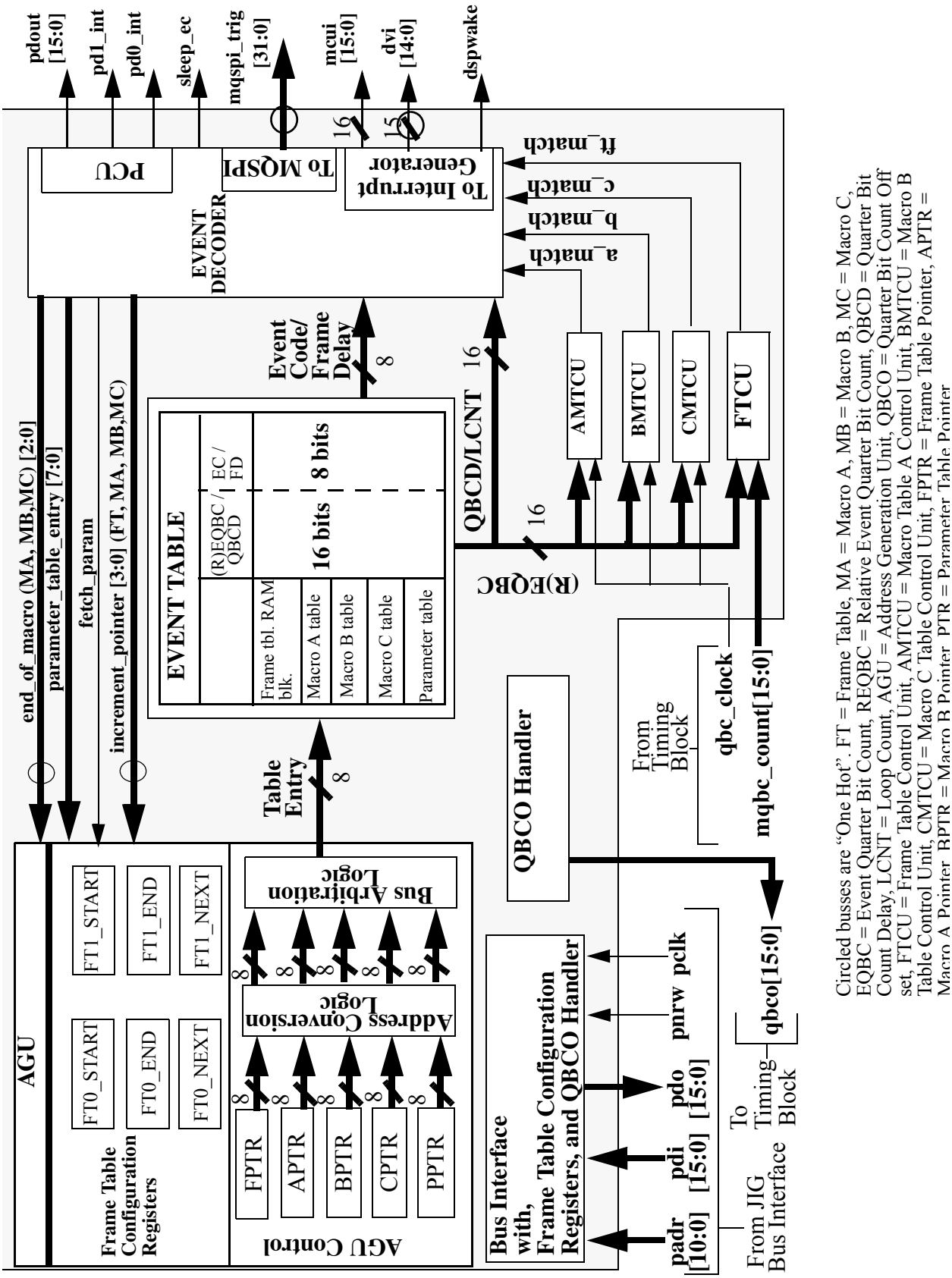


Figure 50-7. Event Generator

Table 50-5. Port Definition: Event Generator

Signal Name	I/O	To/From	Description
mqbc_count[15:0]	I	Timing Block	Matching Quarter Bit Count
qbco[15:0]	O	Timing Block	Timing Offset
qbc_clock	I	Timing Block	Quarter-Bit Clock
ips_addr[11:1]	I	IP Bus Interface	Address bus
ips_wdata[15:0]	I	IP Bus Interface	input data to L1 timer
ips_rdata[15:0]	O	IP Bus Interface	output data to MCU
ips_bus_clk	I	IP Bus Interface	write clock
ips_rwb	I	IP Bus Interface	peripheral read/write
pdout[15:0]	O	Pin Control Unit	Pin driver outputs
pd0_int	O	Pin Control Unit	This signal asserts (active high) when PD[0] is active (either high or low).
pd1_int	O	Pin Control Unit	This signal asserts (active high) when PD[1] is active (either high or low).
sleep_ec	O	Wake Control	Sleep event-code decoded. Puts EG in sleep mode.
mqspi_trig[31:0]	O	MQSPI peripheral	MQSPI control triggers
mcui[15:0]	O	Interrupt Generator	MCU Interrupts
dvi[14:0]	O	Interrupt Generator	DSP Vectored Interrupts
dspwake	O	Interrupt Generator	DSP Wake Interrupt

NOTE:

Event Table RAM is accessed over a 16-bit MIG interface

The events in a frame table can be executed over multiple frames. For example, executing the sequence

EQBCEvent Code

- FT0_START ->100PD0-1
- FT0_END ->200PD0-0

will cause the PD0 pin to be driven high at time 100, then driven low at time 200. This entire table will execute in a single frame.

In contrast, consider the sequence

AEQBCEvent Code

- FT0_START ->100PD0-1
- FT0_END ->100PD0-0

At time 100, PD0 is driven high. Since the next event is also scheduled at time 100 (modulo 5000), the timer must wait until time 100 *in the next channel frame* to execute the PD0-0 event code. Thus, this table will take two frames to execute.

50.4.3 Macro Tables

Macro tables are used when it is desirable to repeatedly execute an event sequence. A typical use for a macro would be to setup an entire Rx or Tx burst. A macro table event consists of two fields: an 8-bit Event Code (EC) and a 16 bit Relative Event QBC (REQBC). The REQBC field specifies a relative delay from the previous event.

A macro table could hold up to 191 events if configured as such, but in practice is often 32 events or less. If a macro START or END pointer is >= 192, a macro boundary error will be asserted. Figure 50-8 shows the relationship between Absolute[1] and Relative Event Quarter-Bit Counts. The first event in a macro will execute at absolute time[2]:

$$AEQBC(\text{Macro Event 0}) = AEQBC\ (\text{Macro start}) + REQBC(\text{Macro Event 0}) + 1$$

Similarly, the second event will execute at absolute time:

$$AEQBC(\text{Macro Event 1}) = AEQBC(\text{Macro Event 0}) + REQBC(\text{Macro Event 1}) + 1$$
$$= AEQBC(\text{Macro start}) + REQBC(\text{Macro Event 0}) +$$
$$REQBC(\text{Macro Event 1}) + 2$$

REQBC's use zero-offset numbering. An event with an REQBC of zero will execute one quarter bit after the previous event. Note also that if the first event in a macro has an REQBC of zero, it will execute one quarter-bit count after the frame table starts the macro

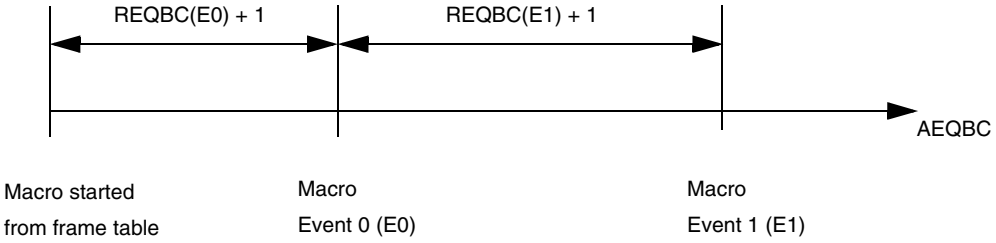


Figure 50-8. Relationship Between Absolute Event Quarter-Bit Counts (AEQBCs) and Relative Event Quarter-Bit Counts (REQBCs)

Macros are invoked with the parameter table entry which is used when executing the macro event DELAY_TBL or LOOP_TBL. The associated delay value for the DELAY_TBL event code is latched when the macro is started from the frame table. For example, executing the Event Code MACRO_A_4 starts Macro A and latches in parameter table entry 4 and increments the parameter table pointer to entry 5. If a DELAY_TBL or LOOP_TBL event is called while Macro A is active, the delay/loop value will be latched from parameter table entry 5.

1. NOTE: AEQBC is the total elapsed time since the Timing Block was turned on. Do not confuse this with EQBC, which is the modulo <**trqbcm** value> channel frame time.
2. It is necessary to specify the following equation in AEQBC time since a macro may execute over multiple channel frames.

Macros can execute delays using either DELAY_TBL or DELAY_REGx events (where x = 0,1,2,3). The DELAY_TBL event provides a mechanism for varying the pulse width of a signal in predetermined increments. The DELAY_REGx events uses the delay value stored in one of 4 delay registers. These register can be reprogrammed at any time[1]. The DELAY_REGx instructions can be used to execute a variable delay which cannot be predetermined (such as a timing advance value).

A LOOP_TBL event code marks the next event entry as the starting location of a repeat loop and causes the 16-bit repeat count to be loaded from the Parameter Table according to the parameter pointer (the same parameter pointer used for the DELAY_TBL events). Note, only the 16-bit QBC section of the Parameter Table is used for loop counts, unlike delays called via the DELAY_TBL event code. The parameter pointer is then incremented to the next address in the Parameter Table. The loop can be repeated up to 65535 times.

Secondly, a loop can be initiated using a register instead of a parameter. There are 4 16-bit registers: EG_LP0, EG_LP1, EG_LP2, and EG_LP3. These registers are accessed using events codes named LOOP_REGx (x=0,1,2,3) that start the loop with the corresponding registers.

Finally, a loop can be initiated using a constant instead of a parameter or a register. There are 16 event codes named LOOP_0 through LOOP_15 which define a loop count of 0 through 15, respectively. The loop will be able to repeat up to 15 times (using LOOP_15).

When LOOP_0 event is executed or a LOOP_TBL/LOOP_REGx event is executed with a loop count parameter of 0, the macro immediately terminates just as if an end of macro (EOM) event was executed.

The END_LOOP event code defines the last event to be executed in the loop. The loop count is decremented upon execution of this event. After the count is decremented, if the count is not 0, then the loop is repeated such that the macro continues executing events starting with the event immediately following the LOOP_x/LOOP_TBL/LOOP_REGx event. If the count is 0, then the loop is no longer repeated and the next sequential event entry is executed as specified by it's associated REQBC.

All macros shall be able to execute more than one loop structure, so long as an active loop terminates before another begins. However, loops cannot be embedded within other loops.

If a LOOP_x/LOOP_TBL/LOOP_REGx is executed while currently a loop is active, an error is generated. Also, if an END_LOOP event code is executed while a loop is not active, an error shall be generated. Refer to Section 50.6 for more on Error Detection.

Macro C will not be functional unless the Quarter Bit Clock Prescaler (**tbprem[8:0]**) >= 15. This is because the event scheduler needs at least 4 system clocks pre frame/macro table to decode and schedule access to the Event Table RAM. Thus for 3 macros and 1 frame table, at least 16 system clocks are required per QBC to process all 4 tables. The Parameter Table has no such requirement since it is accessed indirectly through macro and frame table calls. When the prescaler is less than 16, the Macro C Disable (MCD) bit will automatically be set in the EG_MD register. All three macros can be disabled via software by writing to the EG_MD register. See Table 50-84 on page 50-130 for bit definitions.

Under normal conditions, the macro tables do not need to be reprogrammed once they have been set up. However, there is no hardware to prevent overwriting a macro table. Macro A, Macro B and Macro C can all execute simultaneously. However, attempting to execute Macro A (Macro B,C) while Macro A (Macro B,C) macro is already running will cause an error. Macros can only be started from a frame table - a macro cannot be started from another macro. See Section 50.13, "Event Codes", for more information on restricted event codes, and Section 50.6, "Error Detector", for more information on error traps.

1. Note that the DELAY_REG event latches the register value at execution time. Changing the register contents after the delay has begun will not affect the delay length.

50.4.4 Parameter Table

There is a single Parameter Table which can be configured to hold up to 192 entries. Macros are called with an associated delay table entry. An entry consists of two fields: a 8bit Frame Delay (FD) and a 16 bit QBC Delay (QBCD) or loop count value. When a delay instruction is executed, the timer will wait for FD*(<**trqbcm** value> + 1) + QBCD + 1 quarter-bit counts before fetching the next table entry. Note that if FD = 0 and QBCD = 0, the minimum delay duration is 1 quarter-bit count since expiration of the delay takes 1 QBC to execute. Thus, with FD = 0 and QBCD = 0, a delay statement executed at time X will expire at time X+1. The earliest the next instruction can execute is at time X+2. Therefore, given a pulse width in microseconds, the user should program the delay:

DELAY = (Pulse Width * (CKIH freq. (MHz))/(<tbprem> + 1)) - 3 (QBC's)

This formula accounts for one QBC to execute the instruction to de-assert the pulse.

The user may occasionally need to immediately end a delay. Each macro has an Immediate Delay Terminate (IDT) bit in the Global Timer Control 0 (GTC0) register. When a macro's IDT bit is asserted, all subsequent delays in that macro will be of minimum duration (1 quarter-bit count). IDT bits are automatically cleared upon macro completion. Note that an IDT bit can only be set when the macro is active.

50.4.5 Frame Table Control Unit

The Frame Table Control Unit schedules the execution of all frame table events. Normally. when the contents of the EQBC latch are equal to the Matching Quarter-Bit Count, the **ft_match** signal is asserted (See Figure 50-9). The Event Decoder executes a frame table event, then signals the Address Generation Unit to increment the Frame Table Pointer. Incrementing this pointer loads the EQBC of the next event into the latch.

However, there is a "Catch-Up" mode in which it is sometimes necessary for the event schedule to "catch-up" with the current time. In this case, events must execute if their AEQBC <= MQBC. The "Catch-Up" mode is enabled when the CUP bit in the GTC1 register is asserted high.

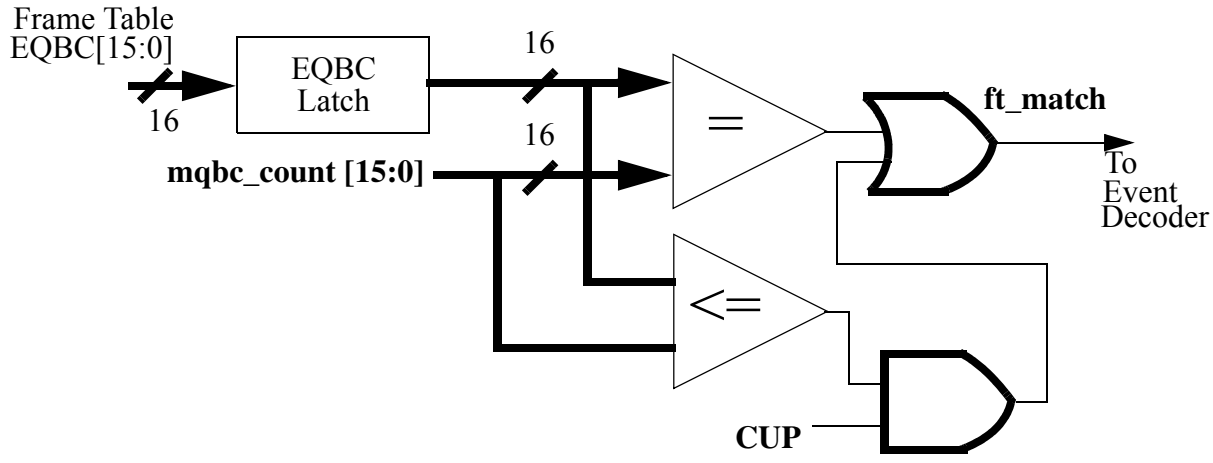


Figure 50-9. Frame Table Control Unit

50.4.6 Macro Table Control Units

The Macro Table Control Units schedule all macro events. The timer has three MTCUs so that Macro A, Macro B and Macro C are independent and can be executed simultaneously. Macro event timing is based on Relative Event Quarter-Bit Counts. This allows macro timing to be done using counters. Starting a macro from the Frame Table causes the REQBC value from Macro Event 0 to be loaded into a counter. Once this counter is enabled, it decrements once per quarter-bit. When the counter reaches zero, the match signal (**a_match**, **b_match, c_match**) is sent to the Event Decoder. The Event Decoder executes the macro table event, then signals the Address Generation Unit to increment the Macro Table Pointer. Incrementing this pointer loads the REQBC of the next event into the MTCU counter. The counter is disabled when the macro terminates, either by executing the End-of-Macro (EOM) event, a LOOP_0 event code or executing the last event in the macro table.

50.4.7 Event Decoder

The Event Decoder contains all the necessary hardware for executing Layer 1 Timer Events. Layer 1 Timer events fall into the following categories:

- Start macro
- Assert an MQSPI trigger
- Program the MQSPI trigger select register MREG
- Assert the MQSPI trigger stored in MREG
- Set/Clear a Layer 1 control pin PD15 - PD0
- Generate an interrupt (MCU or DSP)
- Activate a macro delay (DELAY_TBL or DELAY_REGx)
- Signal the End-Of-Macro to the Address Generation Unit
- Shut-down the event generator (sleep mode)
- Wake up the DSP.
- Reload/change QBCO.
- Do nothing (NOP).

Once an event has been executed, the Event Decoder must signal the AGU that it is time to move on to the next event. This is done through the **increment_pointer** bus (see Figure 50-7). At the start of a macro, the associated delay must be fetched. The Event Decoder uses the **delay_table_entry** bus to select the desired delay (0-15) (See Figure 50-7). The **end_of_macro** bus is used to signal End-of-Macro (“A”, “B” or “C”) to the AGU.

50.4.8 Address Generation Unit (AGU)

The AGU contains the configuration registers and control logic for sequencing through frame, macro, and delay tables and transferring event data from the Event Table to the Event Decoder and Table Control Units.

Frame table boundaries are stored in START and END registers (See Figure 50-7). The FPTR, APTR, BPTR and CPTR registers respectively point to the current event in the frame table, Macro A table, Macro B table and Macro C table.

FPTR is constrained by the active frame table's START and END pointers. When FPTR is equal to the active table's END pointer, the AGU will wait until the end of the current channel frame (**mqbc_count** = trqbc,). FPTR will then be set to either **ft0_start** or **ft1_start**, based on the value of the active table's NEXT_FRAME_TABLE (**ft0_next**, **ft1_next**) bit.

The macro table pointers are handled differently than FPTR. When an EOM event is executed (asserting the **end_of_macro** signal) or the last entry in the macro table has been executed, the macro table pointer is set to entry 0. Macro table pointers are used as an offset to each macro's START pointer when generating the correct address to access the Event Table.

The AGU also checks the START and END frame and macro pointers for out-of-bounds values (>= 192) and flags either a frame or macro boundary error if that condition is detected.

In a given CKIH cycle, only one of the tables (either frame, Macro A, Macro B, Macro C or parameter) can access the Event Table. The Event Decoder handles scheduling to ensure that only one AGU control signal (**fetch_param**, **end_of_macro, increment pointer**)[1] will be asserted in any given CKIH cycle.

Not shown in Figure 50-7 are the "data valid" latch signals the AGU sends to the Event Decoder and the Table Control units. Also not shown is the logic used to allow the MCU and the AGU to share access to Event Table RAM.

50.4.9 QBCO Handler

A timing offset is required to synchronize the handset with the base station. As described in Section 50.3.2, **rqbc_count** and the offset **qbco**[2] are added to form the Matching Quarter Bit Count (**mqbc_count**). This addition converts the reference-frame-relative timing of **rqbc_count** to channel-frame-relative timing. The timing offset must be updated in order to reflect changes in the handset/base station transmission delay. These updates must be done without interfering with event execution or frequently shutting down the timer. To handle offset updates, the QBCO register has a master/slave configuration.

QBCO has two update modes: Synchronous and Asynchronous. Update mode is specified by the Synchronous Update Enable (SUE) bits in the QBCO Control 1 (QBCO_UC1) register. The default mode is Asynchronous Update. The Timing Block always sees the output of the QBCO slave register. Bus Interface access to the QBCO depends on the settings of the SUE bits in the QBCO_UC1 register and the TBE bit in the GTC1 register.

Asynchronous Update: Asynchronous update is required when the mobile equipment has to make a large change in the timing offset, typically during inter-cell handover or after reset. The QBCO master and slave registers are loaded with the same value during an MCU Write.

Synchronous Update: Synchronous update mode is used to increment/decrement the timing offset value by a fixed amount. The synchronous updates can be performed in 4 different ways: (1) via event code. (2) via a user-programmable update time or (3) a single QBC increment/decrement at the end of a channel frame (4) or a user-programmable amount synchronized to the end of the channel frame.

When synchronous increment or decrement modes are selected and the TBE bit is set, writes to the QBCO master register are ignored. The slave register will (nominally) be either incremented or decremented (depending on how the SUE bits are set in the QBCO_UC1 register) at the end of the channel frame (**mqbc_count** = **trqbcm**). Logic is included to ensure that events scheduled at the channel frame boundary (**mqbc_count** = **trqbcm** & **mqbc_count** = 0) are not skipped or executed twice due to an offset value update.

1. Note that if **fetch_param**, the **end_of_macro** bus, and the **increment pointer bus** were all concatenated into a single bus, the entire bus would be "one-hot".
2. The **qbco** value must be less than or equal to the value of **trqbcm.**

Table 50-6 summarizes the asynchronous and all synchronous update modes. The highest priority update mode in the synchronous update via an event code. When the Event QBCO Update Enable bit (EQUE in the QBCO_UC0 register) is set to a logic 1, the master register in the QBCO Handler block transfers the new **qbco**[15:0] value to the slave on the next occurrence of event code, QBCO_UPDATE. When QBCO_UPDATE is executed, the EQUE bit is automatically reset to 0 (this prevents the QBCO_UPDATE from being re-executed due to the QBCO change). When EQUE is set to a logic 0, operation of the master and slave QBCO registers in the QBCO Handler block will be according to the SUE[2:0] bits.

When the MQSPI Trigger Enable bit (EMTE in QBCO_UC0 register) is set to a logic 1, an MQSPI trigger is asserted on one of the 32 MQSPI_TRIG[31:0] signals according to the 5-bit MQSPI Trigger number (MTRIG[4:0] in QBCO_UC1 register). When a QBCO_UPDATE event is executed, the EMTE bit is automatically reset to 0 (this prevents the QBCO_UPDATE from being re-executed due to the QBCO change). The EQUE bit and the EMTE bit work independently of each other, that is, either bit can be enabled without the other being enabled. Setting the EMTE bit does not override the SUE[2:0] bits.

The UPDATE_TIME_FRAME (QBCO_UDT_F) and an UPDATE_TIME_QBC (QBCO_UDT_Q register are used when SUE[2] = 1 (synchronous update at specified time). These registers provide a register-based synchronous update for the slave qbco[15:0]. The QBCO_UDT_F register contains the frame bits udf[1:0], which specify that the update will take place from 0-3 frames after SUE[2:0] is set to 1XX. When SUE[2:0] is set to 1XX, udf[1:0] is decremented by 1 and synchronized to the channel. The QBCO_UDT_Q register contains the variable udq[15:0], specifying the quarter-bit count at which to perform the update, The update occurs when udf[1:0] = 0 and mqbc[15:0] = udq[15:0]. The update automatically clears SUE[2:0] to 3'b000 (asynchronous mode).

Table 50-6. QBCO Update Summary - MCU

TBE	SUE[2:0]	EQUE	EMTE	Function
0	XXX	X	X	Asynchronous update
1	0 0 0	0	X	Asynchronous update
1	0 0 1	0	X	Increment QBCO by 1 on next channel frame interrupt
1	0 1 0	0	X	Decrement QBCO by 1 on next channel frame interrupt
1	0 1 1	0	X	Synchronous update on next channel frame interrupt
1	1 X X	0	X	Synchronous update at UPDATE_TIME_QBC
1	XXX	1	X	Update occurs when a QBCO_UPDATE event code is executed (not the next channel frame interrupt)
1	XXX	X	1	An MQSPI trigger is generated when a QBCO_UPDATE event code is executed

When SUE [2:0] bits = 011 the master register in the QBCO Handler block will transfer the new qbco[15:0] value to the slave register on the next Channel Frame Interrupt (cfi).

When a QBCO_UPDATE is executed and either the EQUE or EMTE bits are set, or when a synchronous update occurs because either or both of the Synchronous Update bits (SUE[1:0]) are set to 1, (SUE[2:0] = 001, 010, 011, or 1XX), the SYNC output signal will be asserted for 1 Quarter Bit Clock period.

The delay registers (EG_D0_QD, EG_D0_FD, etc.) are double buffered with a master and a slave. When a synchronous update occurs or a synchronous event update occurs, the contents in the master registers are copied to the slave registers. If the Synchronous Update Enable bits (SUE[2:0]) are configured for a synchronous update or the new Event QBCO Update Enable bit (EQUE) is set for an update, MCU writes to the delay registers will only write to the master registers. If synchronous updating is not enabled (SUE[2:0] = EQUE = 0), MCU writes to the delay registers writes both the slave and master registers simultaneously.

50.5 Pin Control Unit

The Pin Control Unit (PCU) is used to configure the L1 Timer control pin output drivers. As shown in Figure 50-10, control signals from the Event Generator can be bypassed with values from the default register (PD) on a bit-by-bit basis. The DTX_SELECT bit is combined with the PCU configuration bit in order to select between the Event Generator signal and the value in the bypass register. The MUX output can be read back from the PCU Status (PS) register. The PCU Tristate Configuration (PTC) register configures the tristate buffer of the external pins. The DTX_SELECT bits are contained in a 16-bit DSP R/W register in the L1 Timer.

The Event Generator Layer 1 control output drivers are initialized under the following circumstances:

- At RESET: The PD and pdout_eg registers are set to zero (the pdout_eg register is internal to the PCU and not accessible to the user). PCFG register is set to zero which forces the pin defaults of the PD register onto the tri-state drivers of the external pins. The PTC register is cleared, setting the corresponding tri-state drivers to a high impedance state.[1]
- After RESET, the pin defaults are forced to the tri-state drivers of the external pins until the PCFG is written with the initial configuration. This initial configuration must be written prior to enabling the Event Generator. Writing the PCFG register at this time will generate a one time load of the pin defaults of the PD register into the pdout_eg register. Subsequent writes to the PCFG register will not cause the pin defaults to be loaded into the pdout_eg register.
- Upon entering SLEEP mode, all bits of the PCFG configuration register are cleared which forces the pin defaults of the PD register onto the tri-state drivers of the external pins.(see Section 50.12.1, “Sleep.”). Prior to WAKEUP, the PCFG register should be re-initialized. However, before initializing this register, the DRC (Default Register Copy) bit should be asserted to assure that the pdout_eg register is loaded with the defaults from the PD register. See Section Table 50-80., “PD Description”.
- Upon WAKEUP, the pin defaults are again loaded into the pdout_eg register.

1. On the Neptune IC, PTC[15:0] are not connected.

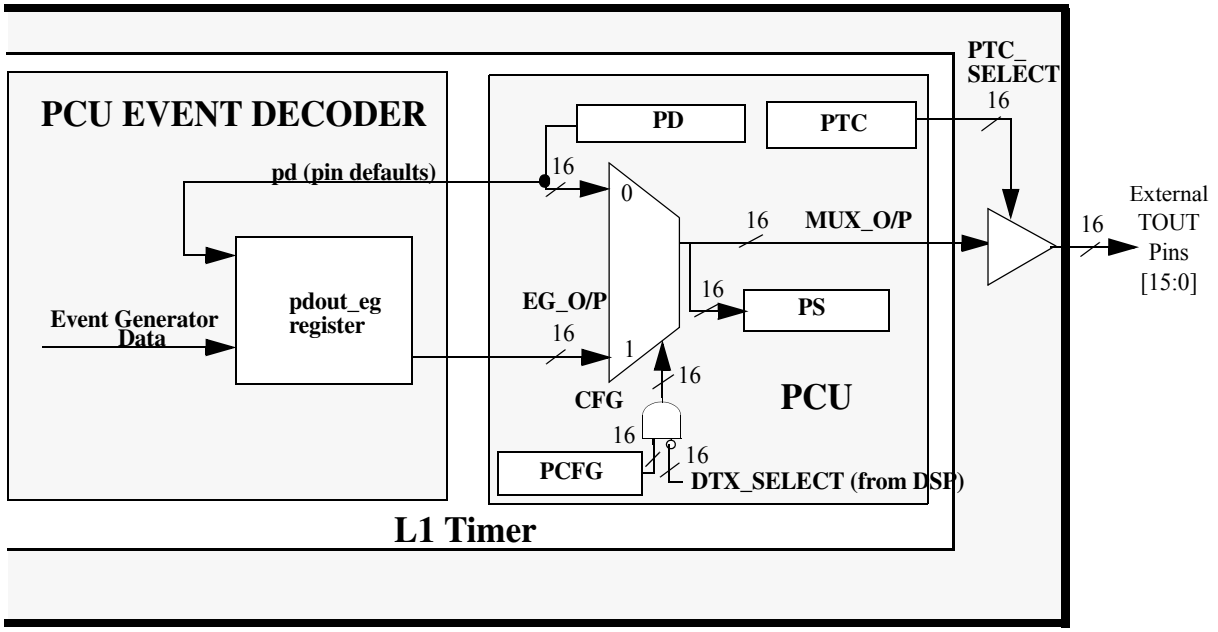


Figure 50-10. Pin Control Unit

There are 2 additional outputs from the PCU, PD0_INTERNAL and PD1_INTERNAL, shown in Figure 50-7. The PD0_INTERNAL signal maps to the L1T_DSP_INT block to generate the L1RX interrupt. PD1_INTERNAL is an always-active-high version of PD[1] and also maps to the L1T_DSP_INT block to generate the L1TX interrupt.

PD0_INTERNAL and PD1_INTERNAL can be masked using the PDIMASK register.

50.6 Error Detector

The Error Detector provides a means for detecting errors within the Layer 1 Timer. Error sources are individually maskable using the Error Masks (EM0, EM1 and EM2). Error sources can be found by reading the Error Source (ES0, ES1 and ES2) registers. The detectable errors are listed in Table 50-7. The behavior of the Layer 1 Timer when an error occurs is described in Table 50-8.

Comments on the response to error conditions:

1. Glitch errors[1] on Layer 1 control pins are handled by using priority assignments. The glitched pin will default to the value assigned it in the PCU bypass register. This will prevent the handset from creating a condition (such as leaving the transmitter on) that could cause a network error.
2. The Event Generator requires resynchronization with the Timing Block when it is reset or woken up from sleep mode. If resynchronization is required at any other time, a serious hardware malfunction has occurred.
3. If the START or END frame table pointer values exceed the Event Table RAM size (>=192) or if the END pointers are less than the START pointers, a Frame Table Boundary error occurs. An FTB error is indicated and the timer is automatically put into SLEEP mode.

1. A “glitch” occurs when a pin is programmed to both zero and a one during a quarter-bit count.

4. When the Event Decoder attempts to prefetch data from the Event Table and does not receive the proper handshaking signals from the Address Generation Unit, a prefetch error has occurred. The timer is automatically put into SLEEP mode.
5. SYNC, Frame Table Boundary, Prescaler Modulus, Macro C Call and prefetch errors are non-maskable.
6. The FATAL bit (bit 7 of the output bus TESTSIGS) is asserted if a Frame Table Boundary, SYNC, or prefetch error occurs.
7. If the Macro Table START or END pointers exceed the size of the Event Table RAM (>= 192) or the Macro Table pointer reaches the END pointer value without executing an EOM instruction, the Macro Table Boundary error will be generated. The Macro Table Pointer then resets itself to entry 0 in the macro table. These macro boundary errors are maskable by writing a 0 to the MABE, MBBE and MCBE bits in the EM0 register

Table 50-7. Layer 1 Timer: Detectable Errors

Source	Source Register	Mask Register	Description
Frame Table Decode Error	ES0	EM0	Illegal frame table event code loaded into Event Decoder. Note: Fetching Event codes B7h, B8h, and DFh, though reserved, do not generate a Frame Table Decode Error.
Macro A Decode Error	ES0	EM0	Illegal macro event code loaded into Event Decoder from Macro A.
Macro B Decode Error	ES0	EM0	Illegal macro event code loaded into Event Decoder from Macro B.
Macro C Decode Error	ES0	EM0	Illegal macro event code loaded into Event Decoder from Macro C.
Macro A Active Error	ES0	EM0	Macro A is already active when another Macro A is started.
Macro B Active Error	ES0	EM0	Macro B is already active when another Macro B is started.
Macro C Active Error	ES0	EM0	Macro C is already active when another Macro C is started.
Frame Table Write Error	ES0	EM0	The MCU attempts to overwrite events within the boundaries of the active frame table.
Boundary Pointer Write Error	ES0	EM0	The MCU attempts to overwrite a boundary pointer of the active frame table.
Macro A Write Error	ES0	EM0	MCU attempts to write to the Macro A Table while macro is active.
Macro B Write Error	ES0	EM0	MCU attempts to write to the Macro B Table while macro is active.
Macro C Write Error	ES0	EM0	MCU attempts to write to the Macro C Table while macro is active.
Frame Table Boundary Error	ES0	None	(ft0_start, ft0_end, ft1_start, ft1_end >= 192) (RAM size) ***OR*** (ft0_end < ft0_start) ***OR*** (ft1_end < ft1_start)

Table 50-7. Layer 1 Timer: Detectable Errors (Continued)

Source	Source Register	Mask Register	Description
Macro A Boundary Error	ES0	EM0	APTR reaches the last entry in Macro A w/o executing an End of Macro (EOM) event ***OR*** (ma_start >= 192) (RAM size) ***OR*** (ma_end >= 192)
Macro B Boundary Error	ES0	EM0	BPTR reaches the last entry of Macro B w/o executing an End of Macro (EOM) event ***OR*** (mb_start >= 192) (RAM size) ***OR*** (mb_end >= 192)
Macro C Boundary Error	ES0	EM0	CPTR reaches the last entry of Macro C w/o executing an End of Macro (EOM) event ***OR*** (mc_start >= 192) (RAM size) ***OR*** (mc_end >= 192)
PD[15:0] Glitch	ES1	ES1	Attempted to program PD[15:0] to both 0 and 1 on the same Quarter Bit.
Frame Table Prefetch Error	ES2	None	Event Decoder did not receive expected handshaking signals from the Address Generation Unit when trying to load data for the next Frame Table event.
Macro A Prefetch Error	ES2	None	Event Decoder did not receive expected handshaking signals from the Address Generation Unit when trying to load data for the next Macro A event.
Macro B Prefetch Error	ES2	None	Event Decoder did not receive expected handshaking signals from the Address Generation Unit when trying to load data for the next Macro B event.
Macro C Prefetch Error	ES2	None	Event Decoder did not receive expected handshaking signals from the Address Generation Unit when trying to load data for the next Macro C event.
Parameter Table Prefetch Error	ES2	None	Event Decoder did not receive expected handshaking signals from the Address Generation Unit when trying to load Parameter Table data when starting a macro.
Macro C Call Error	ES2	None	Macro C event code has been called while prescaler modulus < 15.
Prescaler Modulus Error	ES2	None	Prescaler modulus has been programmed to a value < 11.
SYNC error	ES2	None	Event Generator did not receive QBC mark from Timing Block at expected time.
Not End of Table	ES2	EM2	FPTR has not reached an END pointer when CFI is asserted.
Macro A Parameter Overflow	ES2	EM2	Initial value of pt_start pointer is greater than pt_end pointer ***OR*** (ptable ptr increment) ***AND*** (ma_ptable_entry = pt_end ***OR*** ma_ptable_entry >= 191)) (RAM size)
Macro B Parameter Overflow	ES2	EM2	Initial value of pt_start pointer is greater than pt_end pointer ***OR*** (ptable ptr increment ***AND*** (mb_ptable_entry = pt_end ***OR*** mb_ptable_entry >= 191)) (RAM size)
Macro C Parameter Overflow	ES2	EM2	Initial value of pt_start pointer is greater than pt_end pointer ***OR*** (ptable ptr increment) ***AND*** (mc_ptable_entry = pt_end ***OR*** mc_ptable_entry >= 191)) (RAM size)

Table 50-7. Layer 1 Timer: Detectable Errors (Continued)

Source	Source Register	Mask Register	Description
Macro A Loop Error	ES2	EM2	Macro A has called a second loop event code while the first Macro A loop is active.
Macro B Loop Error	ES2	EM2	Macro B has called a second loop event code while the first Macro B loop is active.
Macro C Loop Error	ES2	EM2	Macro C has called a second loop event code while the first Macro C loop is active.
Parameter Table Boundary Error	ES2	EM2	Parameter table pointers, pt_start or pt_end >= 192 (RMA size)

Table 50-8. Layer 1 Timer: Response to Error Conditions

Error Condition	Response
Decode Errors	Execute No Operation (NOP).
Macro Active Errors	Ignore 2nd start macro instruction (interpret as NOP). Proceed to next entry in Frame Table.
Table Write Errors	Permit MCU write.
Boundary Pointer Write Error	Permit MCU write.
Frame Table Boundary Error	The timer will go into SLEEP in IMMEDIATE mode, regardless of the contents of the Sleep Mode bits in the GTC1 register[1].
Macro Table A Boundary Error	Set APTR to Macro A entry 0.
Macro Table B Boundary Error	Set BPTR to Macro B entry 0.
Macro Table C Boundary Error	Set CPTR to Macro C entry 0.
Glitch Errors	Use pin value assigned in PCU Bypass register.
Prefetch Errors	The timer will go into SLEEP in IMMEDIATE mode, regardless of the contents of the Sleep Mode bits in the GTC1 register.
Macro C Call Error	The timer will go into SLEEP in IMMEDIATE mode, regardless of the contents of the Sleep Mode bits in the GTC1 register.
Prescaler Modulus Error	The timer will go into SLEEP in IMMEDIATE mode, regardless of the contents of the Sleep Mode bits in the GTC1 register.
SYNC Error	The timer will go into SLEEP in IMMEDIATE mode, regardless of the contents of the Sleep Mode bits in the GTC1 register.
Not End of Table	Continue executing active frame table.
Parameter Overflow Errors	Set parameter pointer to pt_start if overflow ***OR*** execute no operation if Macro call has relative pointer starting value > parameter table size.
Macro Loop Errors	Execute No Operation (NOP).
Parameter Table Boundary Error	Execute No Operation (NOP).

1. A description of the different SLEEP modes is contained in Table 50-22 (“Global Timer Control 1 Register”).

50.7 MCU Interrupt Generation

Interrupt sources are summarized in Table 50-9.

Interrupt sources are masked through either the Interrupt Generator Mask (IGM) or MCU Interrupt Mask (MCUIM). All interrupts except TERI can be masked. To prevent generation of TERI, detection of error sources can be blocked using the Error Source Masks. See Section 50.6, “Error Detector”, for more information.

The Channel Interrupt (CI), FFI[1], and MCU Interrupts are the logical OR of multiple interrupt sources. All MCU interrupts are active low.

Interrupt status can be read from the Global Timer Status 1(GTS1) register. When the TERI interrupt is generated, the error source can be read from the Error Source (ES) registers.

Table 50-9. MCU Interrupt Generation: Source/Port Map

OUTPUT PORT	LEVEL/ EDGE	INTERRUPT SOURCE	FROM	DESCRIPTION
Reference Frame Interrupt (RFI)	Active Low	Reference Frame Interrupt (RFI)	TB[1]	RFI will be generated when the Reference Quarter Bit Counter rolls over. See Section 50.3.2 for more information
Reference Frame Counter Interrupt (RFCI)	Active Low	Reference Frame Counter Interrupt (RFCI)	TB	RFCI is used to generate a wake-up for the exec after the processor has been asleep. RFCI is requested when the Reference Frame Counter expires. See Section 50.3.5 for more information
Channel Interrupt (CI)	Active Low	Channel Frame Interrupt (CFI)	TB	CFI will be requested when the Matching Quarter Bit Count rolls over. See Section 50.3.3 for more information.
		Channel Frame Counter Interrupt (CFCI)	TB	CFCI is used to generate a Layer 1 wake-up after the processor has been asleep. CFCI is requested when the Channel Frame Counter expires. See Section 50.3.4 for more information.
MCU Interrupt 0 (MCU_INT0)	Active Low	Programmable Interrupt Timer Interrupt (PITI)	TB	The Periodic Interrupt Timer (PIT) generates a free-running periodic interrupt (PITI) that is routed to the MCU. See Section 50.3 for more information.
		Programmable Wakeup Timer Interrupt (PWTI)	Reset, Sleep, and Wakeup Logic	Generated when the Global Programmable Wakeup Timer counts down to zero. See Section 50.12 for more information.
		Timer Error Interrupt (TERI)	ED	Timer debug interrupt. TERI is generated if the Error Detector finds an error condition. See Section 50.6 for more information.

1. The FFI output is not connected in the Neptune IC.

Table 50-9. MCU Interrupt Generation: Source/Port Map

OUTPUT PORT	LEVEL/ EDGE	INTERRUPT SOURCE	FROM	DESCRIPTION
		DSP Vectored Interrupts Indicator (DVII)	EG	Asserted if any of the (unmasked) DVI14:DVI0 event codes is asserted.
MCU Interrupt 1 (MCU_INT1)	Active Low	MCU Interrupts from event codes (MCUI[15:0])	EG	Bit-wise OR of call-processor interrupts generated from MCUI15:MCUI0 event codes.

1. TB = Timing Block, EG = Event Generator, ED = Error Detector.

50.8 DSP Interrupt Handler

The L1T_DSP_INT block handles generation of DSP interrupt vectors. The block takes interrupt requests, holds these requests in a status register (transparent to the user), prioritizes them, asserts a DSP interrupt vector on the VAB bus, and asserts the DSP interrupt presence signal. DSP servicing of an interrupt clears the interrupt presence line and the status bit of the corresponding interrupt. If other interrupt sources are simultaneously requesting interrupt service, their requests are latched and held until all higher priority interrupts are serviced. Multiple requests to a pending interrupt are not stacked.

NOTE: The MCU interrupts are hold status, then mask. DSP interrupts are mask, then hold status.

The DSP L1RX and L1TX interrupts (see Table 50-10) are generated based on transitions of the PD0_INTERNAL and PD1_INTERNAL pins, respectively. These interrupts can be masked and transition edges selected via the PDIMASK register.

The DVI14:DVI0 interrupts can be masked via the DVIM register. Assertion of any of the unmasked DVI14:DVI0 interrupts will assert the DSP Vectored Interrupt Indicator (DVII) interrupt on the MCU side.

Note that the DSPWAKE interrupt is handled as a separate case, since it is routed to the IRQA port of the MDI

Table 50-10. DSP Interrupt Generation: Source/Port Map

Interrupt Name	Priority	Vector Number	Source	Description
Layer 1 Receive (L1RX)	1	70	EG	Layer 1 receive interrupt generated on edge transition of PD0_INTERNAL
Layer 1 Transmit (L1TX)	2	72	EG	Layer 1 transmit interrupt generated on edge transition of PD1_INTERNAL
DSP Vectored Interrupts (DVI[14:0])	4:17	76-92	EG	General purpose DSP interrupts generated from DVI14:DVI0 event codes.(DVI0 is highest priority, DVI14 lowest)

50.9 Bus Interfaces

The MCU communicates with the Layer 1 Timer via the IP bus interface. The Event Table RAMs are accessed directly through the IP bus. All other timer memory locations are accessed via a IP interface.

The important features of the bus interfaces to the Layer 1 Timer are:

- 4K address space.
- Big Endian Memory format.
- 16-bit accesses supported on word-boundaries
- MCU can access all registers and even if:
  - — The timer is disabled via the DSM signal DIS_L1T
  - — The timer also disabled via the MST_EN bit in the MSTR register.
- When disabled with MST_EN, L1 timer clocks off and R-only access to MCU-accessible L1 registers, except for MSTR register, which is R/W at all times.)

NOTE:

Read only access to MCU-accessible L1 registers is not enforced by hardware, however, attempting to write to these registers with MSTR_EN disabled may produce unpredictable operation.

The IP interface handles two main functions: IP to pbus translation, and bus synchronization. The layer 1 timer module design is re-used from the Whitecap IC. The peripheral bus interface on the Whitecap (pbus) was significantly different than the IP bus. The IP interface was designed to ease the translation between the two bus definitions while minimizing the amount of re-design done to the pbus.

The second function performed by the IP interface is bus synchronization. The main timebase for the Layer 1 Timer is PAT_REF. The IP interface design allows the Layer 1 Timer to run from the reference input frequency (PAT_REF), while still communicating to the IP bus running at the IP bus frequency. This benefits the system by allowing the software to select any frequency for the IP bus (within the specified operating range) independent of the layer 1 timer module. The IP interface's negative effect on the system is that the access time will vary depending on the phase relationship between the reference input (PAT_REF) and the IP bus clock (C1, C2). If PAT_REF is the same frequency and inverted to the IP bus clock, IPS_XFR_WAIT will not be asserted and the access time will be based on the platform's IP bus protocol. If the IP bus clock is greater than PAT_REF, then IPS_XFR_WAIT will assert until one clock cycle of PAT_REF has passed.

NOTE:

For the IP bus to communicate with the IP interface properly, the IP bus clock must always run at a frequency greater than or equal to the reference frequency. The IP bus clock and the reference frequency must be edge-aligned.

The RAMs are accessed via a 3-cycle MIG interface. Use of the MIG allows the MCU to access the RAMs in 3 cycles, rather than the 4-6 imposed by the JIG. By using dual-port RAM, the MCU and the Layer 1 timer can simultaneously access the same Event Table address[1]. The MCU can access the timer RAM even if the timer is disabled via the DSM or the MSTR register.

1. However, in the case of simultaneous MCU/Layer 1 attempts to access the TRQBC, TRAFN, TRAFN_L, TCFN, or TCFN_L registers, the MCU has priority.

50.9.1 Invalid Addresses

The PINVLD signal is an internal active-high signal going to the JIG interface used for invalid address detection. When PINVLD is asserted, the JIG will assert the MTEA_B_Z signal. The Layer 1 timer will assert PINVLD under the following conditions:

- Attempting an access an undefined address space.
- Attempting to write to a read-only register.
- Attempting to write to the EG_QBCO register while it is in “1 QBC Synchronous Update” mode (SUE[1] ^ SUE[0] in the QBCO_UC1 register), Event QBCO update is inactive (EQUE = 0 in the QBCO_UC0 register) and the Timing Block is on (TBE = 1 in the GTC1 register).

50.10 Reset, Sleep, and Wakeup Logic

The operating mode of the Layer 1 Timer is specified by the state of the external pin NRST and the SRST, TBE, and EGE bits in the Global Timer Control (GTC) register. The various operating modes are shown in Table 50-11.

RESET = Software Reset True (SRST) OR Hardware Reset True (NRST)

SRST = Software Reset Bit in the Global Timer Control 0 (GTC0) register

NRST = Active low hardware reset input.

EGE = Event Generator Enable bit in GTS0 status register

TBE = Timing Block Enable bit in GTC1control register

Table 50-11. Layer 1 Timer: Operating Modes

RESET	TBE	EGE	Operating Mode
1	X	X	L1 Timer Reset.
0	0	0	TIMER_OFF. The L1 Timer system clock is gated off to the Timing Block and the Event Generation Block.
0	0	1	UNDEFINED. The L1 Timer system clock is gated off to the Timing Block and on to the Event Generator.
0	1	0	SLEEP. The L1 Timer system clock is gated on to the Timing Block but off to the Event Generator. This mode maintains reference and channel timing but powers down the bulk of the timer.
0	1	1	TIMER_ON The Timing Block and Event Generator are both enabled.

50.10.1 Reset

Asserting RESET has the following effects:

- All interrupt status bits are cleared in the GTS1 register.
- All interrupts are masked in the IGM register.
- All error-source bits are cleared in the ES0, ES1 and ES2 registers.
- All error sources are masked in the EM0, EM1 and EM2 registers.

- All bits of the PTC register are cleared. This sets all bits of the PTC_SELECT output to the "high-impedance" setting.
- All bits of the PD default register are cleared.
- The hyperframe counters RAFN and CFN are cleared.
- The frame counters RFC and CFC are initialized.
- The Reference Quarter Bit Counter (**rqbc_count**) and Quarter-Bit Count Offset (**qbco**) are cleared.
- The RCLOCK and CCLOCK frame clocks are pulled to zero.
- The Event Generator is put to SLEEP in IMMEDIATE mode (see Section 50.12.1).
- All bits in the GTC0 and GTC1 control registers are cleared. Since the Event Generator has been put to SLEEP, the timer is left in "TIMER_OFF" mode when RESET is de-asserted.
- All table pointers are set to their default values.

The L1 timer will be held in RESET for as long as the hardware NRST pin is asserted. However, when the SRST bit is asserted, the timer is only held in RESET mode for one L1 Timer system clock cycle, since RESET automatically clears all bits in the GTC0 register (SRST included).

Note that Event Table RAM is not cleared upon RESET. It is recommended that unused entries should have their EC fields set to NOP (for frame table) or EOM (for macro table). If the unused frame table entries have their EQBC values set to a value greater than the value in the TRQBCM register, the event can never execute.

50.10.2 Wakeup

There are two methods for waking up the Event Generator:

- Synchronous with the channel frame.
- Via a programmable timeout.

For either method, the user should make sure that the following configuration steps have been taken:

- Frame table 0 and frame table 1 boundaries have been defined in the EG_FT0_CFG and EG_FT1_CFG registers.
- Frame table, macro table, and parameter table entries are correct.
- The Layer 1 control pin defaults are set in the PD register.
- The Layer 1 control pins are configured for Event Generator signals in the PCFG register.

At the time of wakeup, the Event Generator will:

- Turn on its system clock.
- Fetch the Event Code and EQBC of frame table 0 entry 0.
- Fetch the Event Code and REQBC of Macro A entry 0.
- Fetch the Event Code and REQBC of Macro B entry 0.
- Fetch the Event Code and REQBC of Macro C entry 0 if TBPREM register value >= 15.
- If necessary, initialize the Event Generator's Layer 1 control outputs (**pdout_eg[15:0]**) by setting the output drivers to the values stored in the PD register of the Pin Control Unit. See Section 50.5, "Pin Control Unit", for an explanation of output driver initialization conditions.

50.11 Synchronized Wakeup

The Event Generator can be woken up by asserting the WAKE bit in the GTC0 control register. When WAKEUP is asserted, the Event Generator must first resynchronize itself with the channel frame. At the end of the channel frame (**mqbc** == **trqbcm**), the Event Generator's system clock will be turned on and the Event Generator will go through it's wakeup sequence. The user can cancel a WAKEUP any time before **mqbc** == **trqbcm** by deasserting the WAKEUP bit.

50.12 Programmable Timeout

The Event Generator can be woken up at any given QBC in the next 32768 QBC's by using the Global Programmable Wakeup Timer (GPWT). The GPWT register consists of a 15-bit counter value field (**pwt_count**) and an enable bit (PWEN).

The user can calculate the current channel frame time (**mqbc**) by reading the **rqbc** value and adding **qbco** (modulo **trqbcm**). The **pwt_count** will then be the difference between the desired wakeup time and the current **mqbc**. Asserting PWEN will decrement pwt_count once per QBC.

When pwt_count reaches zero, an Event Generator wakeup is executed on the following QBC pulse, a PWT interrupt is generated, and PWEN is cleared. A programmable wakeup can be cancelled any time before pwt_count == 0 by clearing PWEN.

50.12.1 Sleep

The Layer 1 timer can disable the Event Generator by asserting the SLEEP bit in the Global Timer Control 0 (GTC0) register or by executing the GOTOSLEEP event code from either a frame or macro table. Entering SLEEP via the event code is done immediately. Entering SLEEP via the SLEEP bit depends on the settings of the SLEEP MODE (SM) bits in the GTC1 register. When SM=00, the Event Generator is put to sleep immediately. When SM=01, the Event Generator will be put to sleep at the start of the next event. When SM=10, the Event Generator is put to sleep at the end of the current reference frame. When SM=11, the Event Generator is put to sleep 1 QBC prior the next channel frame. Entering SLEEP has the following effects:

- All bits of the PCFG configuration register are cleared. This sets the PCU MUXs to select the default register.
- The FPTR pointer in the AGU is set to 0x0000.
- The APTR pointer in the AGU is set to Macro A event 0.
- The BPTR pointer in the AGU is set to Macro B event 0.
- The CPTR pointer in the AGU is set to Macro C event 0.
- The L1 Timer system clock to the Event Generator is turned off, and the EGE bit in the GTS0 status register is de-asserted. The timer is now in "SLEEP" mode.
- The SLEEP bit in the GTC0 register is automatically de-asserted.

50.13 Event Codes

The 8-bit event codes are the instructions that tell the Event Generator what actions to perform. The event code definition is shown in Table 50-12 for all 255 possible event codes. The tables also designates from what type of table (frame / macro) the event code may be executed.

Table 50-12. Neptune Event Codes

Event Code	Event Name	Executable From*:	Description
$00	MACRO_A_0	F	Start macro A with parameter table entry #0
$01	MACRO_A_2	F	Start macro A with parameter table entry #2
$02	MACRO_A_4	F	Start macro A with parameter table entry #4
$03	MACRO_A_6	F	Start macro A with parameter table entry #6
$04	MACRO_A_8	F	Start macro A with parameter table entry #8
$05	MACRO_A_10	F	Start macro A with parameter table entry #10
$06	MACRO_A_12	F	Start macro A with parameter table entry #12
$07	MACRO_A_14	F	Start macro A with parameter table entry #14
$08	MACRO_A_16	F	Start macro A with parameter table entry #16
$09	MACRO_A_18	F	Start macro A with parameter table entry #18
$0A	MACRO_A_20	F	Start macro A with parameter table entry #20
$0B	MACRO_A_22	F	Start macro A with parameter table entry #22
$0C	MACRO_A_24	F	Start macro A with parameter table entry #24
$0D	MACRO_A_26	F	Start macro A with parameter table entry #26
$0E	MACRO_A_28	F	Start macro A with parameter table entry #28
$0F	MACRO_A_30	F	Start macro A with parameter table entry #30
$10	MACRO_B_0	F	Start macro B with parameter table entry #0
$11	MACRO_B_2	F	Start macro B with parameter table entry #2
$12	MACRO_B_4	F	Start macro B with parameter table entry #4
$13	MACRO_B_6	F	Start macro B with parameter table entry #6
$14	MACRO_B_8	F	Start macro B with parameter table entry #8
$15	MACRO_B_10	F	Start macro B with parameter table entry #10
$16	MACRO_B_12	F	Start macro B with parameter table entry #12
$17	MACRO_B_14	F	Start macro B with parameter table entry #14
$18	MACRO_B_16	F	Start macro B with parameter table entry #16

Table 50-12. Neptune Event Codes (Continued)

Event Code	Event Name	Executable From*:	Description
$19	MACRO_B_18	F	Start macro B with parameter table entry #18
$1A	MACRO_B_20	F	Start macro B with parameter table entry #20
$1B	MACRO_B_22	F	Start macro B with parameter table entry #22
$1C	MACRO_B_24	F	Start macro B with parameter table entry #24
$1D	MACRO_B_26	F	Start macro B with parameter table entry #26
$1E	MACRO_B_28	F	Start macro B with parameter table entry #28
$1F	MACRO_B_30	F	Start macro B with parameter table entry #30
$20	MACRO_C_0	F	Start macro C with parameter table entry #0
$21	MACRO_C_2	F	Start macro C with parameter table entry #2
$22	MACRO_C_4	F	Start macro C with parameter table entry #4
$23	MACRO_C_6	F	Start macro C with parameter table entry #6
$24	MACRO_C_8	F	Start macro C with parameter table entry #8
$25	MACRO_C_10	F	Start macro C with parameter table entry #10
$26	MACRO_C_12	F	Start macro C with parameter table entry #12
$27	MACRO_C_14	F	Start macro C with parameter table entry #14
$28	MACRO_C_16	F	Start macro C with parameter table entry #16
$29	MACRO_C_18	F	Start macro C with parameter table entry #18
$2A	MACRO_C_20	F	Start macro C with parameter table entry #20
$2B	MACRO_C_22	F	Start macro C with parameter table entry #22
$2C	MACRO_C_24	F	Start macro C with parameter table entry #24
$2D	MACRO_C_26	F	Start macro C with parameter table entry #26
$2E	MACRO_C_28	F	Start macro C with parameter table entry #28
$2F	MACRO_C_30	F	Start macro C with parameter table entry #30
$30	MREG_0	F	Store MQSPI[0] trigger code in MREG register
$31	MREG_1	F	Store MQSPI[1] trigger code in MREG register
$32	MREG_2	F	Store MQSPI[2] trigger code in MREG register
$33	MREG_3	F	Store MQSPI[3] trigger code in MREG register
$34	MREG_4	F	Store MQSPI[4] trigger code in MREG register
$35	MREG_5	F	Store MQSPI[5] trigger code in MREG register

Table 50-12. Neptune Event Codes (Continued)

Event Code	Event Name	Executable From*:		Description
$36	MREG_6	F		Store MQSPI[6] trigger code in MREG register
$37	MREG_7	F		Store MQSPI[7] trigger code in MREG register
$38	MREG_8	F		Store MQSPI[8] trigger code in MREG register
$39	MREG_9	F		Store MQSPI[9] trigger code in MREG register
$3A	MREG_10	F		Store MQSPI[10] trigger code in MREG register
$3B	MREG_11	F		Store MQSPI[11] trigger code in MREG register
$3C	MREG_12	F		Store MQSPI[12] trigger code in MREG register
$3D	MREG_13	F		Store MQSPI[13] trigger code in MREG register
$3E	MREG_14	F		Store MQSPI[14] trigger code in MREG register
$3F	MREG_15	F		Store MQSPI[15] trigger code in MREG register
$40	MREG_16	F		Store MQSPI[16] trigger code in MREG register
$41	MREG_17	F		Store MQSPI[17] trigger code in MREG register
$42	MREG_18	F		Store MQSPI[18] trigger code in MREG register
$43	MREG_19	F		Store MQSPI[19] trigger code in MREG register
$44	MREG_20	F		Store MQSPI[20] trigger code in MREG register
$45	MREG_21	F		Store MQSPI[21] trigger code in MREG register
$46	MREG_22	F		Store MQSPI[22] trigger code in MREG register
$47	MREG_23	F		Store MQSPI[23] trigger code in MREG register
$48	MREG_24	F		Store MQSPI[24] trigger code in MREG register
$49	MREG_25	F		Store MQSPI[25] trigger code in MREG register
$4A	MREG_26	F		Store MQSPI[26] trigger code in MREG register
$4B	MREG_27	F		Store MQSPI[27] trigger code in MREG register
$4C	MREG_28	F		Store MQSPI[28] trigger code in MREG register
$4D	MREG_29	F		Store MQSPI[29] trigger code in MREG register
$4E	MREG_30	F		Store MQSPI[30] trigger code in MREG register
$4F	MREG_31	F		Store MQSPI[31] trigger code in MREG register
$50	MQSPI_0	F	M	Assert MQSPI[0] trigger
$51	MQSPI_1	F	M	Assert MQSPI[1] trigger
$52	MQSPI_2	F	M	Assert MQSPI[2] trigger

Table 50-12. Neptune Event Codes (Continued)

Event Code	Event Name	Executable From*:		Description
$53	MQSPI_3	F	M	Assert MQSPI[3] trigger
$54	MQSPI_4	F	M	Assert MQSPI[4] trigger
$55	MQSPI_5	F	M	Assert MQSPI[5] trigger
$56	MQSPI_6	F	M	Assert MQSPI[6] trigger
$57	MQSPI_7	F	M	Assert MQSPI[7] trigger
$58	MQSPI_8	F	M	Assert MQSPI[8] trigger
$59	MQSPI_9	F	M	Assert MQSPI[9] trigger
$5A	MQSPI_10	F	M	Assert MQSPI[10] trigger
$5B	MQSPI_11	F	M	Assert MQSPI[11] trigger
$5C	MQSPI_12	F	M	Assert MQSPI[12] trigger
$5D	MQSPI_13	F	M	Assert MQSPI[13] trigger
$5E	MQSPI_14	F	M	Assert MQSPI[14] trigger
$5F	MQSPI_15	F	M	Assert MQSPI[15] trigger
$60	MQSPI_16	F	M	Assert MQSPI[16] trigger
$61	MQSPI_17	F	M	Assert MQSPI[17] trigger
$62	MQSPI_18	F	M	Assert MQSPI[18] trigger
$63	MQSPI_19	F	M	Assert MQSPI[19] trigger
$64	MQSPI_20	F	M	Assert MQSPI[20] trigger
$65	MQSPI_21	F	M	Assert MQSPI[21] trigger
$66	MQSPI_22	F	M	Assert MQSPI[22] trigger
$67	MQSPI_23	F	M	Assert MQSPI[23] trigger
$68	MQSPI_24	F	M	Assert MQSPI[24] trigger
$69	MQSPI_25	F	M	Assert MQSPI[25] trigger
$6A	MQSPI_26	F	M	Assert MQSPI[26] trigger
$6B	MQSPI_27	F	M	Assert MQSPI[27] trigger
$6C	MQSPI_28	F	M	Assert MQSPI[28] trigger
$6D	MQSPI_29	F	M	Assert MQSPI[29] trigger
$6E	MQSPI_30	F	M	Assert MQSPI[30] trigger
$6F	MQSPI_31	F	M	Assert MQSPI[31] trigger

Table 50-12. Neptune Event Codes (Continued)

Event Code	Event Name	Executable From*:	Description
$70	QBCO_UPDATE	F	Activate Quarter Bit Count Offset update (if enabled)
$71–$79	Reserved Event Codes		Flag error if timer tries to execute these Event Codes
$7A	EOM	M	End of macro
$7B	LOOP_TBL	M	Start loop using loop count stored in parameter table
$7C	LOOP_REG0	M	Start loop using loop count stored in Loop Register 0
$7D	LOOP_REG1	M	Start loop using loop count stored in Loop Register 1
$7E	LOOP_REG2	M	Start loop using loop count stored in Loop Register 2
$7F	LOOP_REG3	M	Start loop using loop count stored in Loop Register 3
$80	LOOP_0	M	Special loop case that ends the macro (similar to EOM)
$81	LOOP_1	M	Start loop using loop count of 1
$82	LOOP_2	M	Start loop using loop count of 2
$83	LOOP_3	M	Start loop using loop count of 3
$84	LOOP_4	M	Start loop using loop count of 4
$85	LOOP_5	M	Start loop using loop count of 5
$86	LOOP_6	M	Start loop using loop count of 6
$87	LOOP_7	M	Start loop using loop count of 7
$88	LOOP_8	M	Start loop using loop count of 8
$89	LOOP_9	M	Start loop using loop count of 9
$8A	LOOP_10	M	Start loop using loop count of 10
$8B	LOOP_11	M	Start loop using loop count of 11
$8C	LOOP_12	M	Start loop using loop count of 12
$8D	LOOP_13	M	Start loop using loop count of 13
$8E	LOOP_14	M	Start loop using loop count of 14
$8F	LOOP_15	M	Start loop using loop count of 15
$90	END_LOOP	M	Decrements loop count and repeats loop if not zero
$91	DELAY_TBL	M	Activate delay using delay stored in parameter table
$92	DELAY_REG0	M	Activate delay loaded from Delay Register 0
$93	DELAY_REG1	M	Activate delay loaded from Delay Register 1

Table 50-12. Neptune Event Codes (Continued)

Event Code	Event Name	Executable From*:		Description
$94	DELAY_REG2		M	Activate delay loaded from Delay Register 2
$95	DELAY_REG3		M	Activate delay loaded from Delay Register 3
$96	USE_MQSPI		M	Assert MQSPI trigger stored in MREG
$97	PD0-0	F	M	Reset the Event Generator pin driver 0 (TOUT1 pin)
$98	PD0-1	F	M	Set the Event Generator pin driver 0 (TOUT1 pin)
$99	PD1-0	F	M	Reset the Event Generator pin driver 1 (TOUT2 pin)
$9A	PD1-1	F	M	Set the Event Generator pin driver 1 (TOUT2 pin)
$9B	PD2-0	F	M	Reset the Event Generator pin driver 2 (TOUT3 pin)
$9C	PD2-1	F	M	Set the Event Generator pin driver 2 (TOUT3 pin)
$9D	PD3-0	F	M	Reset the Event Generator pin driver 3 (TOUT4 pin)
$9E	PD3-1	F	M	Set the Event Generator pin driver 3 (TOUT4 pin)
$9F	PD4-0	F	M	Reset the Event Generator pin driver 4 (TOUT5 pin)
$A0	PD4-1	F	M	Set the Event Generator pin driver 4 (TOUT5 pin)
$A1	PD5-0	F	M	Reset the Event Generator pin driver 5 (TOUT6 pin)
$A2	PD5-1	F	M	Set the Event Generator pin driver 5 (TOUT6 pin)
$A3	PD6-0	F	M	Reset the Event Generator pin driver 6 (TOUT7 pin)
$A4	PD6-1	F	M	Set the Event Generator pin driver 6 (TOUT7 pin)
$A5	PD7-0	F	M	Reset the Event Generator pin driver 7 (TOUT8 pin)
$A6	PD7-1	F	M	Set the Event Generator pin driver 7 (TOUT8 pin)
$A7	PD8-0	F	M	Reset the Event Generator pin driver 8 (TOUT9 pin)
$A8	PD8-1	F	M	Set the Event Generator pin driver 8 (TOUT9 pin)
$A9	PD9-0	F	M	Reset the Event Generator pin driver 9 (TOUT10 pin)
$AA	PD9-1	F	M	Set the Event Generator pin driver 9 (TOUT10 pin)
$AB	PD10-0	F	M	Reset the Event Generator pin driver 10 (TOUT11 pin)
$AC	PD10-1	F	M	Set the Event Generator pin driver 10 (TOUT11 pin)
$AD	PD11-0	F	M	Reset the Event Generator pin driver 11 (TOUT12 pin)
$AE	PD11-1	F	M	Set the Event Generator pin driver 11 (TOUT12 pin)

Table 50-12. Neptune Event Codes (Continued)

Event Code	Event Name	Executable From*:		Description
$AF	PD12-0	F	M	Reset the Event Generator pin driver 12 (TOUT13 pin)
$B0	PD12-1	F	M	Set the Event Generator pin driver 12 (TOUT13 pin)
$B1	PD13-0	F	M	Reset the Event Generator pin driver 13 (TOUT14 pin)
$B2	PD13-1	F	M	Set the Event Generator pin driver 13 (TOUT14 pin)
$B3	PD14-0	F	M	Reset the Event Generator pin driver 14 (TOUT15 pin)
$B4	PD14-1	F	M	Set the Event Generator pin driver 14 (TOUT15 pin)
$B5	PD15-0	F	M	Reset the Event Generator pin driver 15 (TOUT16 pin)
$B6	PD15-1	F	M	Set the Event Generator pin driver 15 (TOUT16 pin)
$B7	Not Used			
$B8	Not Used			
$B9	DSPWAKE	F	M	Wake up DSP
$BA	GOTOSLEEP	F	M	Shut down Event Generator
$BB	NOP	F	M	No Operation
$BC-$BF	Reserved			Flag error if timer tries to execute these Event Codes
$C0	MCUI0	F	M	MCU Interrupt 0
$C1	MCUI1	F	M	MCU Interrupt 1
$C2	MCUI2	F	M	MCU Interrupt 2
$C3	MCUI3	F	M	MCU Interrupt 3
$C4	MCUI4	F	M	MCU Interrupt 4
$C5	MCUI5	F	M	MCU Interrupt 5
$C6	MCUI6	F	M	MCU Interrupt 6
$C7	MCUI7	F	M	MCU Interrupt 7
$C8	MCUI8	F	M	MCU Interrupt 8
$C9	MCUI9	F	M	MCU Interrupt 9
$CA	MCUI10	F	M	MCU Interrupt 10
$CB	MCUI11	F	M	MCU Interrupt 11
$CC	MCUI12	F	M	MCU Interrupt 12

Table 50-12. Neptune Event Codes (Continued)

Event Code	Event Name	Executable From*:		Description
$CD	MCUI13	F	M	MCU Interrupt 13
$CE	MCUI14	F	M	MCU Interrupt 14
$CF	MCUI15	F	M	MCU Interrupt 15
$D0	DVI0	F	M	Vectored DSP Interrupt 0
$D1	DVI1	F	M	Vectored DSP Interrupt 1
$D2	DVI2	F	M	Vectored DSP Interrupt 2
$D3	DVI3	F	M	Vectored DSP Interrupt 3
$D4	DVI4	F	M	Vectored DSP Interrupt 4
$D5	DVI5	F	M	Vectored DSP Interrupt 5
$D6	DVI6	F	M	Vectored DSP Interrupt 6
$D7	DVI7	F	M	Vectored DSP Interrupt 7
$D8	DVI8	F	M	Vectored DSP Interrupt 8
$D9	DVI9	F	M	Vectored DSP Interrupt 9
$DA	DVI10	F	M	Vectored DSP Interrupt 10
$DB	DVI11	F	M	Vectored DSP Interrupt 11
$DC	DVI12	F	M	Vectored DSP Interrupt 12
$DD	DVI13	F	M	Vectored DSP Interrupt 13
$DE	DVI14	F	M	Vectored DSP Interrupt 14
$DF–$FF	Reserved Event Codes			

* F = Frame Table, M = Macro Table

50.14 MCU Memory Map Definitions

The following section contains the definitions of MCU memory map including the Event and EQBC RAM as well as MCU accessible registers in the L1 Timer. This section does not define the functional operation of the modules except for what is needed by the end user.

50.14.1 Frame Table RAM

The 192 entry Event Table RAM block supports two frame tables. Table locations are dynamically configurable via the Frame Table Configuration Registers (EG_FT0_CFG, EG_FT1_CFG). Each Frame Table Configuration Register consists of a START pointer (ft0_start, ft1_start) to define the first table entry, and an END pointer (ft0_end, ft1_end) to define the last table entry. The NEXT FRAME TABLE bits (ft0_next, ft1_next) are stored in the Next Frame Table Configuration Register (EG_NFT_CFG).

Each entry in the frame table RAM block contains two fields: an 8-bit Event Code (EC) and a 16-bit Event Quarter Bit Count (EQBC). The EC specifies what action (event) the L1 timer is to execute. The EQBC field specifies the event's execution time relative to the start of the channel frame. The EQBC address is offset by $200 from it's corresponding EC address. For example, the Event Code located at address $2485_3000 would use the Quarter Bit Count located at address $2485_3200.

At any given time, only one of the frame tables can be active. The Layer 1 timer architecture is designed such that the programmer should not have to overwrite entries in the active table. However, there is no hardware to prevent active table overwrites.

When the last event in the active table is executed, the next active table is selected by the ft0_next/ft1_next bits in the **EG_NFT_CFG** register. Depending on the value of this bit, the user can either alternate between frame tables or repeat the same table. For example, assume frame table 0 is executing and ft0_next = 1 (where 1 specifies Frame Table 1). When Frame Table 0 is finished, at the beginning of the next channel frame, Frame Table 1 will become the active frame table. However, if ft0_next = 0 (where 0 specifies Frame Table 0), Frame Table 0 would be repeated. The ft0_next/ft1_next bit of the active frame table can be changed anytime during the channel frame up until two counts before the modulus count, else it will interfere with the event fetch sequence.

FTEC Frame Table Event Code **Addr 2485_3000-2485_317E**

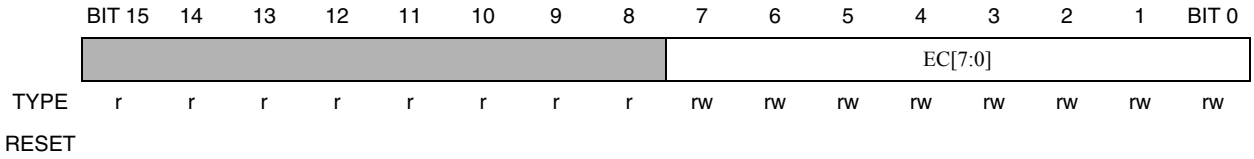


Table 50-13. FTEC Description

Name	Description	Settings
Bits 15-8	**Reserved**	N/A
EC[7:0] Bits 7-0	**Event Code—** EC contains the Event Code to be executed by the L1 Timer's Event Generator. Refer to Section 50-12 for a list of the available Event Codes. Only those event codes that are labeled for Frame Table execution can be used in a Frame Table.	N/A

FTEQBC Frame Table Event Quarter Bit Count **Addr 2485_3200-2485_337E**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	EQBC[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET																

Table 50-14. FTEQBC Description

Name	Description	Settings
EQBC[15:0] Bits 15-0	**Event Quarter Bit Count—** The EQBC field specifies the event's execution time relative to the start of the channel frame	N/A

50.14.2 Macro Table RAM

Macro tables are used when it is desirable to repeatedly execute an event sequence. A typical use for a macro would be to set up an entire receive burst, transmit burst, and secondary receiver power measurement sequence.

There are 3 macro tables: Macro Table A, Macro Table B, and Macro Table C. Macro Table locations within the 192 entry RAM block are configurable via the Macro Table Configuration registers (**EG_MA_CFG**, **EG_MB_CFG**, **EG_MC_CFG**). Each Macro Table Configuration Register consists of a START pointer (ma_start, mb_start, mc_start) to define the first table entry, and an END pointer (ma_end, mb_end, mc_end) to define the last table entry. A macro table consists of two fields: an 8-bit Event Code (EC) and a 16-bit Relative Event Quarter Bit Count (REQBC). The REQBC field specifies a relative delay from the previous event.

MEC Macro A/B/C Event Code **Addr 2485_3000-2485_317E**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									EC[7:0]							
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET																

Table 50-15. MEC Description

Name	Description	Settings
Bits 15-8	**Reserved**	N/A
EC[7:0] Bits 7-0	**Event Code—** EC contains the Event Code to be executed by the L1 Timer's Event Generator. Refer to Section 50-12 for a list of the available Event Codes. Only those event codes that are labeled for Macro Table execution can be used in a Macro Table.	N/A

MREQBC Macro A/B/C Relative Event Quarter Bit Count **Addr 2485_3200-2485_337E**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	REQBC[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET																

Table 50-16. MREQBC Description

Name	Description	Settings
REQBC[15:0] Bits 15-0	**Relative Event Quarter Bit Count—** The REQBC field specifies the event's execution time relative to the previous event. The REQBC uses zero-offset numbering. An event with an REQBC of zero will execute one quarter bit count after the previous event. (Note: The Event Codes DELAY_TBL and LOOP_TBL consume 2 quarter bit counts.) If the first event in a macro has an REQBC of zero, it will execute one quarter bit count after the frame table starts the macro. Macros are invoked with an associated Parameter Table entry pointer which is used when executing the macro events DELAY_TBL and LOOP_TBL. Refer to Section 50.14.3 for a definition of the Parameter Table RAM.	N/A

50.14.3 Parameter Table RAM

The Parameter Table RAM is used to store a table of delay values and/or loop count values to be used by any of the three macro tables. When a macro is invoked, it specifies a pointer into the Parameter Table RAM. The macro can then sequentially use delay values and loop count values starting with the first value pointed to by the pointer. Every time the macro executes a DELAY_TBL or LOOP_TBL event code, the value is fetched from the Parameter Table using the current pointer location. After the value is fetched, the pointer is incremented to the next value in the Parameter Table. The Parameter Table location within the 192 entry RAM block is configurable via the Parameter Table Configuration registers (**EG_PT_CFG**). The Parameter Table Configuration Register consists of a START pointer (pt_start) to define the first table entry, and an END pointer (pt_end) to define the last table entry. The Parameter Table can be used to store delay values, loop count values, or a mixture thereof.

Macros are invoked by using 1 of 16 different event codes for each macro, for a total of 48 event codes. The 16 different event codes per macro specify a pointer into the Parameter Table RAM, but only at even address offsets. Each macro maintains its own pointer into the Parameter Table RAM. For example, assume Macro Table A is defined as follows:

```
DELAY_TBL
LOOP_TBL
...
DELAY_TBL
...
END_LOOP
```

The Macro Table A can be invoked from the Frame Table using any one of 16 event codes. For this example, it is assumed that the event code to invoke Macro Table A is:

```
MACRO_A_8.
```

During execution of Macro Table A, the first event code DELAY_TBL uses the value stored in entry #8 in the Parameter Table RAM to specify a delay. The Parameter Table RAM pointer is then automatically incremented to point to entry #9 in the Parameter Table RAM. When the event code LOOP_TBL is executed, it uses the value stored in entry #9 in the Parameter Table RAM to determine how many times to repeat through the following event codes up to the END_LOOP event code. The Parameter Table RAM pointer is then automatically incremented again to point to entry #10 in the Parameter Table RAM. The second DELAY_TBL event code (between LOOP_TBL and END_LOOP) uses the value stored in entry #10, and the Parameter Table point is incremented to point to entry #11. For every pass through the loop, the DELAY_TBL fetches the next delay value in the Parameter Table RAM, and the pointer automatically increments.

PTFD Parameter Table Frame Delay **Addr 2485_3000-2485_317E**

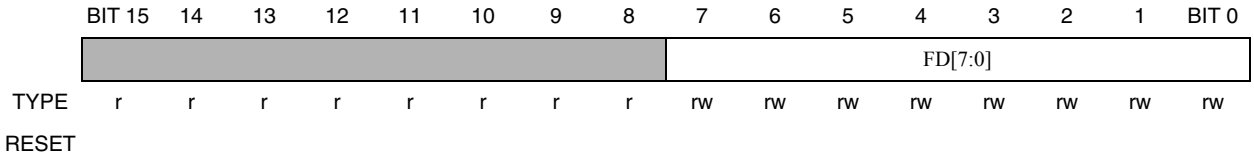


Table 50-17. PTFD Description

Name	Description	Settings
Bits 15-8	**Reserved**	N/A
FD[7:0] Bit 0	**Frame Delay—** FD contains the Frame Delay count which specifies how many frames to delay execution of the DELAY_TBL event code. Note that this Parameter Table RAM is used only for the DELAY_TBL event code and not the LOOP_TBL event code.	N/A

PTDLV Parameter Table Delay/Loop Values **Addr 2485_3200-2485_337E**

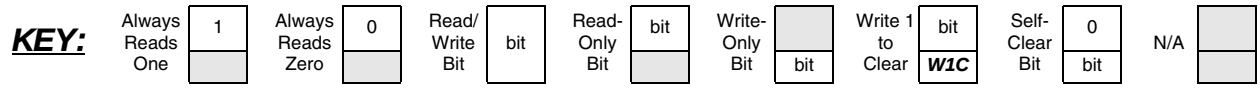
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	REQBCD[15:0]/LC[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET																

Table 50-18. PTDLV Description

Name	Description	Settings
REQBCD[15:0] /LC[15:0] Bits 15-0	**Relative Event Quarter Bit Count Delay—** If a DELAY_TBL event code is executed, the data stored in the Parameter Table Delay/Loop Values will consist of REQBCD data. The REQBCD field specifies the event's delay time. The REQBCD uses zero-offset numbering plus an additional 1 quarter bit count to read the Parameter Table. An REQBCD of zero will cause the DELAY_TBL to consumes 2 quarter bit counts. **Loop Count—**If a LOOP_TBL event code is executed, the data stored in the Parameter Table Delay/Loop Values will consist of LC data. The LC field specifies the event's loop count. An LC of zero will cause the LOOP_TBL event code to terminate the macro execution (similar to the End of Macro, EOM, event code). An LC of 1 will cause the LOOP_TBL event code to execute the subsequent event codes only once, up until the next END_LOOP event code (as if the LOOP_TBL and END_LOOP pair were never needed). An LC of 2 will cause the LOOP_TBL event code to execute the subsequent event codes twice, up until the next END_LOOP event code.	N/A

50.14.4 Register Summary

Figure 0-9.



NOTE:

Writing to any register (besides MSTR) without the MST_EN bit asserted will cause an abort to occur.

Table 50-19. Layer 1 Timer Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSTR $2485_3380	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MST_EN
	W																
GTC0 $2485_3382	R	0	0	0	0	0	0	0	DBEN	DIVEN	DRC	IDTC	IDTB	IDTA	WAKE	SLEEP	SRST
	W																
GTC1 $2485_3384	R	0	0	0	0	0	0	0	0	0	CUP	SM[1:0]		FNE	RFCE	CFCE	TBE
	W																
GTS0 $2485_3386	R	0	0	0	0	0	0	0	0	0	0	0	CMA	BMA	AMA	FTA	EGE
	W																
GTS1 $2485_3388	R	0	0	0	0	0	0	0	0	TERI	PITI	PWTI	DVII	CFCI	RFCI	CFI	RFI
	W									W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
IGM $2485_338A	R	0	0	0	0	0	DWE	FSIE	0	0	PITIE	PWTIE	DVIEE	CFCIE	RFCIE	CFIE	RFIE
	W																
GPWT $2485_338C	R	PWEN	PWT_CNT[14:0]														
	W																
ES0 $2485_338E	R	MCB	MBB	MAB	FTB	MCW	MBW	MAW	BPW	FTW	MCA	MBA	MAA	MCD	MBD	MAD	FTD
	W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
EM 0 $2485_3390	R	MCBE	MBBE	MABE	0	MCWE	MBWE	MAWE	BPWE	FTWE	MCAE	MBAE	MAAE	MCDE	MBDE	MADE	FTDE
	W																
ES1 $2485_3392	R	PDG[15:0]															
	W	W1C															
EM1 $2485_3394	R	PDGE[15:0]															
	W																
ES2 $2485_3396	R	PTB	MCLE	MBLE	MALE	MCPO	MBPO	MAPO	NET	SYNC	MCCE	PME	PREP	PREC	PREB	PREA	PREF
	W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
EM2 $2485_3398	R	PTBE	MCLEE	MBLEE	MALEE	MCPOE	MBPOE	MAPOE	NETE	0	0	0	0	0	0	0	0
	W																
MCUIS $2485_339A	R	MCUI[15:0]															
	W	W1C															
MCUIM $2485_339C	R	MCUIE[15:0]															
	W																
DVIM $2485_339E	R	0	DVIE[14:0]														
	W																
TB_PRE $2485_33A0	R	0	0	0	0	0	0	0	TBPRE[8:0]								
	W																
TBPREM $2485_33A2	R	0	0	0	0	0	0	0	TBPREM[8:0]								
	W																
TRQBC $2485_33A4	R	RQBC[15:0]															
	W																

Table 50-19. Layer 1 Timer Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRQBCM $2485_33A6	R	RQBCM[15:0]															
	W																
TMQBC $2485_33A8	R	MQBC[15:0]															
	W																
TCFC $2485_33AA	R	CFC_SE[5:0]							CFC[9:0]								
	W																
TRFC $2485_33AC	R	RFC_SE[5:0]							RFC[9:0]								
	W																
TRAFN_H $2485_33AE	R	RAFN[31:16]															
	W																
TRAFN_L $2485_33B0	R	RAFN[15:0]															
	W																
TRAFNM_H $2485_33B2	R	RAFNM[31:16]															
	W																
TRAFNM_L $2485_33B4	R	RAFNM[15:0]															
	W																
TCFN_H $2485_33B6	R	CFN[31:16]															
	W																
TCFN_L $2485_33B8	R	CFN[15:0]															
	W																
TQBCFT $2485_33BA	R	0	0	0	0	0	0	0	0	0	0	0	0	0	FTM	FTED	FTEN
	W																
TPIT $2485_33BC	R	PIT[15:0]															
	W																
TPITM $2485_33BE	R	PITM[15:0]															
	W																
QBCO $2485_33C0	R	QBCO[15:0]															
	W																
QBCO_UDT_F $2485_33C2	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	UDF1:0]	
	W																
QBCO_UDT_Q $2485_33C4	R	UDQ[15:0]															
	W																
QBCO_UC0 $2485_33C6	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EMTE	EQUE
	W																
QBCO_UC1 $2485_33C8	R	0	0	0	0	0	0	0	0	SUE[2:0]			MTRIG[4:0]				
	W																
EG_FT0_CFG $2485_33CA	R	FT0_END[7:0]								FT0_START[7:0]							
	W																
EG_FT1_CFG $2485_33CC	R	FT1_END[7:0]								FT1_START[7:0]							
	W																
EG_NFT_CFG $2485_33CE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FT1_NEXT	FT0_NEXT
	W																
EG_MA_CFG $2485_33D0	R	MA_END[7:0]								MA_START[7:0]							
	W																
EG_MB_CFG $2485_33D2	R	MB_END[7:0]								MB_START[7:0]							
	W																
EG_MC_CFG $2485_33D4	R	MC_END[7:0]								MC_START[7:0]							
	W																

Table 50-19. Layer 1 Timer Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EG_PT_CFG $2485_33D6	R	PT_END[7:0]								PT_START[7:0]							
	W																
EG_FPTR $2485_33D8	R	0	0	0	0	0	0	0	0	FPTR[7:0]							
	W																
EG_MPTR0 $2485_33DA	R	BPTR[7:0]								APTR[7:0]							
	W																
EG_MPTR1 $2485_33DC	R	0	0	0	0	0	0	0	0	CPTR[7:0]							
	W																
EG_D0_FD $2485_33DE	R	0	0	0	0	0	0	0	0	DR0FD[7:0]							
	W																
EG_D0_QD $2485_33E0	R	DR0QD[15:0]															
	W																
EG_D1_FD $2485_33E2	R	0	0	0	0	0	0	0	0	DR1FD[7:0]							
	W																
EG_D1_QD $2485_33E4	R	DR1QD[15:0]															
	W																
EG_D2_FD $2485_33E6	R	0	0	0	0	0	0	0	0	DR2FD[7:0]							
	W																
EG_D2_QD $2485_33E8	R	DR2QD[15:0]															
	W																
EG_D3_FD $2485_33EA	R	0	0	0	0	0	0	0	0	DR3FD[7:0]							
	W																
EG_D3_QD $2485_33EC	R	DR3QD[15:0]															
	W																
EG_LP0 $2485_33EE	R	LP0[15:0]															
	W																
EG_LP1 $2485_33F0	R	LP1[15:0]															
	W																
EG_LP2 $2485_33F2	R	LP2[15:0]															
	W																
EG_LP3 $2485_33F4	R	LP3[15:0]															
	W																
PCFG $2485_33F6	R	PCFG[15:0]															
	W																
PD $2485_33F8	R	PD[15:0]															
	W																
PS $2485_33FA	R	PS[15:0]															
	W																
PTC $2485_33FC	R	PTC[15:0]															
	W																
PDIMASK $2485_33FE	R	0	0	0	0	0	0	0	0	0	0	0	0	PD1EDGE	PD0EDGE	PD1INTM	PD0INTM
	W																
EG_MD $2485_3400	R	0	0	0	0	0	0	0	0	0	0	0	0	0	MCD	MBD	MAD
	W																

50.14.5 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the Layer 1 Timer MCU registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

The MSTR register contains the L1 Timer Master Enable bit.

MSTR Master Enable Register **Addr $2485_3380**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
																MST_EN
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-20. MSTR Description

Name	Description	Settings
Bits 15-1	**Reserved**	N/A
mst_en Bit 0	**Master Enable—** This bit gates the master L1 Timer system clock. The MCU has Read only access to the MCU registers (Except MSTR, which is readable and writable) when MST_EN == 0.	0 = L1 Timer clocks off 1 = L1 Timer clocks on

GTC0 contains control bits for software reset, sleep/wakeup, macro termination, and Layer 1 pin control.

GTC0 Global Timer Control 0 Register **Addr $2485_3382**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
								DBEN	DIVEN	DRC	IDTC	IDTB	IDTA	WAKE	SLEEP	SRST
TYPE	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-21. GTC0 Description

Name	Description	Settings
Bits 15-9	**Reserved**	N/A
DBEN Bit 8	**Debug Enable—** Asserting this bit enables the Layer One Timer debug mode. In Debug mode the timer will halt operation when the MCU debug signal is asserted.	0 = Normal Mode 1 = Debug mode enabled
DIVEN Bit 7	**Divider Enable—** Asserting this bit enables a divide by 2 prescaler in the PAT_REF clock path.	0 = Normal Mode 1 = PAT_REF clock is divided by 2
DRC Bit 6	**Default Register Copy—** Asserting this bit will copy the contents of the PCU Default (PD) register into the Event Generator Layer 1 pin drivers. This bit self-clears one system clock cycle after it is asserted. Refer to Table 50-80 for definition of the PD register.	0 = Normal Mode 1 = Copy
IDTC Bit 5	**Immediate Delay Terminate Macro C—** Until the macro terminates, asserting IDTx will force any DELAY, DELAY_REG, and DELAY_TBL event codes to have a delay of 0 Quarter Bit Counts, but will still consume 1 QBC to execute the event code. If the corresponding macro is not active, attempts to write 1 are ignored. This bit is self-clearing upon termination of the macro.	0 = Nominal 1 = Terminate Delays
IDTB Bit 4	**Immediate Delay Terminate Macro B—** Until the macro terminates, asserting IDTx will force any DELAY, DELAY_REG, and DELAY_TBL event codes to have a delay of 0 Quarter Bit Counts, but will still consume 1 QBC to execute the event code. If the corresponding macro is not active, attempts to write 1 are ignored. This bit is self-clearing upon termination of the macro.	0 = Nominal 1 = Terminate Delays
IDTA Bit 3	**Immediate Delay Terminate Macro A—** Until the macro terminates, asserting IDTx will force any DELAY, DELAY_REG, and DELAY_TBL event codes to have a delay of 0 Quarter Bit Counts, but will still consume 1 QBC to execute the event code. If the corresponding macro is not active, attempts to write 1 are ignored. This bit is self-clearing upon termination of the macro.	0 = Nominal 1 = Terminate Delays

Table 50-21. GTC0 Description (Continued)

Name	Description	Settings
WAKE Bit 2	**Event Generator Wake —**To wake up the Event Generator, assert WAKE. The Event Generator will then resync to the channel frame and begin executing from Frame Table 0. Wakeup can be cancelled at any time up to 1 QBC before the start of the next channel frame by clearing this bit. This bit is self-clearing upon successful completion of the wakeup sequence.	0 = Nominal 1 = Event Generator Wakeup scheduled
SLEEP Bit 1	**Event Generator Sleep—** Asserting this bit puts the Event Generator to SLEEP. In the non-immediate sleep modes, SLEEP can be cancelled by clearing this bit before the disabling event (refer to the SM[1:0] bits described above). This bit is self-clearing upon successful entry into the SLEEP state.	0 = Nominal 1 = Initiate SLEEP mode
SRST Bit 0	**Software Reset—**This bit is used in conjunction with the hardware RST pin to put the L1 Timer in RESET mode. Asserting SRST begins a global RESET sequence which affects all blocks within the L1 Timer. This bit self-clears when reset is complete.	0 = Nominal 1 = Initiate Software RESET

GTC1 Global Timer Control 1 Register **Addr $2485_3384**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
										CUP	SM[1:0]		FNE	RFCE	CFCE	TBE
TYPE	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-22. GTC1 Description

Name	Description	Settings
Bits 15-7	**Reserved**	N/A
CUP Bit 6	**Catch Up—** This bit specifies whether or not Frame Table events will execute when MQBC == AEQBC (normal mode) or when MQBC >= AEQBC (catch-up mode).	0 = Nominal 1 = Catch up
SM[1:0] Bits 5-4	**Sleep Mode—** These bits specify when the Event Generator will be disabled when SLEEP is asserted. Note that when SM = 01, the timer will be disabled after the next Frame Table, Macro A, Macro B, or Macro C event is executed.	00 = Immediate 01 = Sleep at the start of the next event 10 = Sleep at the end of this reference frame 11 = Sleep at the end of this channel frame

Table 50-22. GTC1 Description (Continued)

Name	Description	Settings
FNE Bit 3	**Frame Number Enable—** Enabling FNE turns on the Reference Absolute Frame Number counter and the Channel Frame Number latched register. The RAFN is incremented at the start of each reference frame. The CFN latches the RAFN value at the start of each channel frame.	0 = RAFN and CFN disabled 1 = RAFN and CFN enabled
RFCE Bit 2	**Reference Frame Counter Enable—** While RFCE is enabled, the Reference Frame Counter is decremented at the start of the reference frame.	0 = Reference Frame Counter disabled 1 = Reference Frame Counter enabled
CFCE Bit 1	**Channel Frame Counter Enable—**While CFCE is enabled, the Channel Frame Counter is decremented at the start of the channel frame.	0 = Channel Frame Counter disabled 1 = Channel Frame Counter enabled
TBE Bit 0	**Timing Block Enable—** TBE is used to gate the system clock input to the Quarter Bit Block Prescaler. Disabling this bit will cut off the clock to the Timing Block and stop all Timing Block counters from incrementing.	0 = Timing Block disabled 1 = Timing Block enabled

The GTS register contains Event Generator status bits.

GTS0 Global Timer Status 0 Register **Addr $2485_3386**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
												CMA	BMA	AMA	FTA	EGE
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-23. GTS0 Description

Name	Description	Settings
Bits 15-5	**Reserved**	N/A
CMA Bit 4	**Macro C Macro Active—** This read-only bit indicates that the corresponding macro is currently active. This bit is automatically cleared when: • An End of Macro (EOM) event code is executed. • The corresponding macro pointer (CPTR, BPTR, or APTR) reaches the last event in the macro table without executing an EOM event code. In this case, the pointer is set to point to the first event and a macro boundary error is generated. For example, if Macro C executes the last event code in the Macro C Table and it is not an EOM, the CPTR is set to event 0 and a Macro C Boundary Error is generated in the ES0 register.	0 = Macro not active 1 = Macro active

Table 50-23. GTS0 Description

Name	Description	Settings
BMA Bit 3	**Macro B Macro Active—** This read-only bit indicates that the corresponding macro is currently active. This bit is automatically cleared when: • An End of Macro (EOM) event code is executed. • The corresponding macro pointer (CPTR, BPTR, or APTR) reaches the last event in the macro table without executing an EOM event code. In this case, the pointer is set to point to the first event and a macro boundary error is generated. For example, if Macro B executes the last event code in the Macro B Table and it is not an EOM, the BPTR is set to event 0 and a Macro B Boundary Error is generated in the ES0 register.	0 = Macro not active 1 = Macro active
AMA Bit 2	**Macro A Macro Active—** This read-only bit indicates that the corresponding macro is currently active. This bit is automatically cleared when: • An End of Macro (EOM) event code is executed. • The corresponding macro pointer (CPTR, BPTR, or APTR) reaches the last event in the macro table without executing an EOM event code. In this case, the pointer is set to point to the first event and a macro boundary error is generated. For example, if Macro A executes the last event code in the Macro A Table and it is not an EOM, the APTR is set to event 0 and a Macro A Boundary Error is generated in the ES0 register.	0 = Macro not active 1 = Macro active
FTA Bit 1	**Frame Table Active—** This read-only bit indicates the active frame table from the next quarter bit count.	0 = Frame Table 0 active 1 = Frame Table 1 active
EGE Bit 0	**Event Generated Enabled—**This read-only bit indicates if the Event Generator is currently active.	0 = Event Generator disabled 1 = Event Generator enabled

The GTS register contains Event Generator interrupt flags.

GTS1 Global Timer Status 1 Register **Addr $2485_3388**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									TERI	PITI	PWTI	DVII	CFCI	RFCI	CFI	RFI
TYPE	r	r	r	r	r	r	r	r	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-24. GTS1 Description

Name	Description	Settings
Bits 15-8	**Reserved**	N/A
TERI Bit 7	**Timer Error Interrupt—** This bit is asserted if an error condition is detected. Note that the Programmable Wakeup Timer Interrupt (PWTI), Timer Error Interrupt (TERI), the DSP Vectored Interrupt Indicator (DVII) and the Periodic Interrupt Timer Interrupt (PITI) are logically OR'd for driving the MCU_INT0 port of the Layer 1 Timer. The Error Source (ES0, ES1, and ES2) registers and the Error Mask (EM0, EM1, and EM2) registers can be read to find which unmasked interrupt is causing TERI to assert. Writing a 1 to this bit will clear it.	0 = Timer Error Interrupt not asserted 1 = Timer Error Interrupt asserted
PITI Bit 6	**Periodic Interrupt Timer Interrupt—** This bit is asserted if the Periodic Interrupt Timer has rolled over. Note that the Programmable Wakeup Timer Interrupt (PWTI), Timer Error Interrupt (TERI), the DSP Vectored Interrupt Indicator (DVII) and the Periodic Interrupt Timer Interrupt (PITI) are logically OR'd for driving the MCU_INT0 port of the Layer 1 Timer. Writing a 1 to this bit will clear it.	0 = Periodic Interrupt not asserted 1 = Periodic Interrupt asserted
PWTI Bit 5	**Programmable Wakeup Timer Interrupt—** This bit is asserted when the Programmable Wakeup Timer times out. Note that the Programmable Wakeup Timer Interrupt (PWTI), Timer Error Interrupt (TERI), the DSP Vectored Interrupt Indicator (DVII) and the Periodic Interrupt Timer Interrupt (PITI) are logically OR'd for driving the MCU_INT0 port of the Layer 1 Timer. Writing a 1 to this bit will clear it.	0 = Programmable Wakeup Timer Interrupt not asserted 1 = Programmable Wakeup Timer Interrupt asserted
DVII Bit 4	**DSP Vectored Interrupt Indicator—**This bit is asserted when any DSP Vectored Interrupt instruction (DVI15 - DVI0) is executed.	0 = Nominal 1 = DVI instruction executed

Table 50-24. GTS1 Description

Name	Description	Settings
CFCI Bit 3	**Channel Frame Count Interrupt—**This bit is asserted if the Channel Frame Counter times out. Note that Channel Frame Interrupt (CFI) and CFCI are logically OR'd for driving the Channel Interrupt (CI) port of the Layer 1 Timer. Writing a 1 to this bit will clear it.	0 = Channel Frame Count Interrupt not asserted 1 = Channel Frame Count Interrupt asserted
RFCI Bit 2	**Reference Frame Count Interrupt—** This bit is asserted if the Reference Frame Counter times out. This interrupt generates its own interrupt for driving the Reference Frame Count Interrupt (RFCI) port of the Layer 1 Timer. Writing a 1 to this bit will clear it.	0 = Reference Frame Count Interrupt not asserted 1 = Reference Frame Count Interrupt asserted
CFI Bit 1	**Channel Frame Interrupt—** This bit is asserted at the end of every Channel Frame. Note that CFI and Channel Frame Counter Interrupt (CFCI) are logically OR'd for driving the Channel Interrupt (CI) port of the Layer 1 Timer. Writing a 1 to this bit will clear it.	0 = Channel Frame Interrupt not asserted 1 = Channel Frame Interrupt asserted
RFI Bit 0	**Reference Frame Interrupt—** This bit is asserted at the end of every Reference Frame. This interrupt generates its own interrupt for driving the Reference Frame Interrupt (RFI) port of the Layer 1 Timer. Writing a 1 to this bit will clear it.	0 = Reference Frame Interrupt not asserted 1 = Reference Frame Interrupt asserted

The Interrupt Generator Mask is used to select which interrupts will be generated by the Layer 1 Timer. A one enables the interrupt, while a zero masks the interrupt. Note that the TERI interrupt is not maskable. Use the Error Mask registers (EM0, EM1, and EM2) to prevent detection of selected error conditions by the TERI interrupt. Note that all bits in this register map onto the MCU_INT0 interrupt line.

IGM Interrupt Generator Mask Register **Addr $2485_338A**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
						DWE	FSIE			PITIE	PWTIE	DVIEE	CFCIE	RFCIE	CFIE	RFIE
TYPE	r	r	r	r	r	rw	rw	r	r	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-25. IGM Description

Name	Description	Settings
Bits 15-11	**Reserved**	N/A
DWE Bit 10	**DSP Wakeup Interrupt Enable**	0 = No interrupt will be generated 1 = Enable interrupt
FSIE Bit 9	**Frame Sync Interrupt Enable**	0 = No interrupt will be generated 1 = Enable interrupt
Bits 8-7	**Reserved**	N/A
PITIE Bit 6	**Periodic Interrupt Timer Interrupt Enable**	0 = No interrupt will be generated 1 = Enable interrupt
PWTIE Bit 5	**Programmable Wakeup Timer Interrupt Enable**	0 = No interrupt will be generated 1 = Enable interrupt
DVIEE Bit 4	**DSP Vectored Interrupt Indicator Enable**	0 = No interrupt will be generated 1 = Enable interrupt
CFCIE Bit 3	**Channel Frame Counter Interrupt Enable**	0 = No interrupt will be generated 1 = Enable interrupt
RFCIE Bit 2	**Reference Frame Counter Interrupt Enable**	0 = No interrupt will be generated 1 = Enable interrupt
CFIE Bit 1	**Channel Frame Interrupt Enable**	0 = No interrupt will be generated 1 = Enable interrupt
RFIE Bit 0	**Reference Frame Interrupt Enable**	0 = No interrupt will be generated 1 = Enable interrupt

The GPWT register is used to execute an Event Generator wakeup that is independent of the channel frame timing.

GPWT Global Programmable Wakeup Timer **Addr $2485_338C**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PWEN	PWT_CNT[14:0]														
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-26. GPWT Description

Name	Description	Settings
PWEN Bit 15	**Programmable Wakeup Enable—** Setting this bit causes the count value PWT_CNT[14:0] to decrement with every QBC count. When PWT_CNT reaches zero, an Event Generator wakeup is executed, a PWT interrupt is generated, and PWEN is cleared. The Programmable Wakeup can be terminated by clearing the PWEN bit before PWT_CNT reaches zero.	0 = Programmable Wakeup disabled 1 = Programmable Wakeup enabled
PWT_CNT [14:0] Bits 14-0	**Programmable Wakeup Timer Count—** PWT_CNT is a 15-bit time-out value that is decremented at every QBC count when PWEN is set. When PWT_CNT reaches zero, an Event Generator wakeup is executed, a PWT interrupt is generated, and PWEN is cleared.	N/A

Error Source 0 is the first of three registers used to track error conditions. Error conditions in ES0, ES1, and ES2 are logically OR'd to generate a Timer Error Interrupt (TERI). The Error Mask 0 (EM0) register can be used to mask the error conditions stored in this register. Note that the FTB error is non-maskable.

All bits in this register are nominally zero. A one indicates that an error condition has been detected. Bits in this register are cleared by writing a 1. Writing a 0 does not clear the bit.

ESO Error Source 0 Register **Addr $2485_338E**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	MCB	MBB	MAB	FTB	MCW	MBW	MAW	BPW	FTW	MCA	MBA	MAA	MCD	MBD	MAD	FTD
TYPE	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-27. ES0 Description

Name	Description	Settings
MCB Bit 15	**Macro C Boundary Error—**The corresponding table pointer CPTR for Macro C has reached the last entry in the macro without executing an End of Macro (EOM) event code.	0 = Error condition not detected 1 = Error condition detected.
MBB Bit 14	**Macro B Boundary Error—**The corresponding table pointer BPTR for Macro B, has reached the last entry in the macro without executing an End of Macro (EOM) event code.	0 = Error condition not detected 1 = Error condition detected.
MAB Bit 13	**Macro A Boundary Error—**The corresponding table pointer APTR for Macro A has reached the last entry in the macro without executing an End of Macro (EOM) event code.	0 = Error condition not detected 1 = Error condition detected.
FTB Bit 12	**Frame Table Boundary Error—** The Frame Pointer FPTR has reached the last event in the memory (192) without reaching the END pointer of the active frame table. Note that this error will put the timer into SLEEP mode. This error is non-maskable.	0 = Error condition not detected 1 = Error condition detected.
MCW Bit 11	**Macro C Write—** The MCU has written to the corresponding macro table while the macro was active.	0 = Error condition not detected 1 = Error condition detected.
MBW Bit 10	**Macro B Write—** The MCU has written to the corresponding macro table while the macro was active.	0 = Error condition not detected 1 = Error condition detected.
MAW Bit 9	**Macro A Write—** The MCU has written to the corresponding macro table while the macro was active.	0 = Error condition not detected 1 = Error condition detected.
BPW Bit 8	**Boundary Pointer Write—** The MCU has over-written a boundary pointer of the currently active frame table.	0 = Error condition not detected 1 = Error condition detected.

Table 50-27. ES0 Description (Continued)

Name	Description	Settings
FTW Bit 7	**Frame Table Write—**The MCU has over-written events within the boundaries of the active frame table.	0 = Error condition not detected 1 = Error condition detected.
MCA Bit 6	**Macro C Already Active—**The corresponding macro is already active when the same macro is started again.	0 = Error condition not detected 1 = Error condition detected.
MBA Bit 5	**Macro B Already Active—**The corresponding macro is already active when the same macro is started again.	0 = Error condition not detected 1 = Error condition detected.
MAA Bit 4	**Macro A Already Active—**The corresponding macro is already active when the same macro is started again.	0 = Error condition not detected 1 = Error condition detected.
MCD Bit 3	**Macro C Decode—** An illegal event code is executed in the corresponding macro table.	0 = Error condition not detected 1 = Error condition detected.
MBD Bit 2	**Macro B Decode—** An illegal event code is executed in the corresponding macro table.	0 = Error condition not detected 1 = Error condition detected.
MAD Bit 1	**Macro A Decode—** An illegal event code is executed in the corresponding macro table.	0 = Error condition not detected 1 = Error condition detected.
FTD Bit 0	**Frame Table Decode—** An illegal event code is executed in the frame table.	0 = Error condition not detected 1 = Error condition detected.

Error Mask 0 is used to mask detection of the error sources tracked by the Error Source 0 register. Note that the FTB error is non-maskable.

EM0 Error Mask 0 Register **Addr $2485_3390**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	MCBE	MBBE	MABE		MCWE	MBWE	MAWE	BPWE	FTWE	MCAE	MBAE	MAAE	MCDE	MBDE	MADE	FTDE
TYPE	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-28. EM0 Description

Name	Description	Settings
MCBE Bit 15	**Macro C Boundary Enable —** Mask detection of Macro C Boundary Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected.
MBBE Bit 14	**Macro B Boundary Enable —** Mask detection of Macro B Boundary Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected.
MABE Bit 13	**Macro A Boundary Enable —** Mask detection of Macro A Boundary Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected.
Bit 12	**Reserved**	N/A
MCWE Bit 11	**Macro C Write Enable —** Mask detection of Macro C Write Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected.
MBWE Bit 10	**Macro B Write Enable —** Mask detection of Macro B Write Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected.
MAWE Bit 9	**Macro A Write Enable —** Mask detection of Macro A Write Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected.
BPWE Bit 8	**Boundary Pointer Write Enable —** Mask detection of Boundary Pointer Write Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected.
FTWE Bit 7	**Frame Table Write Enable —** Mask detection of Frame Table Write Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected.
MCAE Bit 6	**Macro C Active Enable —** Mask detection of Macro C Active Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected.
MBAE Bit 5	**Macro B Active Enable —** Mask detection of Macro B Active Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected.

Table 50-28. EM0 Description (Continued)

Name	Description	Settings
MAAE Bit 4	**Macro A Active Enable —** Mask detection of Macro A Active Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected.
MCDE Bit 3	**Macro C Decode Enable —** Mask detection of Macro C Decode Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected.
MBDE Bit 2	**Macro B Decode Enable —** Mask detection of Macro B Decode Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected.
MADE Bit 1	**Macro A Decode Enable —** Mask detection of Macro A Decode Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected.
FTDE Bit 0	**Frame Table Decode Enable —** Mask detection of Frame Table Decode Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected.

Error Source 1 is the second of three registers used to track error conditions. Error conditions in ES0, ES1, and ES2 are logically OR'd to generate a Timer Error Interrupt (TERI). The Error Mask 1 (EM1) register can be used to mask the error conditions stored in this register.

ES1 Error Source 1 Register **Addr $2485_3392**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PDG[15:0]															
TYPE	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-29. ES1 Description

Name	Description	Settings
PDG[15:0] Bits 15-0	**Pin Driver Glitch —** A "glitch" error occurs when a Layer 1 Pin Driver is assigned opposing values during the same Quarter Bit. Bits in this register are cleared by writing a 1.	0 = Error condition on PDGn not detected. 1 = Error condition on PDGn detected.

Error Mask 1 is used to mask detection of the error sources tracked by the Error Source 1 register.

EM1 Error Mask 1 Register **Addr $2485_3394**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PDGE[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-30. EM1 Description

Name	Description	Settings
PDGE[15:0] Bits 15-0	**Pin Driver Glitch Enable —** Mask detection of Pin Driver Glitch Error.	0 = No TERI interrupt will be generated for PGDn. 1 = Enable TERI interrupt if an error condition is detected on PGDn.

Error Source 2 is the last of three registers used to track error conditions. Error conditions in ES0, ES1, and ES2 are logically OR'd to generate a Timer Error Interrupt (TERI). The Error Mask 2 (EM2) register can be used to mask the error conditions stored in this register. Note that the errors SYNC, PME, MCCE, PRED, PREC, PREB, PREA, PREF are non-maskable and will put the timer into SLEEP mode.

All bits in this register are nominally zero. A one indicates that an error condition has been detected. Bits in this register are cleared by writing a 1. Writing a 0 does not clear the bit.

ES2 Error Source 2 Register **Addr $2485_3396**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PTB	MCLE	MBLE	MALE	MCPO	MBPO	MAPO	NET	SYNC	MCCE	PME	PREP	PREC	PREB	PREA	PREF
TYPE	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-31. ES2 Description

Name	Description	Settings
PTB Bit 15	**Parameter Table Boundary Error—** This error is asserted if the PT_END pointer is greater than 192 (memory limit).	0 = Error condition not detected 1 = Error condition detected.
MCLE Bit 14	**Macro C Loop Error—** A loop error in Macro C will cause the corresponding bit to be set. Loop errors will occur when a start loop is executed when a loop is already in progress (that is, nested loops are not allowed), and when the END_LOOP event code is executed when no loop is in progress.	0 = Error condition not detected 1 = Error condition detected.
MBLE Bit 13	**Macro B Loop Error—** A loop error in Macro B will cause the corresponding bit to be set. Loop errors will occur when a start loop is executed when a loop is already in progress (that is, nested loops are not allowed), and when the END_LOOP event code is executed when no loop is in progress.	0 = Error condition not detected 1 = Error condition detected.
MALE Bit 12	**Macro A Loop Error—** A loop error in Macro A will cause the corresponding bit to be set. Loop errors will occur when a start loop is executed when a loop is already in progress (that is, nested loops are not allowed), and when the END_LOOP event code is executed when no loop is in progress.	0 = Error condition not detected 1 = Error condition detected.

Table 50-31. ES2 Description (Continued)

Name	Description	Settings
MCPO Bit 11	**Macro C Parameter Pointer Overflow Error—** If the parameter pointer for Macro C is incremented beyond the last entry in the Parameter Table, the corresponding bit will be set. Note that accessing the last entry will cause the error bit to get set since the parameter pointer increments immediately following the fetch. If an error is not desirable, then the corresponding mask bits should be reset in the EM2 register or the last entry of the Parameter Table should not be used.	0 = Error condition not detected 1 = Error condition detected.
MBPO Bit 10	**Macro B Parameter Pointer Overflow Error—** If the parameter pointer for Macro B is incremented beyond the last entry in the Parameter Table, the corresponding bit will be set. Note that accessing the last entry will cause the error bit to get set since the parameter pointer increments immediately following the fetch. If an error is not desirable, then the corresponding mask bits should be reset in the EM2 register or the last entry of the Parameter Table should not be used.	0 = Error condition not detected 1 = Error condition detected.
MAPO Bit 9	**Macro A Parameter Pointer Overflow Error—** If the parameter pointer for Macro A is incremented beyond the last entry in the Parameter Table, the corresponding bit will be set. Note that accessing the last entry will cause the error bit to get set since the parameter pointer increments immediately following the fetch. If an error is not desirable, then the corresponding mask bits should be reset in the EM2 register or the last entry of the Parameter Table should not be used.	0 = Error condition not detected 1 = Error condition detected.
NET Bit 8	**Not End of Table—** The frame table pointer (FPTR) had not reached the active frame table's END pointer when a new channel frame is started (CFI asserted).	0 = Error condition not detected 1 = Error condition detected.
SYNC Bit 7	**Synchronization Error—** The Event Generator did not receive a QBC mark from the Timing Block at the expected time. Note that this error will put the timer into SLEEP mode. This error is non-maskable.	0 = Error condition not detected 1 = Error condition detected.
MCCE Bit 6	**Macro C Call Error —** The prescaler value stored in the Timing Block Prescaler (TB_PRE) register must be 15 or greater to allow operation of the Macro C Table. Note that this error will put the timer into SLEEP mode. This error is non-maskable.	0 = Error condition not detected 1 = Error condition detected.
PME Bit5	**Prescaler Modulus Error—** A prescaler value of 10 or less has been stored in the Timing Block Prescaler (TB_PRE) register. Note that this error will put the timer into SLEEP mode. This error is non-maskable.	0 = Error condition not detected 1 = Error condition detected.

Table 50-31. ES2 Description (Continued)

Name	Description	Settings
PREP Bit 4	**Parameter Table Prefetch Error—** The Event Decoder did not receive expected handshaking signals from the Address Generation Unit when trying to load Parameter Table data. Note that this error will put the timer into SLEEP mode. This error is non-maskable.	0 = Error condition not detected 1 = Error condition detected.
PREC Bit 3	**Macro C Prefetch Error —** The Event Decoder did not receive expected handshaking signals from the Address Generation Unit when trying to load the corresponding Macro Table data. Note that this error will put the timer into SLEEP mode. This error is non-maskable.	0 = Error condition not detected 1 = Error condition detected.
PREB Bit 2	**Macro B Prefetch Error —** The Event Decoder did not receive expected handshaking signals from the Address Generation Unit when trying to load the corresponding Macro Table data. Note that this error will put the timer into SLEEP mode. This error is non-maskable.	0 = Error condition not detected 1 = Error condition detected.
PREA Bit 1	**Macro A Prefetch Error —** The Event Decoder did not receive expected handshaking signals from the Address Generation Unit when trying to load the corresponding Macro Table data. Note that this error will put the timer into SLEEP mode. This error is non-maskable.	0 = Error condition not detected 1 = Error condition detected.
PREF Bit 0	**Frame Table Prefetch Error—** The Event Decoder did not receive expected handshaking signals from the Address Generation Unit when trying to load Frame Table data. Note that this error will put the timer into SLEEP mode. This error is non-maskable.	0 = Error condition not detected 1 = Error condition detected.

Error Mask 2 is used to mask detection of the error sources tracked by the Error Source 2 register.

EM2 Error Mask 2 Register **Addr $2485_3398**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PTBE	MCLEE	MBLEE	MALEE	MCPOE	MBPOE	MAPOE	NETE								
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-32. EM2 Description

Name	Description	Settings
PTBE Bit 15	**Parameter Table Boundary Error Enable —** Mask detection of Parameter Table Boundary Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected
MCLEE Bit 14	**Macro C Loop Error Enable —** Mask detection of Macro C Loop Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected
MBLEE Bit 13	**Macro B Loop Error Enable —** Mask detection of Macro B Loop Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected
MALEE Bit 12	**Macro A Loop Error Enable —** Mask detection of Macro A Loop Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected
MCPOE Bit 11	**Macro C Parameter Pointer Overflow Error Enable —** Mask detection of Macro C Parameter Pointer Overflow Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected
MBPOE Bit 10	**Macro B Parameter Pointer Overflow Error Enable —** Mask detection of Macro B Parameter Pointer Overflow Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected
MAPOE Bit 9	**Macro A Parameter Pointer Overflow Error Enable —** Mask detection of Macro A Parameter Pointer Overflow Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected
NETE Bit 8	**Not End of Table Enable —** Mask detection of Not End of Table Error.	0 = No TERI interrupt will be generated 1 = Enable TERI interrupt if an error condition is detected
Bits 7-0	**Reserved**	N/A

The MCUIS register holds the status of the 16 MCU interrupts.

MCUIS MCU Interrupt Status Register **Addr $2485_339A**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	MCUI[15:0]															
TYPE	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-33. MCUIS Description

Name	Description	Settings
MCUI[15:0] Bits 15-0	**MCU Interrupt —** The MCUIS register is used to track which of the 16 MCU Interrupt instructions have executed.	1 = MCU Interrupt instruction was executed 0 = Nominal

The MCUIM register holds the masks for the corresponding status bits in the MCUIS register. The MCU_INT1 interrupt asserts when any unmasked MCUIS bit is asserted.

MCUIM MCU Interrupt Mask Register **Addr $2485_339C**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	MCUIE[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-34. MCUIM Description

Name	Description	Settings
MCUIE [15:0] Bits 15-0	**MCUIE —** Mask assertion of MCU interrupt.	1 = MCU Interrupt bit unmasked 0 = masked

This register holds the masks for the DSP Vectored Interrupts.

DVIM DSP Interrupt Mask Register **Addr $2485_339E**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
		DVIE[14:0]														
TYPE	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-35. DVIM Description

Name	Description	Settings
Bit 15	**Reserved**	N/A
DVIE [14:0] Bits 14-0	**DVIE —** Mask assertion of DSP interrupt.	1 = DSP Interrupt bit unmasked 0 = masked

This register shows the current value of the prescaler counter. This register is primarily used for test purposes.

TB_PRE Timing Block Prescaler Register **Addr $2485_33A0**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
								TBPRE[8:0]								
TYPE	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1

Table 50-36. TB_PRE Description

Name	Description	Settings
Bits 15-9	**Reserved**	N/A
TBPRE [8:0] Bits 8-0	**Timing Block Prescaler—**The prescaler divides down the L1 Timer system clock into a Quarter Bit Count (QBC) clock. This value represents the current state of the prescaler counter. Writing to this register will change the current prescaler count value, but changing the count modulus value is performed in the Timing Block Prescaler Modulus (TBPREM) register. The prescaler counter counts upward.	N/A

This register allows the prescaler to be programmed from divide-by-12 to divide-by-512. For use in GSM systems using a 13 MHz system clock, the prescaler should be set to divide-by-12 (which is the default value for the register). For use in the Iridium system using a 16.8 MHz system clock, the prescaler should be set to divide-by-24. Note that this register should be written to only when the L1 Timer's Event Generator is disabled.

TBPREM Timing Block Prescaler Modulus Register **Addr $2485_33A2**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
								TBPREM[8:0]								
TYPE	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1

Table 50-37. TBPREM Description

Name	Description	Settings
Bits 15-9	**Reserved**	N/A
TBPREM [8:0] Bits 8-0	**Timing Block Prescaler Modulus—** The prescaler divides down the 16.8 MHz system clock into a Quarter Bit Count (QBC) clock. This value is zero adjusted, that is, a decimal value of TBPREM[8:0] = 23 defines a divide-by-24. The valid range of values is 11 (divide-by-12) through 511 (divide-by-512). A value less than 11 will generate a Prescaler Modulus Error (PME) in the Error Source 2 (ES2) register. A value less than 15 will prevent the Macro C Table from being operational. Attempting to start Macro C when the TBPREM is less than 15 will cause the Macro C Call Error (MCCE) bit in the Error Source 2 (ES2) register to be set, which results in the L1 Timer's Event Generator to go into SLEEP mode.	N/A

This register shows the current value of the Reference Quarter Bit Counter.

TRQBC Timer Reference Quarter Bit Counter Register **Addr $2485_33A4**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	RQBC[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-38. TRQBC Description

Name	Description	Settings
RQBC [15:0] Bits 15-0	**Reference Quarter Bit Count—** This value represents the current state of the Reference Quarter Bit Count counter. Writing to this register will change the current count value, but changing the count modulus value is performed in the Timer Reference Quarter Bit Counter Modulus (TRQBCM) register. The TRQBC counter counts upward.	N/A

This register allows the Timing Reference Quarter Bit Counter to be programmed to the desired modulus. For use in GSM systems using a 13 MHz system clock, the RQBCM should be set to divide-by-5000 (which is the default value for the register). For use in the Iridium system using a 16.8 MHz system clock, the RQBCM should be set to divide-by-63000.

TRQBCM Timer Reference Quarter Bit Counter Modulus Register **Addr $2485_33A6**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	RQBCM[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	1	0	0	1	1	1	0	0	0	0	1	1	1

Table 50-39. TRQBC Description

Name	Description	Settings
RQBCM [15:0] Bits 15-0	**Reference Quarter Bit Count Modulus—** The Reference Quarter Bit Count Modulus divides down the Quarter Bit Count (QBC) clock to define the Reference Frame. This value is zero adjusted, that is, a decimal value of RQBCM[15:0] = 62999 defines a divide-by-63000. The valid range of values is 0 (divide-by-1) through 65535 (divide-by-65536).	N/A

This register allows the MCU to directly read the current channel time, MQBC[15:0].

TMQBC Timer Matching Quarter Bit Counter Register **Addr $2485_33A8**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	MQBC [15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	1	0	0	1	1	1	0	0	0	0	1	1	1

Table 50-40. TMQBC Description

Name	Description	Settings
MQBC [15:0] Bits 15-0	**Reference Quarter Bit Count Modulus—** The Matching Quarter Bit Count (MQBC) indicates position within the channel frame in Quarter-Bit clock (QBC) clock ticks. This value is zero adjusted, that is, a 5000 (decimal) quarter-bit GSM frame results in a maximum value for MQBC[15:0] of 4999. The valid range of values is 0 through 65535.	N/A

The Channel Frame Counter is used to generate a Layer 1 wake-up interrupt after the processor has been asleep. The CFC is enabled by setting the CFCE bit in the Global Timer Control 1 (GTC1) register.

The CFC is a 10-bit signed down-counter. The CFC should be loaded by writing a (positive) 9-bit 2's complement initial value, meaning that CFC[8] must be a 0 to be a positive value. Bits 9-15 are the sign extension of CFC[8:0]. If the CFC is enabled, a trigger signal from the internal rollover detector decrements the CFC. When the counter reaches zero, the CFC will generate a Channel Frame Counter Interrupt (CFCI). After reaching zero, triggers from the rollover detector will continue to decrement the CFC (-1, -2, -3, etc.) until it is reloaded. This allows the interrupt service routine to count the number of frames the interrupt has been pending. This register contains the 9-bit (2's complement) value of the Channel Frame Counter (CFC), sign-extended to 16 bits.

TCFC Timer Channel Frame Counter Register **Addr $2485_33AA**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	CFC_SE[6:0]							CFC[8:0]								
TYPE	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1

Table 50-41. TCFC Description

Name	Description	Settings
CFC_SE [6:0] Bits 15-9	**Channel Frame Count Sign Extension—** This contains the sign extension of CFC[8:0] (CFC_SE[6:0] = CFC[8]). These bits are read-only.	N/A
CFC [8:0] Bits 8-0	**Channel Frame Count —** The Channel Frame Count is set by writing to this register. The current CFC count is read by reading this same register.	N/A

The Reference Frame Counter is used to generate an executive wake-up interrupt after the processor has been in a long sleep period. The RFC is enabled by setting the RFCE bit in the Global Timer Control 1 (GTC1) register.

The RFC is a 10-bit signed down-counter. The RFC should be loaded by writing a (positive) 9-bit 2's complement initial value, meaning that RFC[8] must be a 0 to be a positive number. Bits 9-15 are the sign extension of RFC[8:0]. If the RFC is enabled, a trigger signal from the Reference Quarter Bit Counter decrements the RFC. When the counter reaches zero, the RFC will generate a Reference Frame Counter Interrupt (RFCI). After reaching zero, triggers from the rollover detector will continue to decrement the RFC (-1, -2, -3, etc.) until it is reloaded. This allows the interrupt service routine to count the number of frames the interrupt has been pending. This register contains the 9-bit (2's complement) value of the Reference Frame Counter (RFC), sign-extended to 16 bits.

TRFC Timer Reference Frame Counter Register **Addr $2485_33AC**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	RFC_SE[6:0]							RFC[8:0]								
TYPE	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1

Table 50-42. TRFC Description

Name	Description	Settings
RFC_SE [6:0] Bits 15-9	**Reference Frame Count Sign Extension—** This contains the sign extension of RFC[8:0]. These bits are read-only.	N/A
RFC [8:0] Bits 8-0	**Reference Frame Count—** The Reference Frame Count is set by writing to this register. The current RFC count is read by reading this same register.	N/A

This register, in conjunction with the low-order TRAFN_L register, is a free-running 32-bit, programmable modulus counter. The RAFN is enabled by asserting the Frame Number Enable (FNE) bit in the Global Timer Control 1 (GTC1) register. If FNE is asserted, the RAFN is incremented at the start of the reference frame (RQBC count = 0).

TRAFN_H — Timer Reference Absolute Frame Number High Register — **Addr $2485_33AE**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	RAFN[31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-43. TRAFN_H Description

Name	Description	Settings
RAFN [31:16] Bits 15-0	**Reference Absolute Frame Number— High Order—** This contains the most-significant word of the RAFN. Refer to the TRAFN_L register for the least-significant word. The current count of the most-significant word of the RAFN can be set by writing to this register. The current count is read by reading this same register.	N/A

This register, in conjunction with the high-order TRAFN_H register, is a free-running 32-bit, programmable modulus counter. The RAFN is enabled by asserting the Frame Number Enable (FNE) bit in the Global Timer Control 1 (GTC1) register. If FNE is asserted, the RAFN is incremented at the start of the reference frame (RQBC count = 0).

TRAFN_L — Timer Reference Absolute Frame Number Low Register — **Addr $2485_33B0**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	RAFN[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-44. TRAFN_L Description

Name	Description	Settings
RAFN [15:0] Bits 15-0	**Reference Absolute Frame Number— Low Order —** This contains the least-significant word of the RAFN. Refer to the TRAFN_H register for the most-significant word. The current count of the least-significant word of the RAFN can be set by writing to this register. The current count is read by reading this same register.	N/A

This register, in conjunction with the low-order TRAFNM_L register, defines the free-running, 32-bit, programmable modulus counter. The RAFNM is enabled by asserting the Frame Number Enable (FNE) bit in the Global Timer Control 1 (GTC1) register. If FNE is asserted, the RAFNM is incremented at the start of the reference frame (RQBC count = 0).

TRAFNM_H — Timer Reference Absolute Frame Number Modulus High Register — **Addr $2485_33B2**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	RAFNM[31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1

Table 50-45. TRAFNM_H Description

Name	Description	Settings
RAFNM [31:16] Bits 15-0	**Reference Absolute Frame Number Modulus— High Order** This contains the most-significant word of the RAFNM. Refer to the TRAFNM_L register for the least-significant word. This value is zero adjusted. GSM has a hyperframe of 2715648 frames. For GSM mode, the RAFNM[31:0] would be set to 2715647.	N/A

This register, in conjunction with the high-order TRAFNM_H register, defines the free-running, 32-bit, programmable modulus counter. The RAFNM is enabled by asserting the Frame Number Enable (FNE) bit in the Global Timer Control (GTC) register. If FNE is asserted, the RAFNM is incremented at the start of the reference frame (RQBC count = 0).

TRAFNM_L — Timer Reference Absolute Frame Number Modulus Low Register — **Addr $2485_33B4**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	RAFNM[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1

Table 50-46. TRAFNM_L Description

Name	Description	Settings
RAFNM [15:0] Bits 15-0	**Reference Absolute Frame Number Modulus— Low Order** This contains the most-significant word of the RAFNM. Refer to the TRAFNM_L register for the least-significant word. This value is zero adjusted. GSM has a hyperframe of 2715648 frames. For GSM mode, the RAFNM[31:0] would be set to 2715647.	N/A

This register, in conjunction with the low-order TCFN_L register, is a latched version of the Reference Absolute Frame Number (RAFN). This value is latched at the beginning of every channel frame, that is, when the Channel Frame Interrupt (CFI) is asserted. The purpose of this is to ensure that the Layer 1 software is always reading the correct value. This register is writable for test purposes only.

TCFN_H Timer Channel Frame Number High Register **Addr $2485_33B6**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	CFN[31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-47. TCFN_H Description

Name	Description	Settings
CFN [31:16] Bits 15-0	**Channel Frame Number—High Order —** This contains the most-significant word of the CFN. Refer to the TCFN_L register for the least-significant word.	N/A

This register, in conjunction with the high-order TCFN_H register, is a is a latched version of the Reference Absolute Frame Number (RAFN). This value is latched at the beginning of every channel frame, that is, when the Channel Frame Interrupt (CFI) is asserted. The purpose of this is to ensure that the Layer 1 software is always reading the correct value. This register is writable for test purposes only.

TCFN_L Timer Channel Frame Number Low Register **Addr $2485_33B8**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	CFN[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-48. TCFN_L Description

Name	Description	Settings
CFN [15:0] Bits 15-0	**Reference Absolute Frame Number—Low Order—** This contains the least-significant word of the CFN. Refer to the TCFN_H register for the most-significant word.	N/A

This register is used to synchronize the Layer 1 Timer with an external source. The FRAME_TICK pin can be used to reset the Reference Quarter Bit Count (RQBC) counter. This register allows enabling of the FRAME_TICK to reset the RQBC, select rising or falling edge detection, and select single-shot or continuous reset mode.

TQBCFT Timer Quarter Bit Count Frame Tick Register **Addr $2485_33BA**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
														FTM	FTED	FTEN
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-49. TQBCFT Description

Name	Description	Settings
Bits 15-3	**Reserved**	N/A
FTM Bit 2	**Frame Tick Mode—**This bit selects single-shot or continuous mode of resetting. In continuous mode and when the Frame Tick is enabled using the FTEN, every time the desired edge is detected according to the FTED bit, the RQBC counter is reset. In single-shot mode, when the Frame Tick is enabled using the FTEN and the desired edge is detected according to the FTED bit, the RQBC counter is reset and the FTEN bit is automatically reset to prevent further RQBC resets.	0 = Select continuous mode 1 = Select single-shot mode
FTED Bit 1	**Frame Tick Edge—** This bit selects rising edge or falling edge detection on the FRAME_TICK pin.	0 = Rising edge detection 1 = Falling edge detection
FTEN Bit 0	**Frame Tick Enable—**This bit enables the resetting of the RQBC counter from a detected edge on the FRAME_TICK pin. In single-shot mode (FTM = 1), this bit is automatically cleared after the first edge detection according to the FTED bit.	0 = Frame Tick disabled 1 = Frame Tick reset of the RQBC counter is enabled

The Periodic Interrupt Timer (PIT) is used for circumstances where a continuous stream of interrupts are desired with a known interval, such as may be required by a low-level operating system. This register contains the current value of the PIT counter.

TPIT Timer Periodic Interrupt Timer Register **Addr $2485_33BC**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PIT[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-50. TPIT Description

Name	Description	Settings
PIT [15:0] Bits 15-0	**Periodic Interrupt Time—**This is the instantaneous value of the PIT counter. Writing to this register has the affect of changing the count value, but the Periodic Interrupt Timer Modulus (PITM) register should be used for defining the interrupt periods. The periodic interrupt counter counts upward.	N/A

The Periodic Interrupt Timer (PIT) is used for circumstances where a continuous stream of interrupts are desired with a known interval, such as may be required by a low-level operating system. This register is used to define the modulus value of the PIT counter. The PIT is driven by the Quarter Bit Count (QBC) clock, that is, the output of the Timer Prescaler. The interrupt is enabled using the PITIE bit in the Interrupt Generator Mask (IGM) register.

TPITM Timer Periodic Interrupt Timer Modulus Register **Addr $2485_33BE**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PITM[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-51. TPITM Description

Name	Description	Settings
PITM [15:0] Bits 15-0	**Periodic Interrupt Timer Modulus—** This value defines the number of QBC clocks detected between each generation of the Periodic Interrupt Timer Interrupt (PITI). A value of 0 disables the PIT. This value is zero adjusted, that is, a decimal value of PITM[15:0] = 6999 defines a divide-by-7000. The interval between interrupts can be calculated as follows: Interval = (TBPREM+1) * (PITM+1) / (System Clock Frequency), or, PITM = (Interval * (System Clock Frequency) / (TBPREM+1)) -1, where, • PITM is the PITM[15:0] modulus value, • System Clock Frequency is 16.8 MHz for Iridium mode, and 13.0 MHz for GSM - TBPREM is the TBPREM[5:0] prescaler value in the TBPREM register that is zero offset adjusted. For example, if the desired interval is 10 milliseconds, the system clock frequency is 16.8 MHz, and the prescaler is set to divide-by-24 (TBPREM[5:0] = 23), then PITM[15:0] should be loaded with 6999 decimal, or 1B57 hexadecimal.	N/A

To provide proper timing alignment, the Reference Quarter Bit Count (RQBC) can have an offset applied to it to produce a Matching Quarter Bit Count (MQBC), which is also conditioned with the same modulus value used for the RQBC. The MQBC is the event quarter bit count used in the Frame Tables. The RQBC is a free-running, modulus counter. The QBCO is used to produce an MQBC that can then be aligned in time with an external source (such as a terrestrial base station).

This register is the master for the 16-bit Timer Reference Quarter Bit Count Modulus offset. This register is double buffered. Updating of the slave register depends on the mode selected by the SUE bits in the QBCO Update Control Register 1 (QBCO_UC1) and the EQUE bit in the QBCO Update Control Register 0 (QBCO_UC0).

QBCO Quarter Bit Count Offset Register **Addr $2485_33C0**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QBCO[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-52. QBCO Description

Name	Description	Settings
QBCO [15:0] Bits 15-0	**Quarter Bit Count Offset—** The value is added to the RQBC value, and then is adjusted using the same modulus value as used for the RQBC (TRQBCM[15:0] in the TRQBCM register). A QBCO value of 0 will cause MQBC to be the same as RQBC.	N/A

When QBCO register-update mode is selected, this register specifies the number of frames to wait before applying the update.

QBCO_UDT_F QBCO Update Time - Frame Register **Addr $2485_33C2**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
															UDF[1:0]	
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-53. QBCO_UDT_F Description

Name	Description	Settings
Bits 15-3	**Register**	N/A
UDF [1:0] Bits 1-0	**UDF[1:0]—** When SUE is set to “Synchronous Update at QBCO_UPDATE_TIME” this register stores the number of frames to wait before applying the update. UDF[1:0] is decremented by 1 every channel frame. The Update is applied when UDF[1:0] == 0 and UDQ[15:0] == MQBC[15:0].	N/A

When QBCO register-update mode is selected, this register specifies the channel time at which to apply the update.

QBCO_UDT_Q QBCO Update Time - QBC Register **Addr $2485_33C4**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	UDQ[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-54. QBCO_UDT_Q Description

Name	Description	Settings
UDQ[15:0] Bits 15-0	When SUE is set to “Synchronous update at QBCO_UPDATE_TIME” mode, this register specifies the channel time at which to apply the update. The Update is applied when UDF[1:0] == 0 and UDQ[15:0] == MQBC[15:0].	N/A

This register stores two of the update control bits.

QBCO_UC0 QBCO Update Control 0 Register **Addr $2485_33C6**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
															EMTE	EQUE
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-55. QBCO_UC0 Description

Name	Description	Settings
Bits 15-2	**Reserved**	N/A
EMTE Bit 1	**Event Code MQSPI Trigger Enable—**When this bit is set, it allows an MQSPI trigger to be generated when a QBCO_UPDATE event code is executed, regardless of the EQUE and SUE[1:0] settings. The selected trigger is defined in the MTRIG[4:0] bits. This bit is automatically cleared when a QBCO_UPDATE event code is executed.	0 = An MQSPI Trigger will not be generated when a QBCO_UPDATE event code is executed 1 = An MQSPI Trigger will be generated when a QBCO_UPDATE event code is executed
EQUE Bit 0	**Event Code QBCO Update Enable—**When this bit is set, it allows the QBCO slave register to be updated with the contents of the QBCO register (master) when the QBCO_UPDATE event code is executed. This bit must be set before the new QBCO value is written to the QBCO register, else the QBCO slave register may be updated immediately due to the default asynchronous mode characteristics of the Synchronous Update Enable (SUE[1:0]) bits in the GTC register. This bit is automatically cleared when a QBCO_UPDATE event code is executed. While this bit is set, the SUE[1:0] bits are not functional.	0 = The QBCO slave register will not be updated with the contents in the QBCO register 1 = The QBCO slave register will be updated with the contents in the QBCO register when the next QBCO_UPDATE event code is executed.

This register stores two of the update control bits.

QBCO_UC1 QBCO Update Control 1 Register **Addr $2485_33C8**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									SUE[2:0]			MTRIG[4:0]				
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-56. QBCO_UC1 Description

Name	Description	Settings
Bits 15-8	**Reserved**	N/A
SUE[2:0] Bits 7-5	**Synchronous Update Enable Mode**	000 - Asynchronous Update 001 - Increment QBCO by 1 on next channel frame interrupt 010 - Decrement QBCO by 1 on next channel frame interrupt 011 - Synchronous update at next channel frame interrupt 1XX - Update at UPDATE_TIME (UDF == 0, UDQ == MQBC)
MTRIG[4:0] Bits 4-0	**MQSPI Trigger Number—**This value specifies 1 of the 32 triggers to trigger an MQSPI transfer. An MQSPI trigger is generated if the EMTE bit is set and a QBCO_UPDATE event code is executed.	00000 = Trigger 0 … 11111 = Trigger 31

This register contains the START and END boundary pointers for Frame Table 0. Note that upon RESET, FT0_START points to Event Table entry 0 and FT0_END points to Event Table event 31. **The first event to be executed following wakeup will be Frame Table 0 entry 0**.

EG_FT0_CFG Frame Table 0 Configuration Register **Addr $2485_33CA**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	FT0_END[7:0]								FT0_START[7:0]							
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0

Table 50-57. EG_FT0_CFG Description

Name	Description	Settings
FT0_END[7:0] Bits 15-8	**Frame Table 0 End Pointer—**This contains the address offset for the last entry for Frame Table 0 in the Frame Table RAM. The internal frame pointer (FPTR) is incremented as each event code entry in the frame table is executed. The FT0_END[7:0] should be equal to or greater than the FT0_START[7:0] pointer.	N/A
FT0_START[7:0] Bit 7-0	**Frame Table 0 Start Pointer—**This contains the address offset for the first entry for Frame Table 0 in the Frame Table RAM. The internal frame pointer (FPTR) is incremented as each event code entry in the frame table is executed. The FT0_START[7:0] should be equal to or less than the FT0_END[7:0] pointer.EG_FT0_CFG – Frame Table 0 Configuration Register	N/A

This register contains the START and END boundary pointers for Frame Table 1. Note that upon RESET, FT1_START points to Event Table entry 32 and FT1_END points to Event Table event 63.

EG_FT1_CFG Frame Table 1 Configuration Register **Addr $2485_33CC**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	FT1_END[7:0]								FT1_START[7:0]							
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	1	1	1	1	1	1	0	0	1	0	0	0	0	0

Table 50-58. EG_FT1_CFG Description

Name	Description	Settings
FT1_END[7:0] Bits 15-8	**Frame Table 1 End Pointer—**This contains the address offset for the last entry for Frame Table 1 in the Frame Table RAM. The internal frame pointer (FPTR) is incremented as each event code entry in the frame table is executed. The FT1_END[7:0] should be equal to or greater than the FT1_START[7:0] pointer.	N/A
FT1_START[7:0] Bit 7-0	**Frame Table 1 Start Pointer—**This contains the address offset for the first entry for Frame Table 1 in the Event Table RAM. The internal frame pointer (FPTR) is incremented as each event code entry in the frame table is executed. The FT1_START[7:0] should be equal to or less than the FT1_END[7:0] pointer.EG_FT1_CFG – Frame Table 1 Configuration Register	N/A

This register contains the FT_NEXT bits for Frame Table 0 and Frame Table 1. Note that upon RESET, FT0_START points to Event Table entry 0 and FT0_END points to Event Table event 31. **The first event to be executed following wakeup will be Frame Table 0 entry 0**.

EG_NFT_CFG Next Frame Table Configuration Register **Addr $2485_33CE**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
															FT1_NEXT	FT0_NEXT
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-59. EG_NFT_CFG Description

Name	Description	Settings
Bits 15-2	**Reserved**	N/A
FT1_NEXT Bit 1	**Next Frame Table—** If Frame Table 1 is executing, this bit selects whether the next frame table to be executed is Frame Table 0 or Frame Table 1.	0 = Next table is Frame Table 0 1 = Next table is Frame Table 1
FT0_NEXT Bit 0	**Next Frame Table—** If Frame Table 0 is executing, this bit selects whether the next frame table to be executed is Frame Table 0 or Frame Table 1. Since FT0_NEXT = 0 at RESET, by default the timer will repeat Frame Table 0 until FT0_NEXT is set to 1.	0 = Next table is Frame Table 0 1 = Next table is Frame Table 1

This register contains the START and END boundary pointers for Macro A. Note that upon RESET, MA_START points to Event Table entry 64 and MA_END points to Event Table event 95.

EG_MA_CFG Macro A Table Configuration Register **Addr $2485_33D0**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	MA_END[7:0]								MA_START[7:0]							
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	1	0	1	1	1	1	1	0	1	0	0	0	0	0	0

Table 50-60. EG_MA_CFG Description

Name	Description	Settings
MA_END[7:0] Bits 15-8	**Macro A Table End Pointer**— This contains the address offset for the last entry for Macro A in Event Table RAM. The internal frame pointer (APTR) is incremented as each event code entry in the frame table is executed. The MA_END[7:0] should be equal to or greater than the MA_START[7:0] pointer.	N/A
MA_START[7:0] Bits 7-0	**Macro A Table Start Pointer—**This contains the address offset for the first entry for Macro A in Event Table RAM. The internal frame pointer (APTR) is incremented as each event code entry in the frame table is executed. The MA_START[7:0] should be equal to or less than the MA_END[7:0] pointer.	N/A

This register contains the START and END boundary pointers for Macro B. Note that upon RESET, MB_START points to Event Table entry 96 and MB_END points to Event Table event 127.

EG_MB_CFG Macro B Table Configuration Register **Addr $2485_33D2**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	MB_END[7:0]								MB_START[7:0]							
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	1	1	1	1	1	1	1	0	1	1	0	0	0	0	0

Table 50-61. EG_MB_CFG Description

Name	Description	Settings
MB_END[7:0] Bits 15-8	**Macro B Table End Pointer**—This contains the address offset for the last entry for Macro B in Event Table RAM. The internal frame pointer (BPTR) is incremented as each event code entry in the frame table is executed. The MB_END[7:0] should be equal to or greater than the MB_START[7:0] pointer.	N/A
MB_START[7:0] Bits 7-0	**Macro B Table Start Pointer—**This contains the address offset for the first entry for Macro B in Event Table RAM. The internal frame pointer (BPTR) is incremented as each event code entry in the frame table is executed. The MB_START[7:0] should be equal to or less than the MB_END[7:0] pointer.	N/A

This register contains the START and END boundary pointers for Macro C. Note that upon RESET, MC_START points to Event Table entry 128 and FT0_END points to Event Table event 159.

EG_MC_CFG Macro C Table Configuration Register **Addr $2485_33D4**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	MC_END[7:0]								MC_START[7:0]							
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	1	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0

Table 50-62. EG_MC_CFG Description

Name	Description	Settings
MC_END[7:0] Bits 15-8	**Macro C Table End Pointer**—This contains the address offset for the last entry for Macro C in Event Table RAM. The internal frame pointer (CPTR) is incremented as each event code entry in the frame table is executed. The MC_END[7:0] should be equal to or greater than the MC_START[7:0] pointer.	N/A
MC_START[7:0] Bits 7-0	**Macro C Table Start Pointer—**This contains the address offset for the first entry for Macro C in Event Table RAM. The internal frame pointer (CPTR) is incremented as each event code entry in the frame table is executed. The MC_START[7:0] should be equal to or less than the MC_END[7:0] pointer.	N/A

This register contains the START and END boundary pointers for the Parameter Table. Note that upon RESET, PT_START points to Event Table entry 160 and PT_END points to Event Table event 191.

EG_PT_CFG Parameter Table Configuration Register **Addr $2485_33D6**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PT_END[7:0]								PT_START[7:0]							
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	1	0	1	1	1	1	1	1	1	0	1	0	0	0	0	0

Table 50-63. EG_PT_CFG Description

Name	Description	Settings
PT_END[7:0] Bits 15-8	**Parameter Table End Pointer**—This contains the address offset for the last entry for the Parameter Table in Event Table RAM. PT_END[7:0] should be equal to or greater than the PT_START[7:0] pointer.	N/A
PT_START[7:0] Bits 7-0	**Parameter Table Start Pointer—**This contains the address offset for the first entry for the Parameter Table in Event Table RAM. PT_START[7:0] should be equal to or less than the PT_END[7:0] pointer.	N/A

This register shows the Frame Table pointer used by the Event Generator.

EG_FPTR Frame Table Pointer Register **Addr $2485_33D8**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									FPTR[7:0]							
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?

Table 50-64. EG_FPTR Description

Name	Description	Settings
Bits 15-8	**Reserved**	N/A
FPTR[7:0] Bits 7-0	**Frame Table Pointer**—This pointer specifies which entry in the Frame Table is next to be executed. The pointer represents an offset from the first entry into the Frame Table RAM. Note that FPTR is unknown after RESET. These bits will have a valid value when: • - RESET is asserted a second time, or • - An Event Generator WAKEUP is successfully executed. Explanation: the Frame Table pointer (FPTR) is initialized at Event Generator wakeup by the value stored in the FT0_START bit of the EG_FT0_CFG register. When the first RESET is asserted, the FT0_START bits are given an initial value, and FPTR latches an invalid value. If a second reset is asserted (or a wakeup executed), FPTR will latch a valid FT0_START value.	N/A

This register shows the Macro Table pointers used by the Event Generator for Macro B and Macro A.

EG_MPTR0 Macro Table Pointer Register 0 **Addr $2485_33DA**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	BPTR[7:0]								APTR[7:0]							
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 50-65. EG_MPTR0 Description

Name	Description	Settings
BPTR[7:0] Bits 15-8	**Macro B Table Pointer**—This pointer specifies which entry in the Macro B Table is next to be executed. The pointer represents an offset from the first entry into the Macro B Table RAM.	N/A
APTR[7:0] Bits 7-0	**Macro A Table Pointer**—This pointer specifies which entry in the Macro A Table is next to be executed. The pointer represents an offset from the first entry into the Macro A Table RAM.	N/A

Note: APTR and BPTR are unknown after RESET. These bits will have a valid value when:

- - RESET is asserted a second time, or
- - An Event Generator WAKEUP is successfully executed.

Explanation: same as for FPTR

This register shows the Macro Table pointer used by the Event Generator for Macro C.

EG_MPTR1 Macro Table Pointer Register 1 **Addr $2485_33DC**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									CPTR[7:0]							
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?

Table 50-66. EG_MPTR1 Description

Name	Description	Settings
Bits 15-8	**Reserved**	N/A
CPTR[7:0] Bits 7-0	**Macro C Table Pointer**—This pointer specifies which entry in the Macro C Table is next to be executed. The pointer represents an offset from the first entry into the Macro C Table RAM.	N/A

Note: CPTR is unknown after RESET. These bits will have a valid value when:

- - RESET is asserted a second time, or
- - An Event Generator WAKEUP is successfully executed.

Explanation: same as for FPTR

This register contains the Frame Delay value for Delay Register 0. The value in this register is loaded by the Event Generator when DELAY_REG0 event code is executed. This register is used in conjunction with the Delay Register 0 QBC Delay register (EG_D0_QD).

EG_D0_FD Delay Register 0 Frame Delay **Addr $2485_33DE**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									DROFD[7:0]							
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-67. EG_D0_FD Description

Name	Description	Settings
Bits 15-8	**Reserved**	N/A
DROFD[7:0] Bits 7-0	**Delay Register 0 Frame Delay**—This value specifies how many frames to delay when DELAY_REG0 event code is executed.	N/A

This register contains the Quarter Bit Count (QBC) Delay value for Delay Register 0. The value in this register is loaded by the Event Generator when DELAY_REG0 event code is executed. This register is used in conjunction with the Delay Register 0 Frame Delay register (EG_D0_FD).

EG_D0_QD Delay Register 0 Quarter Bit Count Delay **Addr $2485_33E0**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	DR0QD[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-68. EG_D0_QD Description

Name	Description	Settings
DR0QD[15:0] Bits 15-0	**Delay Register 0 QBC Delay—**This value specifies how many Quarter Bit Counts to delay when DELAY_REG0 event code is executed.	N/A

This register contains the Frame Delay value for Delay Register 1. The value in this register is loaded by the Event Generator when DELAY_REG1 event code is executed. This register is used in conjunction with the Delay Register 1 QBC Delay register (EG_D1_QD).

EG_D1_FD Delay Register 1 Frame Delay **Addr $2485_33E2**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									DR1FD[7:0]							
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-69. EG_D1_FD Description

Name	Description	Settings
Bits 15-8	Reserved	N/A
DR1FD[7:0] Bits 7-0	**Delay Register 1 Frame Delay**—This value specifies how many frames to delay when DELAY_REG1 event code is executed.	N/A

This register contains the Quarter Bit Count (QBC) Delay value for Delay Register 1. The value in this register is loaded by the Event Generator when DELAY_REG1 event code is executed. This register is used in conjunction with the Delay Register 1 Frame Delay register (EG_D1_FD).

EG_D1_QD Delay Register 1 Quarter Bit Count Delay **Addr $2485_33E4**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	DR1QD[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-70. EG_D1_QD Description

Name	Description	Settings
DR1QD[15:0] Bits 15-0	**Delay Register 1 QBC Delay**—This value specifies how many QBCs to delay when DELAY_REG1 event code is executed.	N/A

This register contains the Frame Delay value for Delay Register 2. The value in this register is loaded by the Event Generator when DELAY_REG2 event code is executed. This register is used in conjunction with the Delay Register 2 QBC Delay register (EG_D2_QD).

EG_D2_FD Delay Register 2 Frame Delay **Addr $2485_33E6**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									DR2FD[7:0]							
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-71. EG_D2_FD Description

Name	Description	Settings
Bits 15-8	**Reserved**	N/A
DR2FD[7:0] Bits 7-0	**Delay Register 2 Frame Delay**—This value specifies how many frames to delay when DELAY_REG2 event code is executed.	N/A

This register contains the Quarter Bit Count (QBC) Delay value for Delay Register 2. The value in this register is loaded by the Event Generator when DELAY_REG2 event code is executed. This register is used in conjunction with the Delay Register 2 Frame Delay register (EG_D2_FD).

EG_D2_QD Delay Register 2 Quarter Bit Count Delay **Addr $2485_33E8**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	DR2QD[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-72. EG_D2_QD Description

Name	Description	Settings
DR2QD[15:0] Bits 15-0	**Delay Register 2 QBC Delay**—This value specifies how many QBCs to delay when DELAY_REG2 event code is executed.	N/A

This register contains the Frame Delay value for Delay Register 3. The value in this register is loaded by the Event Generator when DELAY_REG3 event code is executed. This register is used in conjunction with the Delay Register 3 QBC Delay register (EG_D3_QD).

EG_D3_FD Delay Register 3 Frame Delay **Addr $2485_33EA**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									DR3FD[7:0]							
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-73. EG_D3_FD Description

Name	Description	Settings
Bits 15-8	**Reserved**	N/A
DR3FD[7:0] Bits 7-0	**Delay Register 3 Frame Delay—**This value specifies how many frames to delay when DELAY_REG3 event code is executed.	N/A

This register contains the Quarter Bit Count (QBC) Delay value for Delay Register 3. The value in this register is loaded by the Event Generator when DELAY_REG3 event code is executed. This register is used in conjunction with the Delay Register 3 Frame Delay register (EG_D3_FD).

EG_D3_QD Delay Register 3 Quarter Bit Count Delay **Addr $2485_33EC**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	DR3QD[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-74. EG_D3_QD Description

Name	Description	Settings
DR3QD[15:0] Bits 15-0	**Delay Register 3 QBC Delay—**This value specifies how many QBCs to delay when DELAY_REG3 event code is executed.	N/A

This register contains the loop count value that provides a means of repeating a set of event codes in a macro. The value in this register is loaded by the Event Generator when LOOP_REG0 event code is executed.

EG_LP0 Loop Count 0 Register **Addr $2485_33EE**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	LP0[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-75. EG_LP0 Description

Name	Description	Settings
LP0[15:0] Bits 15-0	**Loop Count 0**—This value specifies how many times the event codes will be executed when a LOOP_REG0 event code is executed. The event codes that are repeated are placed between the LOOP_REG0 event code and an END_LOOP event code.	N/A

This register contains the loop count value that provides a means of repeating a set of event codes in a macro. The value in this register is loaded by the Event Generator when LOOP_REG1 event code is executed.

EG_LP1 Loop Count 1 Register **Addr $2485_33F0**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	LP1[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-76. EG_LP1 Description

Name	Description	Settings
LP1[15:0] Bits 15-0	**Loop Count 1**—This value specifies how many times the event codes will be executed when a LOOP_REG1 event code is executed. The event codes that are repeated are placed between the LOOP_REG1 event code and an END_LOOP event code.	N/A

This register contains the loop count value that provides a means of repeating a set of event codes in a macro. The value in this register is loaded by the Event Generator when LOOP_REG2 event code is executed.

EG_LP2 Loop Count 2 Register **Addr $2485_33F2**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	LP2[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-77. EG_LP2 Description

Name	Description	Settings
LP2[15:0] Bits 15-0	**Loop Count 2**—This value specifies how many times the event codes will be executed when a LOOP_REG2 event code is executed. The event codes that are repeated are placed between the LOOP_REG2 event code and an END_LOOP event code.	N/A

This register contains the loop count value that provides a means of repeating a set of event codes in a macro. The value in this register is loaded by the Event Generator when LOOP_REG3 event code is executed.

EG_LP3 Loop Count 3 Register **Addr $2485_33F4**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	LP3[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-78. EG_LP3 Description

Name	Description	Settings
LP3[15:0] Bits 15-0	**Loop Count 3**—This value specifies how many times the event codes will be executed when a LOOP_REG3 event code is executed. The event codes that are repeated are placed between the LOOP_REG3 event code and an END_LOOP event code.	N/A

This register (along with the DTX_SELECT input bus from the DSP, located in the GPIO module) configures the Pin Control Unit (PCU) multiplexer (MUX), allowing the MCU to override the Event Generator's pin driver signals with the values stored in the PCU Default (PD) register.

PCFG Pin Control Unit Configuration Register **Addr $2485_33F6**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PCFG[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-79. PCFG Description

Name	Description	Settings
PCFG[15:0] Bits 15-0	**PCU Configuration**—Each bit represents one of the pin driver output pins on the Neptune IC. PCFG[0] represents the TOUT0 pin on the Neptune IC.	0 = The corresponding pin driver signal is defined by the corresponding bit in the PCU Default register 1 = The corresponding pin driver signal is controlled by the Event Generator

This register defines the default values for the Event Generator's pin driver output signals. Since the PDx_INTERNAL signals are always active high, if the default for TOUT0 or TOUT1 is specified as 1, the PDx_INTERNAL signal will be inverted with respect to TOUTx.

PD Pin Control Unit Default Register **Addr $2485_33F8**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PD[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-80. PD Description

Name	Description	Settings
PD[15:0] Bits 15-0	**PCU Default**—Each bit represents one of the pin driver output pins on the Neptune IC. PD[0] represents the TOUT0 pin on the Neptune IC.	0 = The corresponding pin driver signal output is defaulted to a logic 0 1 = The corresponding pin driver signal output is defaulted to a logic 1

This read-only register contains the value of the pin driver signals out of the multiplexer (MUX), where the MUX consists directs the pin driver signals from either the Event Generator or from the PCU Default register, according to the PCU Configuration (PCFG) register. This register reflects the actual state of the pin driver outputs with the exception of any pins that are currently tri-stated, according to the PCU Tri-state Configuration (PTC) register.

PS Pin Control Unit Status Register **Addr $2485_33FA**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PS[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-81. PS Description

Name	Description	Settings
PS[15:0] Bits 15-0	**PCU Status**— Each bit represents one of the pin driver output pins on the Neptune IC. PS[0] represents the TOUT0 pin on the Neptune IC. These bits are read-only.	0 = The corresponding pin driver signal output is a logic 0 1 = The corresponding pin driver signal output is a logic 1

This register configures the tri-state pin driver outputs of the Pin Control Unit. After RESET, all of the pin driver outputs are defaulted to tri-state mode. External pull-up and pull-down resistors must be applied to the output pins to define the default values during and immediately after a RESET.

PTC Pin Control Unit Tri-state Configuration Register **Addr $2485_33FC**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PTC[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-82. PTC Description

Name	Description	Settings
PTC[15:0] Bits 15-0	**PCU Tri-state Configuration**— Each bit represents one of the pin driver output pins on the Neptune IC. PTC[0] represents the TOUT0 pin on the Neptune IC.	0 = The corresponding pin driver signal output is in a high impedance state 1 = The corresponding pin driver signal output is configured as an output driver

This register holds the masks and transition edges for the PD0_INTERNAL and PD1_INTERNAL signal. If these bits are set to 1, the INTERNAL signals will track the assert/negate status of PD0 or PD1, respectively. If these bits are 0, the INTERNAL signals will be negated.

PDIMASK PD_Internal Mask Register **Addr $2485_33FE**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
													PD1 EDGE	PD0 EDGE	PD1 MASK	PD0 MASK
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-83. PDIMASK Description

Name	Description	Settings
Bits 15-4	**Reserved**	N/A
PD1 EDGE Bit 3	**PD1 Interrupt Edge —** Selects the PD1 edge that will generate the PD1 interrupt	1 = Rising edge of PD1 generates PD1 interrupt 0 = Falling edge of PD1 generates PD1 interrupt
PD0 EDGE Bit 2	**PD0 Interrupt Edge —** Selects the PD0 edge that will generate the PD0 interrupt.	1 = Rising edge of PD0 generates PD0 interrupt 0 = Falling edge of PD0 generates PD0 interrupt
PD1 MASK Bit 1	**PD1 Interrupt Mask —** Masks assertion of DSP Layer 1 transmit interrupt generated on edge transition of PD1_INTERNAL	1 = DSP Layer 1 transmit interrupt unmasked 0 = masked
PD0 MASK Bit 0	**PD0 Interrupt Mask —** Masks assertion of DSP Layer 1 receive interrupt generated on edge transition of PD0_INTERNAL	1 = DSP Layer 1 receive interrupt unmasked 0 = masked

This register allows the disabling of Macro C, Macro B, and Macro A Tables.

EG_MD — Macro Table Disable Register — Addr $2485_3400

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
														MCD	MBD	MAD
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-84. EG_MD Description

Name	Description	Settings
Bits 15-3	Reserved	N/A
MCD Bit 2	**Macro C Disable—** This bit disables Macro C Table from being executed. If an event code in the Frame Table attempts to start the macro when it has been disabled, the event code is ignored. For Macro C Table only, if the Timing Block prescaler is set to less than 15 (divide-by-16) in the Timing Block Prescaler (TB_PRE) register, then Macro C is disabled. However, there is no indication from the MCD bit that it is disabled.	0 = Macro C is enabled 1 = Macro C is disabled
MBD Bit 1	**Macro B Disable—**This bit disables Macro B Table from being executed. If an event code in the Frame Table attempts to start the macro when it has been disabled, the event code is ignored.	0 = Macro B is enabled 1 = Macro B is disabled
MAD Bit 0	**Macro A Disable—** This bit disables Macro A Table from being executed. If an event code in the Frame Table attempts to start the macro when it has been disabled, the event code is ignored.	0 = Macro A is enabled 1 = Macro A is disabled

50.15 DSP Memory Map

The base address for the Layer 1 Timer DSP registers is Y:$FF80.

All addresses are specified as offsets from the base address.

Change[1]	Address	Access	Register	Description
P	Y:$FF87	R	RQBC	Reference Quarter Bit Count
P	Y:$FF88	R	MQBC	Matching Quarter Bit Count
P	Y:$FF89	R	QBCO	QBCO - MCU
P	Y:$FF8A	R/W	DTX	Discontinuous Transmit Override

1. P - New register for Patriot.

Figure 0-10.

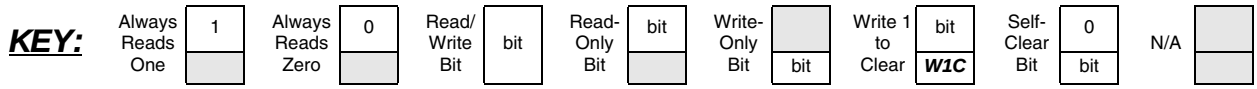


Table 50-85. Layer 1 Timer DSP Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RQBC Y:$FF87	R	RQBC[15:0]															
	W																
MQBC Y:$FF88	R	MQBC[15:0]															
	W																
QBCO Y:$FF89	R	QBCO_MCU[15:0]															
	W																
DTX Y:$FF8A	R	DTX[15:0]															
	W																
DSP_EG_MD Y:$FF8B	R	0	0	0	0	0	0	0	0	0	0	0	0	0	MCD	MBD	MAD
	W																

50.15.1 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various Layer 1 Timer DSP registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - — **r**: read only. Writing this bit has no effect.
  - — **w**: write only.
  - — **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).

  - — **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - — **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - — **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - — **0**: Will reset to a logic 0
  - — **1**: Will reset to a logic 1
  - — **?**: The reset state is unknown.
  - — **u**: Unaffected by reset

This register shows the current value of the Reference Quarter Bit Counter. Since this register shadows the MCU side register, and must be synchronized between the MCU and DSP clock domains, it will be updated only every qbc clock period (12 MCU clocks). This register also lags the MCU-side equivalent register by as many as 3 DSP clock cycles.

RQBC Reference Quarter Bit Counter Register **Addr Y:$FF87**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	RQBC[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-86. RQBC Description

Name	Description	Settings
RQBC[15:0] Bits 15-0	**Reference Quarter Bit Count—** This read only register holds the Reference Quarter Bit Count. Note that this register is only updated when the qbc_clock is detected by the DSP system. Therefore, the Timing Block must be enabled prior to fetching the RQBC from this register.	N/A

This register allows the DSP to directly read the current channel time, MQBC[15:0]. Since this register shadows the MCU side register, and must be synchronized between the MCU and DSP clock domains it will be updated only every qbc clock period (12 MCU clocks). This register also lags the MCU-side equivalent register by as many as 3 DSP clock cycles.

MQBC Timer Matching Quarter Bit Counter Register **Addr Y:$FF88**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	MQBC[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-87. MQBC Description

Name	Description	Settings
MQBC[15:0] Bits 15-0	**Matching Quarter Bit Count—** This read only register holds the Matching Quarter Bit Count. This is a shadow register of the MCU TMQBS register. Note that this register is only updated when the qbc_clock is detected by the DSP system. Therefore, the Timing Block must be enabled prior to fetching the MQBC from this register.	N/A

This register is R-only copy of the MCU-writable QBCO offset. Since this register shadows the MCU side register, and must be synchronized between the MCU and DSP clock domains, it will be updated only every qbc clock period. plus by as many as 3 DSP clock cycles.

QBCO MCU Quarter Bit Counter Offset Register **Addr Y:$FF89**

	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	QBCO_MCU[15:0]														
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-88. QBCO Description

Name	Description	Settings
QBCO[15:0] Bits 15-0	**Quarter Bit Count Offset—** This read only register holds the Matching Quarter Bit Count Offset. Note that this register is only updated when the qbc_clock is detected by the DSP system. Therefore, the Timing Block must be enabled prior to fetching the MQBC from this register.	N/A

DTX Discontinuous Transmit Register **Addr Y:$FF8A**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	DTX[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-89. DTX Description

Name	Description	Settings
DTX Bits 15-0	**Discontinuous Transmit Register—** This register allows the DSP to override the event generator outputs.	0 = Nominal 1 = Force PD[X] output to PD register value, overriding event generator value

DSP_EG_MD DSP Macro Table Disable Register **Addr Y:$FF8B**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
														MCD	MBD	MAD
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50-90. DSP_EG_MD Description

Name	Description	Settings
Bits 15-3	Reserved	N/A
MCD Bit 2	**Macro C Disable—** This bit disables Macro C Table from being executed. If an event code in the Frame Table attempts to start the macro when it has been disabled, the event code is ignored. For Macro C Table only, if the Timing Block prescaler is set to less than 15 (divide-by-16) in the Timing Block Prescaler (TB_PRE) register, then Macro C is disabled. However, there is no indication from the MCD bit that it is disabled.	0 = Macro C is enabled 1 = Macro C is disabled
MBD Bit 1	**Macro B Disable—**This bit disables Macro B Table from being executed. If an event code in the Frame Table attempts to start the macro when it has been disabled, the event code is ignored.	0 = Macro B is enabled 1 = Macro B is disabled
MAD Bit 0	**Macro A Disable—** This bit disables Macro A Table from being executed. If an event code in the Frame Table attempts to start the macro when it has been disabled, the event code is ignored.	0 = Macro A is enabled 1 = Macro A is disabled

NOTE:

Writes to DSP_EG_MD will not be reflected in the MCU register EG_MD, and vice versa.

NOTE:

Enabling/Disabling macros will take 4-8 DSP clocks to take effect due to synchronization issues.

50.16 Appendix

The following example is not GSM specific

50.16.1 Example Traffic Channel Timing

This appendix gives example timing for a receive burst, transmit burst, and power measurement.

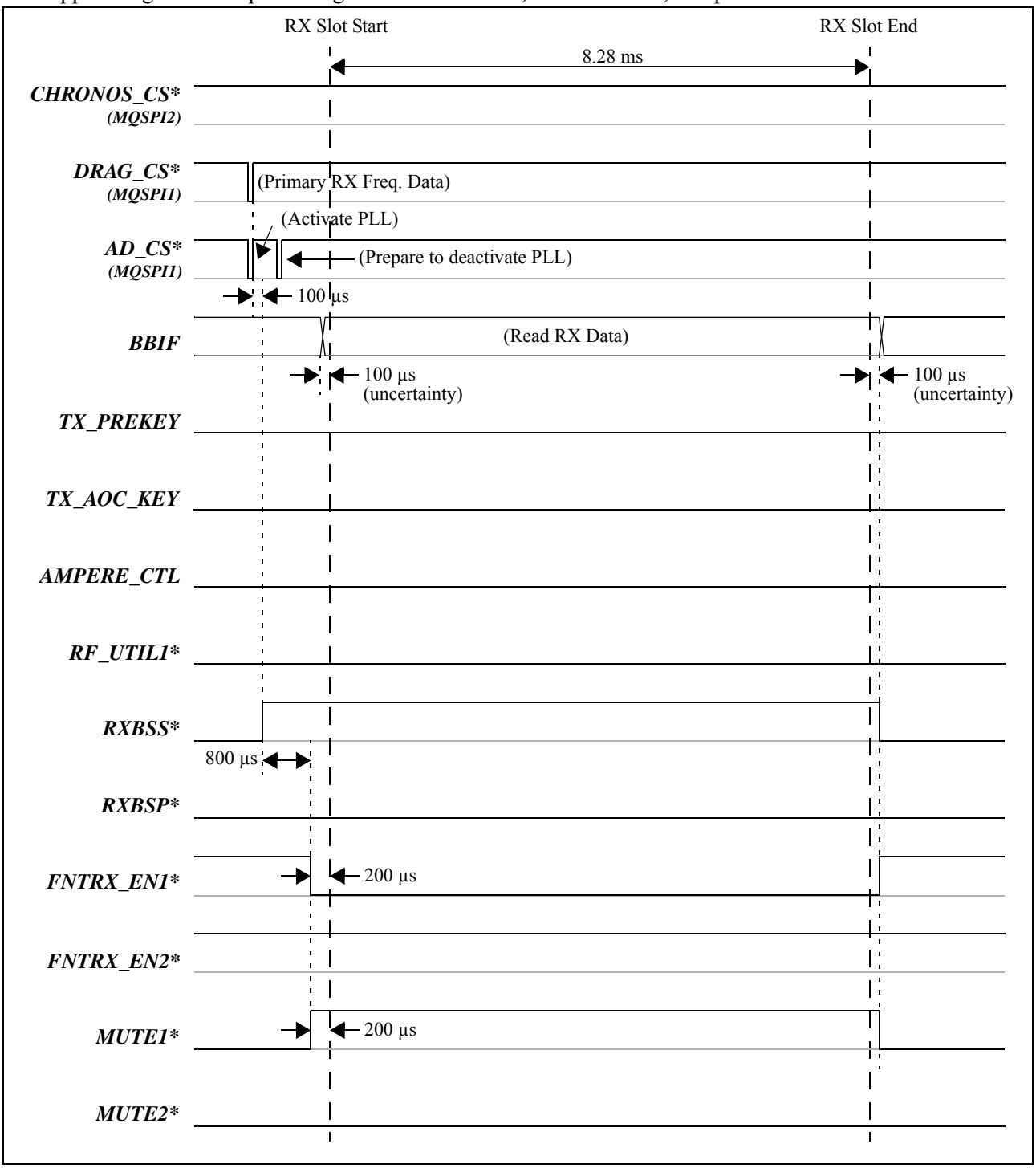


Figure 50-11. Example TCH Receive Timing Sequence

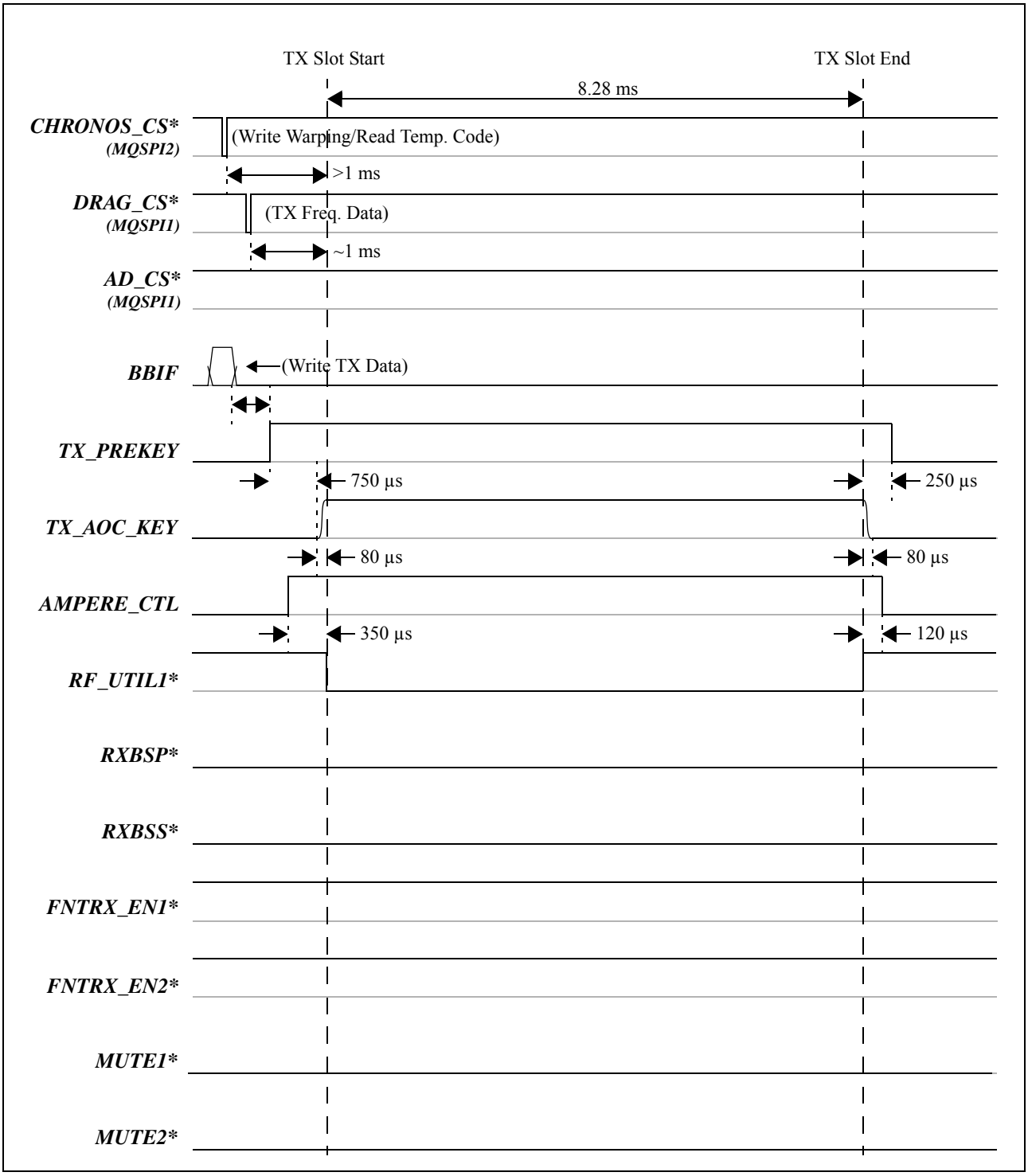


Figure 50-12. Example TCH Transmit Timing Sequence

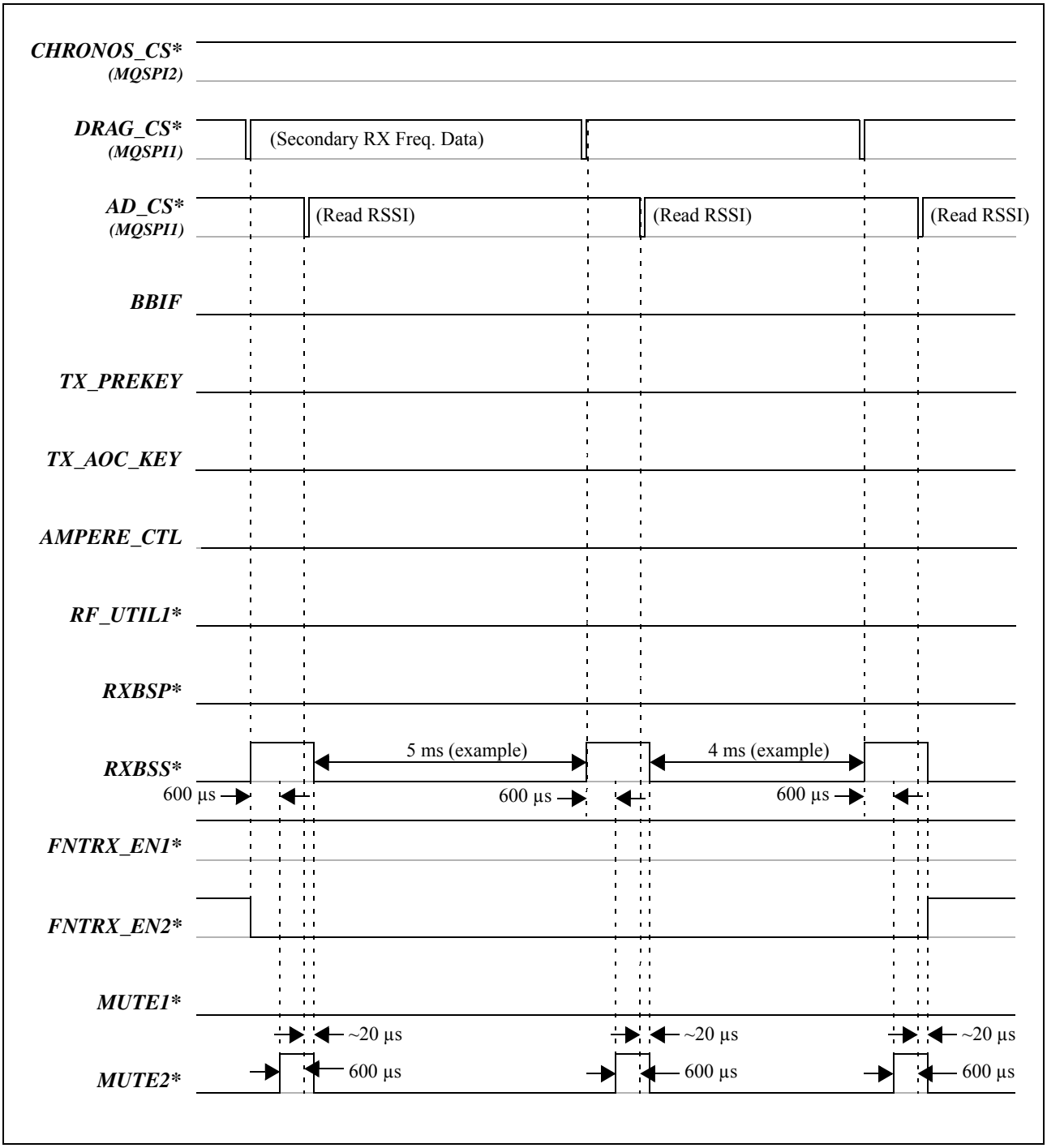


Figure 50-13. Example Power Measurement Timing Sequence

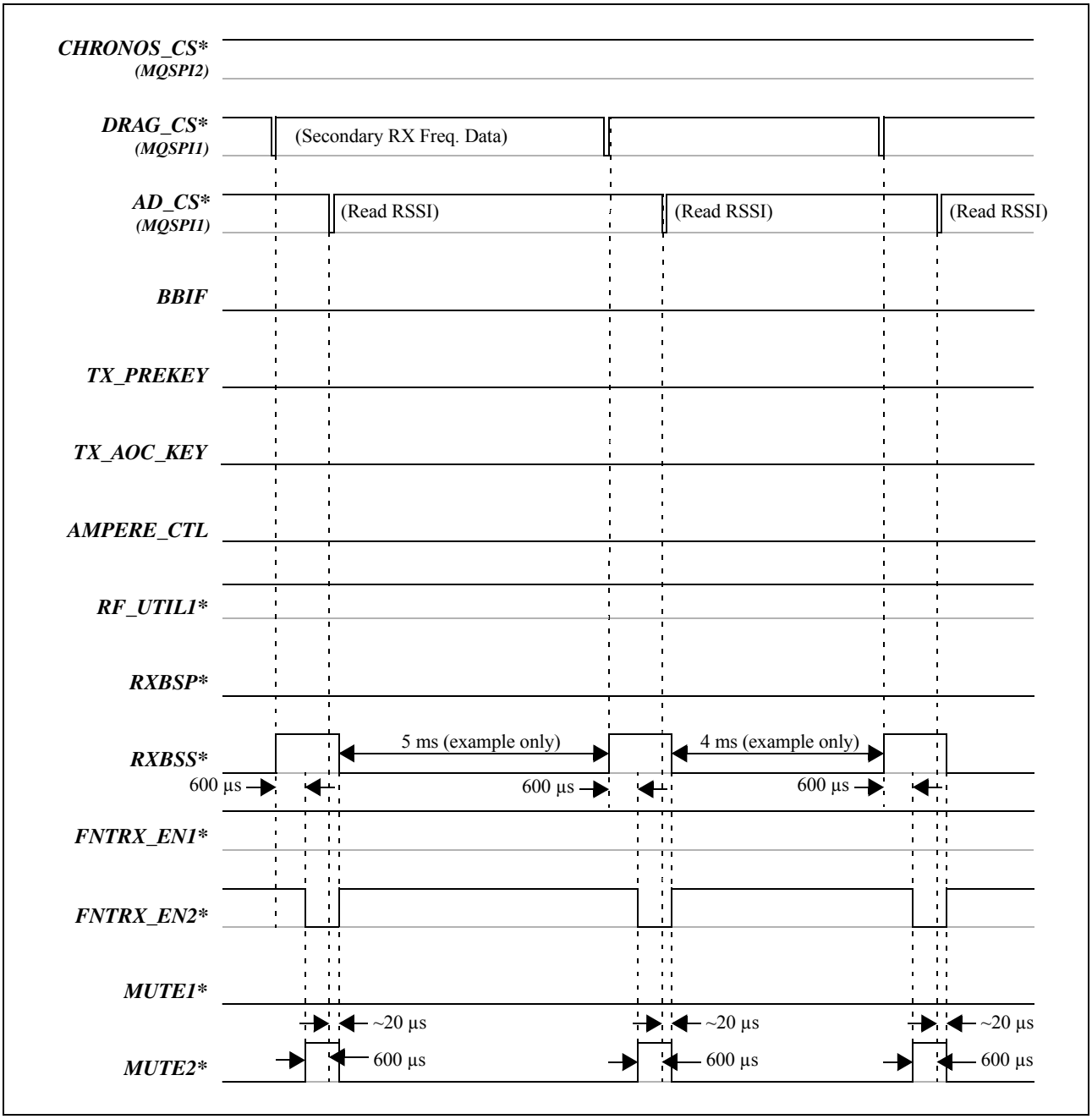


Figure 50-14. Example Alternate Power Measurement Timing Sequence

50.16.2 Example Traffic Channel Frame/Macro Tables

This appendix identifies a possible SV/ISU TDMA frame timing structure to be used by the L1 Timer. This appendix also attempts to estimate the resource utilization within the L1 Timer Module by defining example frame and macro table definitions that support the TCH timing signals shown in Appendix 50.16.1. These examples are used to help justify the changes to the L1 Timer to meet the needs of the Baseband/RF hardware. The L1 Timer is analyzed with the view to reduce the amount of software overhead and eliminate software intervention. The goal is to have the software operate at the frame level and not the sub-frame level. To achieve this, the L1 Timer must also be closely analyzed with the MQSPI module.

50.16.2.1 Frame Timing Overview

The following TCH Frame/Macro Table scenario shows an example of receiving a TCH burst on downlink time slot 1 and uplink time slot 1 with power measurements taken for BCCH downlink time slots 1 and 4. The frame structure for this example begins 1 millisecond prior to the end of uplink time slot 4 (relative to the timing at the SV). That is, the Matching Quarter Bit Counter (**mqbc_count[15:0]**) resets to 0 at 1 millisecond before the end of the uplink time slot 4. All timing assumes to be relative to the SV. Event Quarter Bit Counts (EQBC) and Relative Quarter Bit Counts (RQBC) are based on a 16.8 MHz system clock and the Quarter Bit Clock Prescaler set to divide by 24. This scenario uses the worst case secondary receiver timing where the MUTE2* and RXBSS* are disabled between power measurements. An overall TDMA frame structure timing for this scenario is shown in Figure 50-15.

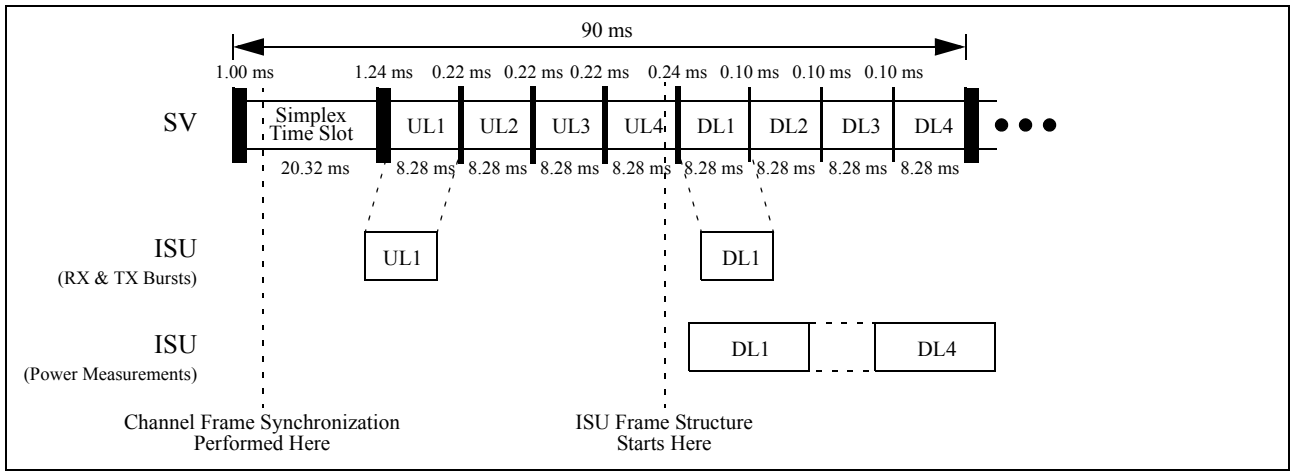


Figure 50-15. Example SV/ISU TDMA Frame Structure Timing

NOTE:

Since the frame timing will need to be periodically changed to account for propagation delay changes and synchronization improvements, the L1 Timer frame comparator will be adjusted (using the Quarter Bit Count Offset, QBCO). This can cause problems for the frame table when the QBCO is modified. Refer to Section 50.4.9, “QBCO Handler,” for the QBCO changes. By placing the frame structure starting point at 1 millisecond prior to the end of uplink time slot 4 and assuming a minimum SV to ISU propagation delay of approximately 2 milliseconds, all macro and frame table events should be inactive and thus avoiding conflicts when small timing adjustments are made to the QBCO using synchronous mode updates. By using the new L1 Timer event code synchronous update, the update can take place at anytime, such as after the receive burst (to read timing change information) and before the transmit burst (to implement the new changes). An expanded view of the frame timing at the point where the TDMA frame structure begins is shown in Figure 50-16. Note that the power measurement and receive bursts have a limitation on how soon it can start for downlink time slot 1, however, this is not the case for the remaining three downlink time slots. However, the transmit burst does not have a limitation on how soon it must complete for uplink time slot 4 since macros, once started, run independently of the frame table.

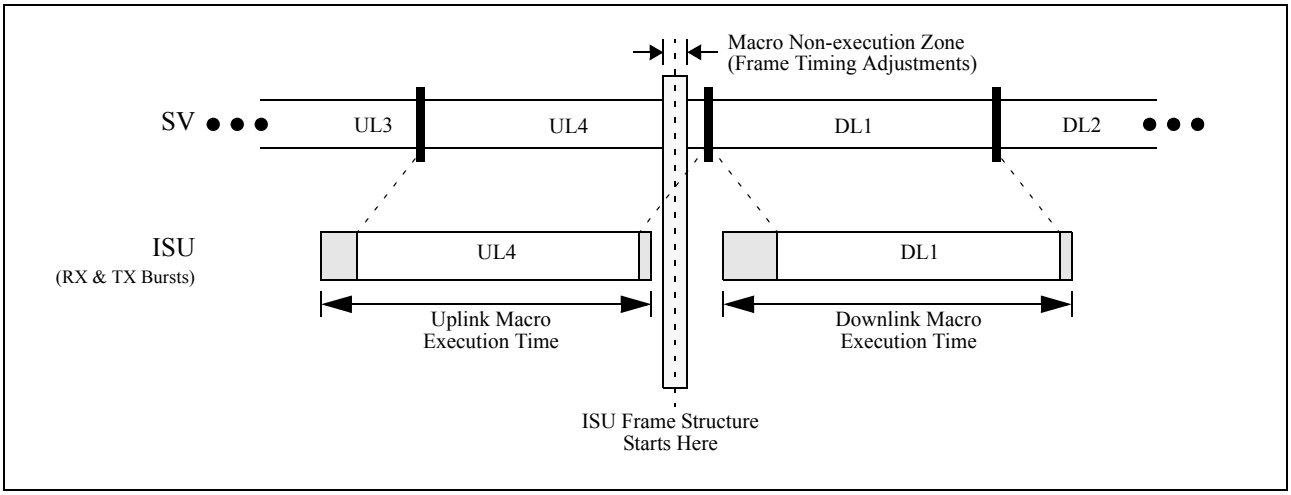


Figure 50-16. Expanded ISU TDMA Frame Structure Rollover

50.16.2.2 Downlink Frame Timing

For this example, the receive macro (Macro A) will be executed at the beginning of the downlink time slot relative to SV time. This assumes that the propagation delay is greater than the receive setup time. Since the ISU's frame time starts 1 msec before the end of uplink time slot 4, and the guard time between uplink time slot 4 and downlink time slot 1 is 0.24 msecs, Macro A will be executed for time slot 1 at ISU frame time 1.24 msecs, which is Event Quarter Bit Count (EQBC) of 868 (0.00124 * 16.8 MHz / 24). This EQBC value is used in the Frame Table to execute the Macro A. Table 50-91 defines the ISU frame time for each downlink slot.

Table 50-91. ISU Downlink Frame Times

Downlink Time Slot	ISU Frame Time (ms)	Event Quarter Bit Count (EQBC)
1	1.24	868
2	9.62	6734
3	18.00	12600
4	26.38	18466

For this example, the receive macro uses delay register 0. This register holds the calculated/measured propagation delay from the SV minus the setup time required by the ISU's receiver. For this example, the propagation delay is assumed to be calculated/measured at 2.00 msecs. The receiver setup time is a fixed number that is calculated from the receive macro table. Refer to Appendix 50.16.4.3 for the setup time, which is 912 µsecs, or 638 quarter bit counts. The delay register 0 is calculated as:

2.00 ms (propagation delay) - 912 µs (RX setup) - 100 µs (RX settling) - 100 µs (uncertainty)

= 888 µs, or 622 quarter bit counts.

The delay register could also be used to select the desired downlink time slot.

50.16.3 Uplink Frame Timing

The transmit macro (Macro B) must be executed in advance of the uplink time slot relative to SV time. At a minimum, the transmit macro must be executed before the uplink time slot by the propagation delay plus the transmitter setup time. For this example, it is assumed that the worst case propagation delay is 10 msecs. Also, the transmitter setup time is assumed to be less than 1 msec. Therefore, all transmit frame times will be 11 msecs before uplink time slot relative to SV time. Since the ISU's frame time starts 1 msec before the end of uplink time slot 4, Macro B will be executed at ISU frame time 46.22 msecs, which is Event Quarter Bit Count (EQBC) of 32354 (0.04622 * 16.8 MHz / 24). This EQBC value is used in the Frame Table to execute the Macro Table. Table 50-92 defines the ISU frame time for each downlink slot.

Table 50-92. ISU Uplink Frame Times

Uplink Time Slot	ISU Frame Time (ms)	Event Quarter Bit Count (EQBC)
1	46.22	32354
2	54.72	38304
3	63.22	44254
4	71.72	50204

For this example, the transmit macro uses delay register 1. This register holds the 11 msec offset minus the calculated/measured propagation delay from the SV minus the setup time required by the ISU's receiver. For this example, the instantaneous propagation delay is assumed to be calculated/measured at 2.00 msecs. The transmitter setup time is a fixed number that is calculated from the transmit macro table. Refer to Appendix 50.16.4.4 for the setup time, which is 840 µsecs, or 588 quarter bit counts. The delay register 1 is calculated as:

11 ms - 2.00 ms (propagation delay) - 840 µs (TX setup) = 8.160 ms, or 5712 quarter bit counts.

The delay register could also be used to select the desired uplink time slot.

50.16.4 Power Measurement Frame Timing

For this example, the power measurement macro (Macro C) will take three power measurements in time slot 1 and 5 power measurements in time slot 4. This macro is capable of operating simultaneously with the receive macro (Macro A). The current power measurement concept is to take up to 6 separate measurements per time slot but a maximum of 8 measurements per frame. Measurement times are dependent upon the time slot, propagation delays, etc. Since determining the measurement times is beyond the scope of this document, it will be assumed that eight measurements will be taken, and that the time periods between measurements may vary.

The first measurement is limited on how early it can start for time slot 1 since the frame timing begins 1.24 msecs before uplink time slot 1 (see Appendix 50.16.2.1). All timing must be inactive near the beginning and end of the ISU frame timing for possible synchronous timing adjustments.

Since executing a macro can specify many delay parameters, these delays will be used as the delay periods between each measurements where the time period between each measurement can be different. The macro will use the new delay register 2 to define the initial delay for all measurements. The macro can be executed once to cover both time slots or the macro can be called twice, once for each time slot. For this example, the macro will be executed only once to perform measurements in time slots 1 and 4.

50.16.4.1 Frame Table 0 (Odd Frames)

The Indy software only took power measurements on odd frames. For this example, Frame Table 0 is set up for this odd frame scenario and is shown in Table 50-93. Frame Table 1 is used to set up the even frame scenario. The Channel Frame Interrupt signal (**cfi**) switches between Frame Table 0 and Frame Table 1. The synchronous update for this example is performed at 40 ms (28000 quarter bit clocks) into the frame, which is 6.22 ms before the macro call for the transmit burst.

Table 50-93. TCH Frame Table 0

Event #	Event Quarter Bit Count (EQBC)	Event Name	Comment
E0	400	MACRO_C_4	Start TCH Power Measurement scenario
E1	868	MACRO_A_0	Start TCH Receive scenario, time slot 1
E2	28000	SYNC_UPDATE	Perform synchronous L1 Timer update
E3	32354	MACRO_B_2	Start TCH Transmit scenario, time slot 1

50.16.4.2 Frame Table 1 (Even Frames)

On even frames, the software copies the power measurement data and updates the candidate power measurement assignments as well as the candidate RF programming values upon receiving an ARMI2 interrupt from the L1 Timer. The Frame Table in Table 50-94 can be used to show how this can be accomplished with no power measurements taken during this even frame. The synchronous update for this example is performed at 40 ms (28000 quarter bit clocks) into the frame, which is 6.22 ms before the macro call for the transmit burst.

Table 50-94. TCH Frame Table 1

Event #	Event Quarter Bit Count (EQBC)	Event Name	Comment
E0	868	MACRO_A_0	Start TCH Receive scenario, time slot 1
E1	28000	SYNC_UPDATE	Perform synchronous L1 Timer update
E2	32354	MACRO_B_2	Start TCH Transmit scenario, time slot 1

50.16.4.3 TCH Receive Macro Table (Macro Table A)

This example uses Macro Table A to define the TCH receive burst scenario and is shown in Table 50-95. This macro implements the timing shown in Figure . The shaded area represents the setup events needed before the actual receive burst. The setup time is the number of events plus any additional Relative Quarter Bit Counts per event. (An RQBC of 0 is 1 quarter bit count, an RQBC of 1 is 2 quarter bit counts, etc.) The setup time is a total RQBC of 638, which is 912 µsecs (638/(16.8MHz/24)). The macro event begins after the delay defined in the DELAY_REG0 register, which is based on propagation delay and the setup time.

Table 50-95. TCH Receive Macro Table A

Event #	Relative Quarter Bit Count (RQBC)	Event Name[1]	Comment
E0	0	DELAY_REG0	Receive burst delay based on propagation delay and setup (and possibly slot selection)
E1	0	MQSPI_5	Dragster data program (program primary receive frequency)
E2	0	LEADI0	Activate DSP interrupt to wake up DSP from its IDLE mode state.
E3	0	ARMI0	L1_RX_INT interrupt
E4	0	MQSPI_6	AD7342 data program (activate PLL)
E5	69	RXBSP*1	Wait 100[2] µsecs, then deassert RXBSP* (activate Dragster's primary receiver)
E6	0	MQSPI_7	AD7342 data program (deactivate PLL, the RXBSP* signal will control the PLL)
E7	560	FNTRX_EN*0	Wait 800[3] µsecs, then assert FNTRX_EN*
E8	0	MUTE1*1	Deassert MUTE1* (activate Transconverter's primary receiver)
E9	0	DELAY_TBL	Wait while receiving data[4]
E10	0	FNTRX_EN*1	Deassert FNTRX_EN*
E11	0	MUTE1*0	Assert MUTE1* (deactivate Transconverter's primary receiver)
E12	0	RXBSP*0	Assert RXBSP* (deactivate Dragster's primary receiver)
E13	0	EOM	End of Macro

1. Some event names (LEADI0 and ARMI0 for example) refer to opcodes used in a previous implementation of the timer. They are include here for reference only.
2. AD7342's PLL settling time (100 µs).
3. Dragster settling time (600 µs) + AD7342 calibration time (200 µs) = 800 µs.
4. Slot time (8.28 ms) + RX front end settling time (100 µs) + uncertainty (200 µs).

50.16.4.4 TCH Transmit Macro Table (Macro Table B)

This example uses Macro Table B to define the TCH transmit burst scenario and is shown in Table 50-96. This macro implements the timing shown in Figure . Shaded areas in Table 50-96 indicate all of the events consisting of transmit burst setup. The setup time is the number of events plus any additional Relative Quarter Bit Counts per event. (An RQBC of 0 is 1 quarter bit count, an RQBC of 1 is 2 quarter bit counts, etc.) The setup time is a total RQBC of 587, which is 839 µsecs (588/(16.8MHz/24)). The macro events begin after the delay defined in the DELAY_REG1 register, which is based on propagation delay and setup times.

Table 50-96. TCH Transmit Macro Table B

Event #	Relative Quarter Bit Count (RQBC)	Event Name	Comment
E0	0	DELAY_REG1	Transmit burst delay based on propagation delay and setup
E1	0	LEADI1	Activate DSP interrupt to wake up DSP from its IDLE mode state
E2	0	ARMI1	L1_TX_INT interrupt
E3	0	MQSPI_4	Dragster data program (program primary transmit frequency)
E4	0	MQSPI_9	Write Customer Warp Word, Read A/D Temperature Code
E5	0	TX_PREKEY1	Assert TX_PREKEY (activate Transmitter)
E6	335	AMP_CTL1	Wait 480 µs, assert AMP_CTL (activate Amperes)
E7	244	RF_UTIL0	Wait 350 µs, deassert RF_UTIL (slow power control loop response)
E8	0	DELAY_TBL	Wait pulse width (8.28 ms)
E9	0	RF_UTIL1	Assert RF_UTIL (fast power control loop response)
E10	83	AMP_CTL0	Wait 120 µs, deassert AMP_CTL (deactivate Amperes)
E11	90	TX_PREKEY0	Wait 130 µs, deassert TX_PREKEY (deactivate Transmitter)
E12	0	ARMI1	TX_PREKEY interrupt
E13	0	EOM	End of Macro

50.16.4.5 TCH Power Measurement Macro Table (Macro Table C)

This example uses Macro Table C to define the TCH power measurement scenario and is shown in Table 50-97. This macro implements the timing shown in Figure . All measurements, are delayed according to the DELAY_REG2 register. A loop structure is then formed to repeat for each measurement. The loop executes and repeats the events E2 through E12 the number of times defined by the loop count in the EG_LP0 register. If the loop count is 0, the macro will immediately terminate (as if an EOM event was executed). Next, event E3 performs a delay to separate the power measurements in time. The event code DELAY_TBL reads the delay from the Parameter Table. The remaining events through E12 perform a power measurement.

Events E3, E7, and E8 use the new Multiple Message mode feature of the MQSPI. In addition, events E3 and E8 also use the new Burst mode feature of the MQSPI. Multiple Message mode allows several back-to-back messages to be sent to the same device using only 1 trigger. Multiple Message mode with Burst mode is similar with the exception that a trigger must occur for each message in the multiple

message queue. Other messages are allowed to be sent on the SPI in between each burst message. Therefore, event E7's MQSPI_8 sends the same messages through every pass of the loop. Event E3's and E8's MQSPI_0 and MQSPI_1, respectively, sends a different message for every pass of the loop.

Event E12's END_LOOP decrements the loop count loaded by the LOOP_REG0 event code. If the count is not 0, the event pointer resets back to event E2 to repeat the loop. If the count is 0, the loop terminates and event E13 gets executed on the next quarter bit clock. Event E13 executes the new ARMI2 event code to interrupt the ARM processor to analyze the power measurement data read using MQSPI_1.

Table 50-97. TCH Power Measurement Macro Table C

Event #	Relative Quarter Bit Count (RQBC)	Event Name	Comment
E0	0	DELAY_REG2	Perform general purpose delay before performing any power measurements
E1	0	LOOP_REG0	Repeat (loop) the following event codes the number of times defined in EG_LP0 register. (A loop count of zero terminates the macro, similar to EOM.)
E2	0	DELAY_TBL	Delay according to the current delay in the parameter table (also, increment the pointer to the next delay in the table)
E3	0	MQSPI_0	Dragster secondary receive frequency (Multiple Message Burst Mode)
E4	0	RXBSS*1	Deassert RXBSS* (activate Dragster's secondary receiver)
E5	419	MUTE2*1	Wait 600 µs for Dragster settling time, then deassert MUTE2* (activate Transconverter's secondary receiver)
E6	0	FNTRX_EN2*0	Assert FNTRX_EN2* (activate receiver front-end)
E7	419	MQSPI_8	Wait 600 µs for RSSI settling time, then initiate the A/D Conversion of RSSI, and specify reading of RSSI data (Multiple Message)
E8	8	MQSPI_1	Wait 12 µs for A/D conversion, then read RSSI Data (Multiple Message Burst Mode)
E9	14	RXBSS*0	Wait ~20 µs for RSSI measurement, then assert RXBSS* (deactivate Dragster's secondary receiver)
E10	0	MUTE2*0	Assert MUTE2* (deactivate Transconverter's secondary receiver)
E11	0	FNTRX_EN2*1	Deassert FNTRX_EN2* (deactivate receiver front-end)
E12	0	END_LOOP	Repeat loop
E13	0	ARMI2	Power Measurement interrupt
E14	0	EOM	End of Macro

50.16.4.6 Delay Registers

The Delay Registers are used as programmable start delays for any of the three macros. Refer to Table 50-98 for examples of the Delay Register values. Refer to Appendix 50.16.2.2, Appendix 50.16.3, and Appendix 50.16.4 for the calculations of these delays.

Table 50-98. Delay Registers

Register	Relative Quarter Bit Count (RQBC)	Comment
EG_D0_QD	622	TCH Receive burst start delay
EG_D1_QD	5712	TCH Transmit burst start delay
EG_D2_QD	User Defined	Power Measurement start delay
EG_D3_QD	0	Not Used (Spare)

50.16.4.7 Loop Registers

The Loop Registers are used as programmable loop counts for any of the three macros. Refer to Table 50-99 for examples of the Loop Register values.

Table 50-99. Loop Registers

Register	Relative Quarter Bit Count (RQBC)	Comment
EG_LP0	8	Power Measurement loop count (8 power measurements)
EG_LP1	—	Not Used (Spare)
EG_LP2	—	Not Used (Spare)
EG_LP3	—	Not Used (Spare)

50.16.4.8 Parameter Table

The Parameter Table is used to store delay and loop parameters for all 3 macro tables. Macro calls specify an address (parameter pointer) into the Parameter Table. Macro calls can only specify parameter pointers starting at even addresses. Even though Macro A and Macro B only use 1 parameter, they consume two parameter entries in the Parameter Table.

Table 50-100. Example Parameter Table

Parameter #	Relative Quarter Bit Count (RQBC)	Comment
P0	6004	TCH Receive timing pulse width (8.28 ms slot + 100 µs settling + 200 µs uncertainty)
P1	—	Not Used (Spare)

Table 50-100. Example Parameter Table

Parameter #	Relative Quarter Bit Count (RQBC)	Comment
P2	5796	TCH Transmit timing pulse width (8.28 ms slot)
P3	—	Not Used (Spare)
P4	3500	1st Power Measurement delay (5 ms)
P5	2800	2nd Power Measurement delay (4 ms)
P6	User Defined	3rd Power Measurement delay
P7	User Defined	4th Power Measurement delay
P8	User Defined	5th Power Measurement delay
P9	User Defined	6th Power Measurement delay
P10	User Defined	7th Power Measurement delay
P11	User Defined	8th Power Measurement delay
P12–P31	—	Not Used (Spare)

50.17 References

1. SATCAP L1 Timer Change Specification, Motorola SSTG, Version 2.4
2. GSM WHITECAP Layer 1 Timer Specification, Motorola CSG, Version 2.0
3. REDCAP2 Layer 1 Timer, Revision 2.1, Motorola Semiconductor Israel
4. Patriot Layer 1 Timer DSP Requirements, Motorola CSG, PACSG, December 29, 1998
5. IRIDIUM Air Interface Specification, Satellite Communications Division, SPC-E0003.SYS, Rev. E
6. CTM Utilization in the MC68338, SSP-ITO-CPSW-RPT-0001: Version 1.0
7. AD7342 Complete 3V Codec Specification, Analog Devices Handbook, Preliminary Technical Information
8. DRAGSTER Contract Book, Motorola CSG, Version 0.1
9. MQSPI Specification, Motorola CSG, Version 2.0
10. Satcap Timer Module & MQSPI Module Change Proposal, Motorola SSTG, Version 1.0
11. Satcap Changes, Motorola CSG, 9 September 1997
12. Laguna Timing Diagrams, Motorola CSG, 5 December 1997, Drew Raclaw

50.18 Glossary

- AEQBC:Absolute Event Quarter Bit Count
- MCUIO:MCU Interrupt 0 (OR'd interrupt port)
- MCUI1:MCU Interrupt1 (OR'd interrupt port)
- CFI:Channel Frame Interrupt
- CI:Channel Interrupt (OR'd interrupt port)
- DSM:Deep Sleep Module
- EC:Event Code
- I/Q:In-Phase/Quadrature Phase Signals
- ISU:Iridium Subscriber Unit
- LEAD: Satcap DSP Core
- DSP56600: Neptune DSP Core
- DVI: DSP Vectored Interrupt
- MARD:Modulo Adder and Rollover Detector
- MCU:Microcontroller Unit
- PIT:Periodic Interrupt Timer
- PITI:Periodic Interrupt Timer Interrupt
- QBC:Quarter Bit Count
- QBCM:Quarter Bit Count Modulus
- QBCO:Quarter Bit Count Offset
- QBCP:Quarter Bit Clock Prescaler
- REQBC:Relative Event Quarter Bit Count
- RFI - Reference Frame Interrupt
- RFCI - Reference Frame Counter Interrupt
- RQBC:Reference Quarter Bit Count
- VAB: Vector Address Bus (to DSP interrupt handler)
- SV:Space Vehicle (Satellite)
- TBD:To Be Determined

50.19 RTL Design Hierarchy

Layer 1 Timer Top

- Tristate drivers for scan
- Layer 1 Timer Wrapper
- JIG
- FMIG16
- BOG
- bus keepers
- Layer 1 Timer (core logic)
- clock gates
- asynchronous reset bypasses
- MSTR register
- GTC 0 register
- GTC1 register
- GTS0 register
- wakeup controller
- programmable wakeup timer
- sleep/wake state machine
- pin drive controller
- bus interface
- timing block
- prescaler
- reference quarter bit counter (RQBC)
- rafncfn
- reference absolute frame number (RAFN)
- channel frame number (CFN)
- rfc_cfc
- reference frame counter (RFC)
- channel frame counter (CFC)
- pit
- cfic - add RQBC, QBCO to get MQBC
- generate channel frame marker
- event generator (eg_main)
- EQBC RAM (192x16)
- EC RAM (192x8)
- QBCO_UDT_F register

QBCO_UDT_Q register
QBCO_UC0 register
QBCO_UC1 register
EG_FT0_CFG register
EG_FT1_CFG register
EG_NXT_CFG register
EG_MA_CFG register
EG_MB_CFG register
EG_MC_CFG register
EG_FPTR register
EG_MPTR0 register
EG_MPTR1 register
EG_D0_FD register
EG_D0_QD register
EG_D1_FD register
EG_D1_QD register
EG_D2_FD register
EG_D3_FD register
EG_D3_QD register
EG_LP0 register
EG_LP1 register
EG_LP2 register
EG_LP3 register
PCFG register
PB register
PS register
PTR register
PDIMASK register
EG_MD register
write_errtrap
macro a write trap
macro b write trap
macro c write trap
boundary pointer write trap
frame table write trap
address generation unit (AGU)

event decoder (ED)

pin/interrupt generator (PIG)

pin control unit (PCU)

16 pin drivers

mqspi trigger driver

delay execution unit (DEX)

3 loop counters (1 per macro)

3 delay counters (1 per macro)

frame table control unit (FTCU)

macro table control unit (MTCU)

macro a controller

macro b controller

macro c controller

interrupt generator

status & mask for RFI, RFCI, FFI, CI, MCUINT0, MCUINT1

mcu interrupts

drivers for DSPWAKE, DVI14:DVI0 DSP interrupts

GTS1 status register

IGM interrupt mask register

MCUIS status register

MCUIM status register

DVIM register

error detector - status & mask registers for error conditions

ES0 error source register

EM0 error mask register

ES1 error source register

EM1 error mask register

ES2 error source register

EM2 error mask register

xint

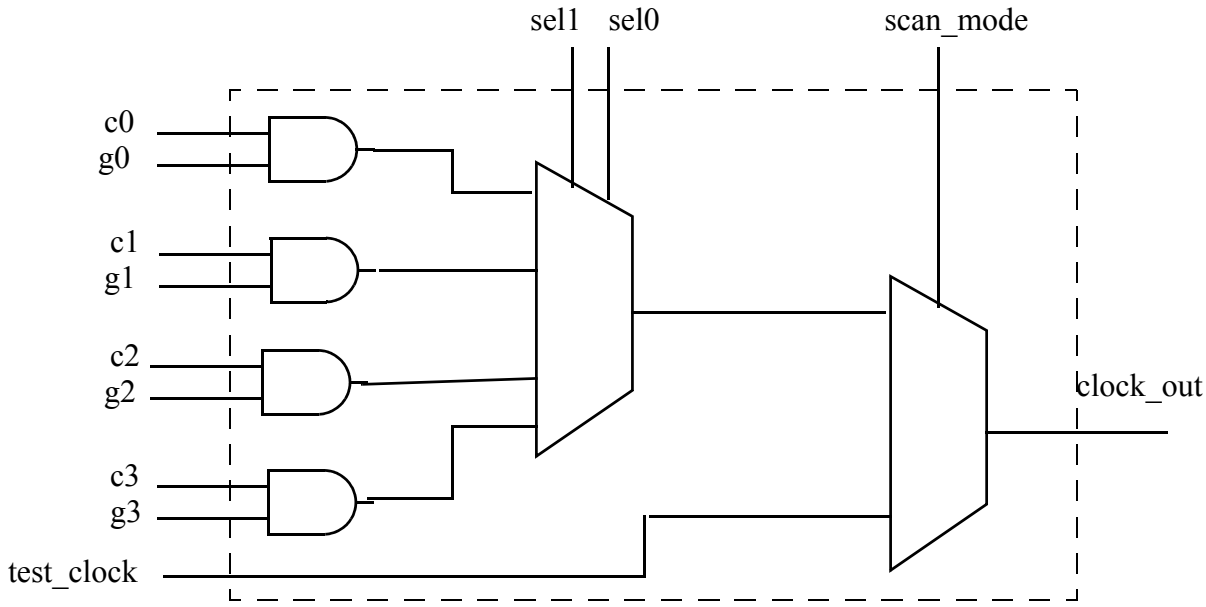
status hold, prioritization, and vector generator logic for

L1RX, L1TX and DVI0:DVI14 DSP interrupts

50.20 Clock Gating

The Layer 1 timer design includes extensive clock gating logic for minimization of power dissipation during idle and run modes. While the addition of clock gates complicates design synthesis and debug, it is well-justified since there are large sections of the logic which run at a very low duty cycle. This section is intended to give the designer familiarity with the RTL implementation of the Layer 1 Timer and a synopsis of how and when the various clocks within the design are generated.

The generic clock gating module used is shown below:



where: c0, c1, c2, c3 are input clocks, either PAT_REF, MCU_CLOCK, or PCLK; g0, g1, g2, g3 are clock gates; sel0, sel1 are clock selection conditions; and scan_mode and test_clock are for test.

Chapter 51
Smartcard Interface Module (SIM)

This section details the features, operation, and programming interface to the Smartcard Interface Module(s).

Revision History Table

Version	Date	Author	Changes
1.0	10/8/99		Initial Release.
1.1	01/12/2000	Keith Kinerk	• Complete documentation of T=1 Changes for the Patriot ROM1 part - Programming model update - Description of new features • All known errata for Patriot RAM1 SIM module is documented within • Other minor cosmetic changes, clarifications
1.2	03/06/2000	Keith Kinerk	• Added DIVISOR Register description to register section • Moved CHAR_WAIT and GPCNT register descriptions below RESET_CNTL • Added DIVISOR register to programming model section
1.3	03/29/2000	Keith Kinerk	• Added detail to section 33.7.7 about how to enable the GPCNT counter and when it starts counting • Added references to GPCNT register definition and GPCNT clksel bits definition in CNTL register

Revision History Table

Version	Date	Author	Changes
1.4	05/12/2000	Nazim Aziz	• Err: Table 51-25; cwtm and gpcntm bits default to 1, instead of 0 as specified previously. • Err: Table 51-23; tdt reg referred to as tft. Changed to the correct name. • Added Section 51.2.1.3. • Update: Section 51.2.5. • Update: Section 51.3.1 • Update: Section 51.3.4 • Update: Section 51.4.1 • Update: Table 51-13 • Update: Table 51-21 • Update: Table 51-24 • Update: Table 51-25 • Update: Section 51.9.2.1 • Update: Figure 51-4
1.5	11/22/2000	Nazim Aziz	• TFE is not set when TX FIFO cleared by setting the flush TX bit.
1.6	01/25/2001	Nazim Aziz	• Err: Table 51-31; RCVR11 bit defaults to 0, instead of 1 (specified in the previous versions).
1.7	02/19/2001	Nazim Aziz	• Err: Table 51-20; bit description edited to show the correct value for ICM and ANACK bits. • Err: Figure 51-21; Generator Polynomial and Expected Residual values corrected.
1.8	04/03/2001	Nazim Aziz	Errata: SIM Operation in DOZE mode Clarification: sfpd bit clears after 5 CKIL cycles.
1.9	06/28/2001	Nazim Aziz	• T=1 SIM card hardware re-design • Updates for Patriot Indy
2.0	04/17/2002	Richard Lubeck	• Updated Specification for Neptune LTS. Removed IPLAMB / PIG interface and replace with IPBUS interface
2.1	08/19/02	Shannon Osgood	• Updated per DDTS# DSPhl15156 - added clarification to section 51.1.3.
2.2	05/07/03		Updated for LTE specification release.

51.1 Errata History

51.1.1 Patriot RAM1 Errata

This section will detail known errata for the Patriot SIM module that is implemented on the initial Patriot RAM1 device. A suggested workaround is also provided.

- Invalid Initial Character not flagged in RX_FIFO
  - When the SIM receiver is configured to operate in Initial Character Mode (**icm** = 1), there is the possibility that a character could be falsely decoded by the software as a valid initial character. This situation occurs if a $3F is decoded without a parity error while in Initial Character Mode. The SIM hardware will reject this as an initial character and remain in Initial Character Mode. However, when the software searches for a valid initial character in the receive FIFO, it will see a valid initial character and start to process the following data as part of the Answer To Reset (ATR).
  - Workaround: The workaround for this condition is to check the status of the **icm** and **ic** bits in the CNTL and DATA_FORMAT registers. The **icm** bit is automatically cleared when a valid initial character is received. Software should poll on this bit until it is cleared. Another option is to set the receive threshold (**rdt4:0]**) to 1 and check the value of **icm** after each received character. Once the **icm** bit is cleared, the software can read the initial character out of the receive FIFO. The **ic** bit will reflect which initial character was detected by the SIM hardware. If **ic** is set, a $3F was received. If **ic** is clear, a $3B was received. The characters following the initial character are part of the ATR. The receive threshold (**rdt[4:0]**) can be set to a higher level once the initial character has been received.
- Port 1 Open Drain control reversed
  - The **OD_P1** bit in the OD_CONFIG register is inverted in the open drain control for the SIM module. This causes the Port1 Transmit output to default to open drain mode out of reset. When the **OD_P1** bit is set, the Port 1 Transmit output will operate as push/pull.
  - Workaround: The workaround is to treat this bit as the inverse polarity of the Port0 bit. Setting the **OD_P1** bit will disable open drain mode for Port 1. Clearing it will enable the open drain mode for Port 1.
- 3voltX bits unreadable
  - The **3volt0** and **3volt1** bits in the PORT0_CNTL and PORT1_CNTL registers are not readable. The bits function properly and can be written to a zero or one, but can only be read as a zero.
  - Workaround: There is no workaround for reading the correct value of the **3voltX** bits.
- 11 ETU Character Transmission
  - When the **getu[7:0]** value in the GUARD_CNTL register is programmed to $FF, the SIM module transmitter should transmit characters with only 1 stop bit making the character length 11 ETU times. The Transmitter does not reduce the character length. Instead, characters are sent with 2 stop bits always regardless of the value of **getu[7:0]**.
  - Workaround: There is no workaround to configure the SIM module to transmit characters with only 1 stop bit.
- Re-transmission with Guardtime > 1
  - When the **getu[7:0]** value in the GUARD_CNTL register is programmed to a value greater than 1, and the **xth[3:0]** value in the XMT_THRESHOLD register is programmed to a value greater than 1, the transmitter will not retransmit a character when a NACK is received.
  - Workaround: If a guardtime greater than 1 is required between characters, then the **xth[3:0]**

value should be set to 1. This will disable the re-transmission capability of the SIM module, but will generate an interrupt when a NACK is detected. The retransmission can then be handled in software.

- Transmit FIFO Empty Interrupt
  - The Transmit FIFO Empty Interrupt (TFE) in the XMT_STATUS register does not act as it was intended. The interrupt was intended to become active after the last byte in the transmit FIFO was transferred to the transmit shift register. Instead, the interrupt becomes active after the last byte has been transmitted. In this way, the interrupt acts exactly how the Early Transmit Complete Interrupt is intended to work (see errata below).
  - Workaround: The EGPT module output compare functionality can be used to generate an interrupt at a time when the transmit FIFO will have sent any number of bytes. The output compare should be set for (T + N*(11+GT)), where T is the current EGPT free-running timer value, N is the number of characters to be sent before the interrupt occurs, and GT is the guard time between characters.
- Early Transmit Complete Interrupt
  - The Early Transmit Complete Interrupt (ETC) in the XMT_STATUS register does not act as it was intended. The interrupt was intended to become active after the last byte in the TX FIFO had been sent, regardless of the value of the GUARD_CNTL register. Instead, the interrupt becomes active after the first byte in the FIFO has been transmitted.
  - Workaround: The TFE Interrupt performs this function exactly how it was intended. Use TFE if this function is needed.
- Transmit FIFO Overfill Interrupt
  - The Transmit FIFO Overfill Interrupt (TFO) in the XMT_STATUS register does not act as it was intended. The interrupt was intended to become active if more than 16 bytes were written to the TX FIFO. Instead, the interrupt flag and mask were both inverted. This causes the TFO interrupt to be active during and after a reset event. The flag will reflect the expected value (i.e. set high) when the transmit FIFO is written to with more than 16 bytes.
  - Workaround: The TFO Mask bit (TFOM) in the INT_MASK register now operates as an interrupt enable bit. Setting this bit will enable the interrupt. Clear it to disable the interrupt. The Interrupt should remain disabled for the workaround. When writing the TX FIFO, the TFO bit can be polled to determine if too many bytes have been written to the FIFO. If the TFO bit is high after a write to the TX FIFO, then more than 16 Bytes have been written. The TFO bit can only be cleared by a Transmit Flush operation or a software reset of the SIM module.
- SIM Presence Detect Status Bit
  - The SPDP1 and SPDP0 bits in the PORT1_DETECT and PORT0_DETECT registers are latched versions of the SIMPD1 and SIMPD0 pins respectively. These register bits were to reflect the current value of the SIMPD1 and SIMPD0 pins.
  - Workaround: The PORT1_DETECT and PORT0_DETECT register must be read twice consecutively to determine the current state of the SIMPD1 and SIMPD0 pins. The value returned will be three MCU_Clock cycles older than the actual state of the pin.

51.1.2 Patriot ROM1 (CDR3) Errata

This section will detail known errata for the Patriot SIM module that is implemented on the Patriot ROM1 device.

At this time, no known errata exist on the Patriot ROM1 version of the SIM module. All errata listed in Section 51.1 for the Patriot RAM1 device has been fixed on the Patriot ROM1 device.

51.1.3 Neptune LT (LCA) Errata

The following errata are applicable only to Neptune LT , they are not applicable to Neptune LTE.

- SIM Operation in DOZE mode
  - SIM shuts down the clocks, when entering DOZE mode, and stops transmission immediately. However, the specification states that the SIM would shut down the clocks after completing the current transmission.
  - Workaround: To work around this problem, avoid setting the DOZE bit in the RESET_CNTL register, preventing the SIM to react to system DOZE mode. Software could either poll for the Transmit Complete (**tc**) flag in the XMT_STATUS register, or setup the Transmit Complete Interrupt, to correctly identify the transmit complete event, before entering doze mode. It is recommended, however, that DOZE bit should not be used to shut off the SIM clocks during system doze mode.
- **kill_clock** bit operation
  - When **kill_clock** bit is set in the **RESET_CNTL** register, or in low-power modes, the receive and the transmit clock could switch to PAT_REF frequency rather than shutting down completely.
  - Workaround: Disable the receiver and transmitter using the **rcv_en** or **xmt_en** bits in the **ENABLE** register**,** before switching to low-power modes or setting the **kill_clock** bit in the **RESET_CNTL** register.

51.2 Enhancements

The section will detail the enhancements made to the Whitecap 1.0 SIM module for Patriot. These features will also be explained in the appropriate sections later in this specification.

51.2.1 New Features

51.2.1.1 Patriot RAM1

- Increased Buffer depth
  - Transmit buffer was doubled from 8 bytes to 16 bytes
  - Receive Buffer was doubled from 16 bytes to 32 bytes
- Software Module Reset (see page 51-75)
  - The current SIM module has no software reset capability. A new control bit was added to reset the SIM module.

- 24X card support (see page 51-56)
  - A divide by 16 mode of operation was added for the transmit baud rate.
- Multiple dividers (see page 51-56)
  - A divide by 3 and divide by 5 circuit was added to allow flexibility on generating the baud clock for the SIM card. This allows support for 19.44, 16.8, and 13Mhz operation.
- Virtual DMA Support via Data interrupt (see page 51-14)
  - A separate Data interrupt was added. This interrupt is active when the receive buffer has reached its programmed threshold, and/or the transmit buffer has generated an interrupt.
- Support for second SIM module (see page 51-37)
  - The second card port on the SIM module now can be muxed with a second SIM module. This provides the ability to talk to two SIM cards simultaneously when an additional SIM module is provided on the IC. Patriot has two SIM modules instantiated.
- Transmit Early interrupt, Transmit FIFO Empty Interrupt (see page 51-61)
  - Two transmit interrupt sources are added. The first becomes active when the last byte in the transmit FIFO has been loaded into the transmit shift register. The second becomes active before the guard time is completed for the last transmitted byte in the transmit buffer. This complements the transmit complete interrupt that becomes active after the guard time expires for the last transmitted byte in the transmit buffer. All interrupt sources have separate mask bits.
- 3V Card I/O option (see page 51-48 and page 51-54)
  - When a SIM port is configured as a three volt port only, the SIM transmit and receive lines can be connected internally. An option bit has been added to connect the transmit and receive lines internally, and disable the connection to the I/O pad. This frees the SIM Transmit pin on Patriot to be used for other purposes via the GPIO module.
- Doze mode support
  - The MCore MCU provides a DOZE instruction for low power operation. A DOZE bit is added to the SIM module to allow software to choose how the SIM reacts to a DOZE instruction. When set, a DOZE instruction acts like a STOP instruction, gating the SIM module clocks after the current transmission is completed. If a transmission is not in progress, the clocks are gated immediately. When clear, the SIM module treats DOZE like a WAIT instruction.

51.2.1.2 Patriot ROM1

These features are new for Patriot ROM1. They are not available on Patriot RAM1.

- T=1 Hardware Support
  - Hardware has been added to the SIM module to provide support for communication with T=1 SIM cards. The hardware changes include: General Purpose Counter, Character Wait Timer, TX FIFO writes while transmitting, Transmit FIFO Threshold interrupt, Linear Redundancy Checking, and Cyclic Redundancy Checking.
- Debug Support
  - The **debug** bit in the RESET_CNTL has been added to allow software to inhibit receive FIFO read pointer updates while in debug mode.
- “No-Stop Clock” SIM Card Support
  - The **stop** bit in the RESET_CNTL has been added to allow the software to configure the SIM clock to continue to run when a STOP instruction is executed by the MCore MCU.
- Flexible Baud Rate Support

— The DIVISOR register was added to allow software to program the value of the divisor used for generating the baud clock and oversample clock. This allows for the use of the SIM module as a simple UART (i.e no flow control) by providing the flexibility of generating most standard UART baud rates...

51.2.1.3 Patriot Hip6W

The following features are new for parts manufactured with Motorola HIP6W technology.

- Receive NACK threshold feature
  - — The Receive NACK Threshold bits were added to the RCV_THRESHOLD register for specifying the number of NACKs that would be generated before triggering a nack threshold error interrupt and forcing a SIM card power down.
- SIM card, force power down sequence
  - — SIM card force power down on port was added to provide a means to initiate the power down sequence manually, which is similar to an auto power down, except the SIM card presence detect trigger, that indicates either a SIM card being connected or removed.

51.2.1.4 Patriot Indy

The following bug fixes are new for part manufactures for Motorola Hip7 technology.

- WWT violation detection (T=0 mode)
  - — Select bit for wwt function is provided to detect WWT violations, which uses the (32-bit) BWT register for etu time comparison.
- CWT counter enhancement
  - — Accuracy on the CWT counter has improved, plus cwtf is now being stored in the receive FIFO to indicate a cwt violation for the last two characters received.
- Block Wait Time (BWT) is now programmable through a 32-bit register.
- Block Guard Time (BGT) is now programmable through an 8-bit register.
- Interrupts / masks for **cwt**, **bwt** and **bgt** are added to the register block.

51.2.2 Neptune LTE

All of the above enhancements / bug fixes that are available on Patriot Indy are also available on Neptune LTE. Patriot Indy SIM was an enhanced version of the Neptune LT SIM, thus the SIM module was ported from the Patriot Indy SIM.

Neptune LTE also has a hardware bug fix for the write-once-to-clear registers in the register block.

51.2.3 Hardware Bug Fixes

These changes were made to fix the hardware bugs that existed on Whitecap and thus remove the need for all existing software workarounds.

- Auto Receive Management (see page 51-87)
  - This feature was added to automatically switch the receiver on after the last byte in the transmit buffer has been sent. Previously, the software had to enable the receiver during the last byte, actively manage NACKs, and throw away the received byte from the receive buffer. The Patriot SIM handles the switch from transmit to receive automatically without software intervention. This feature is always active when the receiver and transmitter are both enabled.
- Hardware Inverse Convention (see page 51-31)
  - Inverse convention cards will automatically be detected in “initial character mode”, and the data received will be converted appropriately.
- Parity bit removed (see page 51-56)
  - Data is always transmitted with even parity if inverse convention support works correctly. The parity bit has been removed.
- Transmit Enable Synchronizer fixed
  - Previously, there was a situation where the transmit state machine could send an additional start bit if the transmitter was disabled and re-enabled too quickly. The Patriot SIM module now handles this situation without restrictions.
- Zero Guard Time support (see page 51-73)
  - Previously, a bug in the hardware did not allow for a guard time of zero between transmitted bytes. The Patriot SIM module supports zero guard time.
- Last Byte guard time (see page 51-61)
  - Previously, the guard time of the last byte was not implemented. This caused the software to ensure that the guard time was not violated between the last transmission of the current buffer, and the first transmission of the next buffer. The transmit complete interrupt has been changed to become active after the guard time is completed. Additionally, the software can enable an early transmit interrupt to allow for refilling the transmit buffer during the last byte guard time.
- Programming Model Improvement (see page 51-81)
  - Previously, the software was not allowed to write to the transmit buffer while the transmitter was enabled. This is allowed on the Patriot version.

51.2.4 SIM Performance Improvements

These changes have no affect on the programming interface to the SIM module, but improve the overall performance with respect to re-usability, power management, clock management, etc.

- Asynchronous Presence Detect
  - Previously, the SIM card presence detection logic required a 32Khz clock to detect if the card is inserted or removed during deep sleep. This detection is now performed asynchronously on Patriot. The 32Khz clock must still be presented to the SIM module in order to power down the SIM card port.

- Reduce gate count
  - — Remove redundant states for guard time counting in Receive State Machine block
  - — Remove redundant guard time counter in Receive State Machine block
  - — Remove extra registers for transmit/receive FIFO entry.

51.2.5 Register Interface Change History

The following bullets list the changes made to the register interface of the SIM module that differentiate it from the Whitecap 1.0 SIM module.

51.2.5.1 Port Control Registers (PORTx_CNTL)

- **sfpd** bit added
  - — This bit was added to provide software the capability to invoke the auto power down sequence without having to deal with port auto power down enable or SIM card presence detect event.

51.2.5.2 Setup Register (SETUP)

- **smode** replaced with **amode**
  - — The **smode** bit was previously used to enable the synchronous SIM interface on Port 0. Synchronous mode is no longer supported. However, the ability to surrender a SIM port to a secondary SIM module was added. The **amode** bit controls whether a second SIM module can access one of the SIM card ports controlled by the primary SIM module.
- wwt_sel bit added
  - — The wwt_sel was added to accommodate the WWT (work waiting time) measurement required for T=0 SIM cards. The bit defaults to a value of zero, which makes the SIM module measure CWT (character wait time), BWT (block waiting time), and BGT (block guard time), required for T=1 SIM cards.

51.2.5.3 Control Register (CNTL)

- **xmt_crc_lrc** added.
  - — This bit was added to the Patriot ROM1 version of the SIM module for T=1 support. This bit is not available on the Patriot RAM1.
- **crcen** added.
  - — This bit was added to the Patriot ROM1 version of the SIM module for T=1 support. This bit is not available on the Patriot RAM1.
- **lrcen** added.
  - — This bit was added to the Patriot ROM1 version of the SIM module for T=1 support. This bit is not available on the Patriot RAM1.
- **cwten** added.
  - — This bit was added to the Patriot ROM1 version of the SIM module for T=1 support. This bit is not available on the Patriot RAM1.

- **gpcnt_clk_sel[1:0]** added.
  - These bits were added to the Patriot ROM1 version of the SIM module for T=1 support. These bits are not available on the Patriot RAM1.
- **sbs[2:0]** replaced by **baud_sel[2:0]**
  - The **sbs[2:0]** bits were renamed to **baud_sel[2:0]**. An extra baud rate divider was defined for **baud_sel[2:0]** = 100. The **baud_sel[2:0]** bits were shifted one bit position to the left in the CNTL register.
- **csrc** replaced by **clk_sel[1:0]**
  - The **csrc** bit was replaced by the **clk_sel[1:0]** bits. New definitions for the **clk_sel[1:0]** bits were defined. These bits provide the ability to choose divide by 3, 4, or 5 of the SIM reference clock input. The option to run the SIM at the reference clock frequency was removed.
- **pt** bit removed
  - The parity bit (**pt**) is no longer needed. It was removed to avoid confusion on the operation of the SIM module.
- **bten** added.
  - This bit was added to complement the cwten bit added for T=1 support functions.

51.2.5.4 Receive Threshold Register (RCV_THRESHOLD)

- **rdt[3:0]** expanded to **rdt[4:0]**
  - The receive FIFO was expanded to 32 bytes, so the receive data threshold (**rdt[4:0]**) was expanded by one bit to allow software to select any value from 1 to 32.
- rth[3:0] added
  - This set of bits were added to integrate the Receive NACK Threshold Error function. See Table 51-21 for details.

51.2.5.5 Transmit Status Register (XMT_STATUS)

- **gpcnt** added.
  - This bit was added to the Patriot ROM1 version of the SIM module for T=1 support. This bit is not available on the Patriot RAM1.
- **tdtf** added.
  - This bit was added to the Patriot ROM1 version of the SIM module for T=1 support. This bit is not available on the Patriot RAM1.
- Status flags **tc**, **xte** now write-one-to-clear
  - The **tc** status flag was previously cleared by reading it set, then writing the transmit FIFO register. The **xte** status flag was previously cleared by setting the **xtc** bit in the XMT_THRESHOLD register. These bits are now write-one-to-clear.
- Status flags **etc**, **tfe, tfo** added
  - Interrupt status flags for the early transmit complete (**etc**), transmit FIFO empty (**tfe**), and transmit FIFO overfill (**tfo**) functions were added.

51.2.5.6 Receive Status Register (RCV_STATUS)

- **cwt** added.
  - This bit was added to the Patriot ROM1 version of the SIM module for T=1 support. This bit is not available on the Patriot RAM1.
- **crcok** added.
  - This bit was added to the Patriot ROM1 version of the SIM module for T=1 support. This bit is not available on the Patriot RAM1.
- **lrcok** added.
  - This bit was added to the Patriot ROM1 version of the SIM module for T=1 support. This bit is not available on the Patriot RAM1.
- Status flag **oef** now write-one-to-clear
  - The **oef** status flag was previously cleared by reading it set, then reading the receive FIFO register. This bit is now write-one-to-clear.
- **rte** added
  - This bit was added on Patriot Hip6W version for the Receive Threshold Error flag. See Table 51-24 for details.
- Block Time violation flags, **bwt** (Block Wait Time) and **bgt** (Block Guard Time) added.
  - These flags were added to support T=1 SIM cards.

51.2.5.7 Interrupt Mask Register (INT_MASK)

- **cwtm** added.
  - This bit was added to the Patriot ROM1 version of the SIM module for T=1 support. This bit is not available on the Patriot RAM1.
- **gpcntm** added.
  - This bit was added to the Patriot ROM1 version of the SIM module for T=1 support. This bit is not available on the Patriot RAM1.
- **tdtfm** added.
  - This bit was added to the Patriot ROM1 version of the SIM module for T=1 support. This bit is not available on the Patriot RAM1.
- Transmit threshold interrupt mask (**xtm**) moved
  - The transmit threshold error interrupt mask (**xtm**) was moved from the XMT_THRESHOLD register to the INT_MASK register.
- Interrupt masks **tfeim**, **etcim** added
  - Interrupt mask bit for the transmit FIFO empty (**tfeim**) and early transmit complete (**etcim**) interrupts were added.
- **rtm** added
  - Interrupt mask bit for the Receive Threshold Error Flag added. See Table 51-25 for details.
- **bwtm** and **bgtm** added
  - Interrupts mask bits for Block Wait Time and Block Guard Time violation flags are added.

51.2.5.8 Transmit Threshold Register (XMT_THRESHOLD)

- **tdt[3:0]** added.
  - These bits were added to the Patriot ROM1 version of the SIM module for T=1 support. These bits are not available on the Patriot RAM1.
- Transmit threshold interrupt mask (**xtm**) moved
  - The transmit threshold error interrupt mask (**xtm**) was moved from the XMT_THRESHOLD register to the INT_MASK register.
- Transmit threshold error interrupt flag clear (**xtc**) removed
  - The transmit threshold error interrupt flag (**xte**) is now a write-one-to-clear bit in the XMT_STATUS register. The **xtc** bit is no longer needed.

51.2.5.9 Guard Control Register (GUARD_CNTL)

- Receive 11 ETU mode bit (**rcvr11**) added
  - The **rcvr11** bit is added to provide the SIM receiver the ability to receive 11 ETU transmissions.

51.2.5.10 Reset Control Register (RESET_CNTL)

- **dbug** added.
  - This bit was added to the Patriot ROM1 version of the SIM module for debug support. This bit is not available on the Patriot RAM1.
- **stop** added.
  - This bit was added to the Patriot ROM1 version of the SIM module for no-stop-clock SIM card support. This bit is not available on the Patriot RAM1.
- Reset bits **doze, kill_clk**, **soft_reset**, **flush_xmt**, and **flush_rcv** added
  - The RESET_CNTL register is a new register for the SIM module. The five bits allow doze support, various degrees of resetting a portion, or all of the SIM module, and/or disabling the SIM module clock.

51.2.5.11 Character Wait Counter Register (CHAR_WAIT)

- **char_wait_time[15:0]** added.
  - This register was added to the Patriot ROM1 version of the SIM module for T=1 support. This register is not available on the Patriot RAM1.

51.2.5.12 General Purpose Counter Register (GPCNT)

- **general_purpose_count[15:0]** added.
  - This register was added to the Patriot ROM1 version of the SIM module for T=1 support. This register is not available on the Patriot RAM1.

51.2.5.13 Divisor Register (DIVISOR)

- **divisor[6:0]** added.
  - This register was added to the Patriot ROM1 version of the SIM module for flexible baud rate support. This register is not available on the Patriot RAM1.

51.2.5.14 Receive Buffer Registers (PORTx_RCV_BUF)

- **portx_cwt** added
  - This register bit was added to store the Character Wait Time violation flag in the Receive FIFO, along with the frame error and the parity error.

51.2.5.15 Block Wait Time Registers (BWT_xSB)

A pair of registers was added to store the comparator value for the Block Waiting Time measurements, required for T=1 SIM cards.

- bwt_msb [15:0]
  - Most significant bits {31:16]
- bwt_lsb [15:0]
  - Least significant bits [15:0]

51.2.5.16 (Accurate) Character Wait Time Register (CHAR_WAIT_ACC)

- **char_wait_acc[15:0]** added
  - Due to some inaccuracies, CHAR_WAIT register was not programmed to the exact value needed. Instead, it was being programmed to a value which compensated for inaccuracies present in the character wait time counter logic. This register is added to keep the SIM software backwards compatible, and to also provide a the new, more accurate comparator register for CWT measurement.

51.2.5.17 Block Guard Time Register (BGT)

- **bgt [7:0]** added
  - Block Guard Time register was added for T=1 SIM card support.

51.3 Module Overview

The SIM Interface Module (SIM) is designed to facilitate communication to SIM cards or Eurochip pre-paid phone cards. The SIM module has two ports that can be used to interface with the various cards. The interface with the MCU is a 16 bit connection via the AIPI Controller and the IP-Bus Interface, as described in the reference documents *AIPI Controller*.

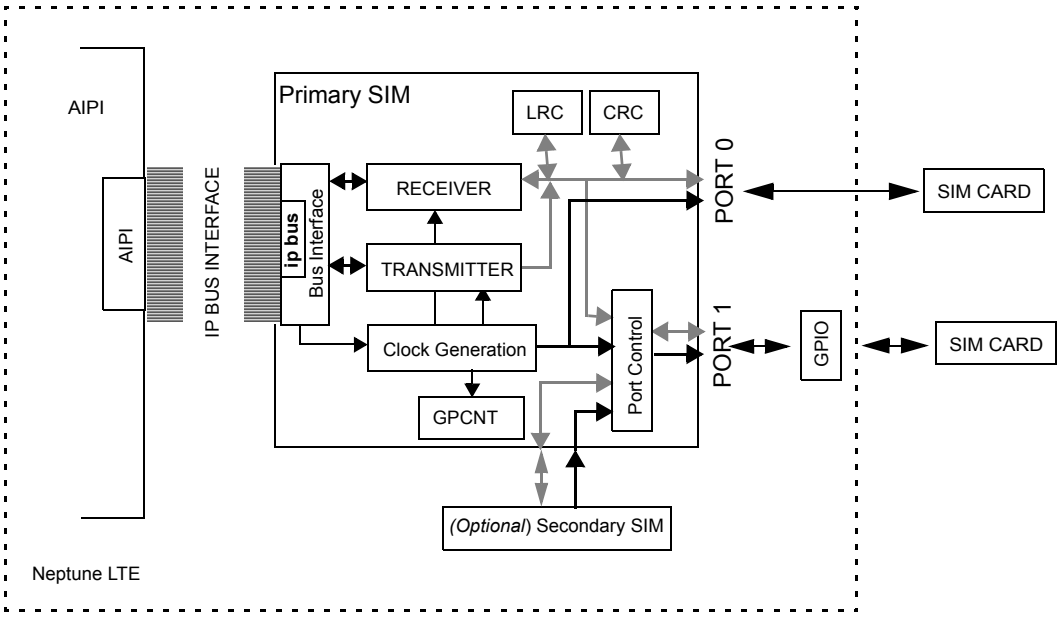


Figure 51-1. Top Level SIM Module Diagram

The SIM design is summarized in the following sections.

- Section 51.3.1 gives an overview of the SIM Bus Interface.
- Section 51.3.2 gives an overview of the SIM Clock Generator.
- Section 51.3.3 gives an overview of the SIM Transmitter.
- Section 51.3.4 gives an overview of the SIM Receiver.
- Section 51.3.5 gives an overview of the SIM Port Control block.
- Section 51.3.6 gives an overview of the SIM General Purpose Counter block.
- Section 51.3.7 gives an overview of the SIM LRC block.
- Section 51.3.8 gives an overview of the SIM CRC block.
- **Section 51.3.9** gives and overview of the SIM ETU counter block.

51.3.1 SIM Bus Interface Overview

The SIM module is designed with a separate block that encompasses the interface to the IP Bus. The bus interface is built in such a way as to be easily modified to other bus definitions. The bus interface has the following features:

- Peripheral Address/chip select decoding
- Illegal Address generation
- Dynamic wait state generation (tied inactive for Patriot)
- Register organization
- Register bit implementations

Table 51-1. SIM Top Level Interrupt Summary

Flag	Interrupt sources	Interrupt masks	Description
SIMIRQ_N	xte, oef, sdi0, sdi1, tfo, gpcnt, cwt, rte, bwt, bgt	xtm, oim, sdim0, sdim1, tfom, gpcntm, cwtm, rtm, bwtm, bgtm	SIM General Interrupt
SIMDAT_N	tc, etc, tfe, rdrf, tdtf	tcim, etcim, tfeim, rim, tdtfm	SIM Data Interrupt

51.3.2 SIM Clock Generator Overview

The SIM module contains a block designed specifically for generating the clocks used internal to the SIM module, and the clocks provided to the SIM cards. The clock generator has the following features:

- REF_CLK (CKIH) divide by 3, 4, or 5 SIM card clock generation (50% duty cycle)
- Legacy Transmitter clock generation (/16, /32, /64, /128, /256, /372)
- Legacy Receiver clock generation (16X transmitter clock; 12X in /372 mode)
- Programmable Divisor for Transmitter clock generation

There are no interrupt sources generated by the SIM clock generator block.

51.3.3 SIM Transmitter Overview

The SIM Transmitter block contains the transmit state machine, transmit shift register, and transmit FIFO. The transmitter has the following features.

- 16 deep transmit FIFO
- Automatic NACK generation on parity and overrun errors
- Hardware data format support (inverse convention or direct convention)
- Retransmission of data upon SIM card NACK request with configurable maximum threshold of retransmissions
- Programmable guard time between transmitted bytes

Table 51-2. SIM Transmitter Interrupt Summary

Flag	Flag register	Mask	Mask register	Description
tc	XMT_STATUS (page 51-61)	tcim	INT_MASK (page 51-66)	Transmit complete
etc		etcim		Early transmit complete
tfe		tfeim		Transmit FIFO empty
xte		xtm		Transmit threshold error
tfo		tfom		Transmit FIFO Overfill Error
tdtf		tdtfm		Transmit Data Threshold

51.3.4 SIM Receiver Overview

The SIM Receiver block contains the receive state machine, receive FIFO, and control logic. The receiver has the following features.

- 32 deep receive FIFO
- Decoding of initial character mode for setting data format

- Hardware data format support (inverse convention or direct convention)
- NACK detection, and configurable Receive NACK Threshold counter and interrupt
- 11 ETU character support
- Character Wait Time Counter
- Block Wait Time / Block Guard Time support.

Table 51-3. SIM Receiver Interrupt Summary

Flag	Flag register	Mask	Mask register	Description
rdrf	RCV_STATUS (page 51-63)	rim	INT_MASK (page 51-66)	Receive data register full (FIFO threshold level reached)
oef		oim		Overrun error flag
rte		rtm		Receive Threshold Error Flag

51.3.5 SIM Port Control Overview

The SIM module supports numerous port control functions necessary for SIM card interaction. Each of the two SIM card ports has the following I/O: CLK, $\overline{\text{RST}}$, DATA_RCV, DATA_XMT, SIMPD (SIM Presence Detect), and SVEN (SIM Vcc enable). The following list gives a number of the important features of the port logic.

- Open-drain or push pull DATA_XMT pin
- Port 1, port 0, or alternate Port selection
- SIM card presence detect with interrupt capability (see Table 51-4)
- Deep sleep wake-up via SIM card presence detect interrupt
- Manual control of all SIM card interface signals
- Automatic power down of port logic on SIM card presence detect
- Manual Power Down of Port logic on SIM card
- Two interrupt sources (see table below)

Table 51-4. SIM Port Control Interrupt Summary

Flag	Flag register	Mask	Mask register	Description
sdi1	PORT1_DETECT (page 51-51)	sdim1	PORT1_DETECT (page 51-51)	SIM detect interrupt for port 1
sdi0	PORT0_DETECT (page 51-70)	sdim0	PORT0_DETECT (page 51-70)	SIM detect interrupt for port 0

51.3.6 SIM General Purpose Counter Overview

The SIM module provides a 16-bit counter that can be configured to count using clock sources from the card clock, receive clock, or transmitter clock (ETU Clock). A programmable 16-bit register is provided to compare with the counter for interrupt generation.

- Selectable clock source
- 16-bit counter and comparator
- Interrupt source (see table below)

Table 51-5. SIM Port Control Interrupt Summary

Flag	Flag register	Mask	Mask register	Description
gpcnt	XMT_STAT (page 51-61)	gpcntm	INT_MASK (page 51-66)	GPCNT Comparator Interrupt

51.3.7 SIM LRC Block Overview

The SIM module contains a block designed to generate Linear Redundancy Checking (LRC) information for both received and transmitted characters. The LRC block has the following features:

- 8-bit LRC value
- Valid LRC Detector

There are no interrupt sources generated by the SIM LRC block.

51.3.8 SIM CRC Block Overview

The SIM module contains a block designed to generate Cyclic Redundancy Checking (CRC) information for both received and transmitted characters. The CRC block has the following features:

- 16-bit CRC value
- 16-bit CRC Polynomial Generator
- Valid CRC Detector

There are no interrupt sources generated by the SIM CRC block.

51.3.9 SIM ETU counter Block Overview

The SIM contains a block designated to count the number of the etu clock cycles between characters received, or transmitted in to/from the SIM card. Depending upon SIM mode of operation, the etu counter block would flag violations as defined in ISO 7816/3. The ETU counter has the following features:

- 16-bit programmable Character Wait Time register.
- 16-bit programmable Block Wait Time register.
- Interrupt sources (see table below)

Flag	Flag register	Mask	Mask register	Description
cwt	RCV_STATUS (page 51-63)	cwtm	INT_MASK (page 51-66)	Character Wait Timer flag
bwt		bwtm		Block Wait Time Violation
bgt		bgtm		Block Guard Time Violation

51.4 Module Interface

The following sub-sections give the pertinent features of the main sections of the SIM design:

- SIM IPBUS Interface (see **Example 51.6.1**)
- SIM Bus Interface (see Section 51.6.2)
- SIM Clock Generation (see Section 51.6.3)

- SIM Transmitter (see Section 51.6.4)
- SIM Receiver (see Section 51.6.5)
- SIM port control (see Section 51.6.6)
- SIM General Purpose Counter (see Section 51.6.6.4)
- SIM LRC block (see Section 51.6.8)
- SIM CRC block (see Section 51.6.9)
- SIM ETU Counter block (see **Example 51.3.9**)

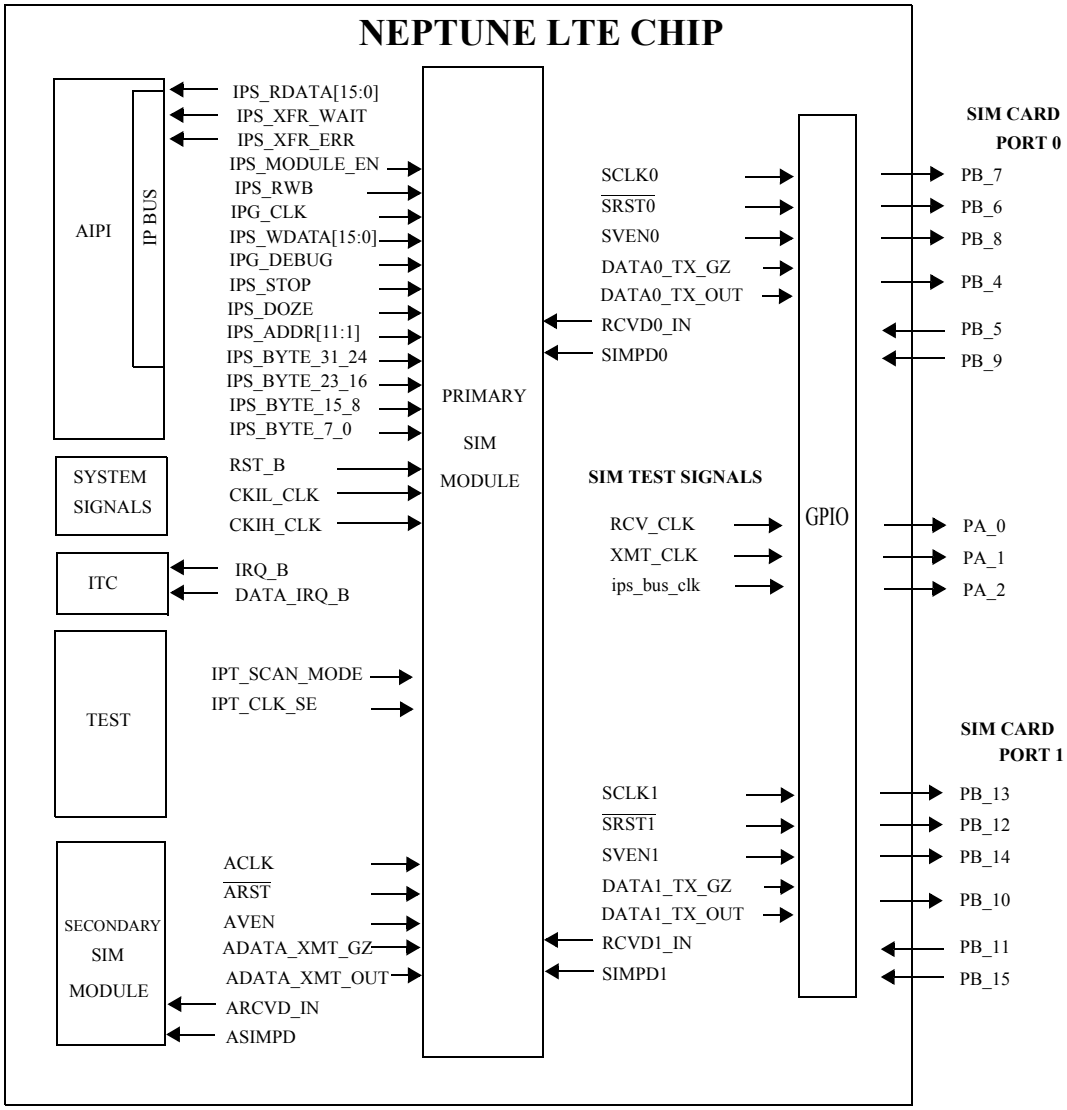


Figure 51-2. SIM Port Block Diagram

51.4.1 Port Description

Table 51-6. MCore Interrupt Controller Interface Signals

Signal	I/O	Description
IRQ_B	output	Active low SIM Interrupt.
DATA_IRQ_B	output	Active low SIM Data Interrupt.

Table 51-7. System Signals

Signal	I/O	Description
CKIH_CLK	input	Reference Frequency input. One of 13Mhz, 16.8Mhz, or 19.44Mhz.
CKIL_CLK	input	Low frequency reference clock. 32.768 Khz
IPG_CLK	input	MCore frequency used for bus clocking
RST_B	input	Active low, system reset input

Table 51-8. TEST Interface Signals

Signal	I/O	Description
IPT_SCAN_MODE	input	Scan Mode input; Configures scan muxes to allow TCK into Flops
IPT_CLK_SE	input	Scan Enable; Connects to “se” input on all Scan Flops

Table 51-9. IP BUS Interface Signals

Signal	I/O	Description
IPS_XFR_WAIT	output	Signal indicating a wait state is active. (Tied low in SIM)
IPS_XFR_ERR	output	Indicates that an invalid access has occured
IPS_RDATA[15:0]	output	IP peripheral module databus in from AIPI
IPS_MODULE_EN	input	IP peripheral module select
IPS_RWB	input	IP peripheral module read/write signal
IPS_BYTE_31_24	input	IP module byte write enable signals; signifies write access size
IPS_BYTE_23_16	input	IP module byte write enable signals; signifies write access size
IPS_BYTE_15_8	input	IP module byte write enable signals; signifies write access size
IPS_BYTE_7_0	input	IP module byte write enable signals; signifies write access size
IPG_STOP	input	IP peripheral stop mode decoded from lpmd[1:0]
IPS_ADDR[11:1]	input	IP peripheral module address bus
IPG_DEBUG	input	IP peripheral module debug event signal
IPS_WDATA[15:0]	output	IP peripheral module data out bus from module

Table 51-10. GPIO Interface Signals

Signal	I/O	Description	GPIO Pin
RCV_CLK	output	Test Signal. Receiver sampling clock.	PA_0
XMT_CLK	output	Test Signal. Transmitter bit rate clock.	PA_1
ips_bus_clk_test	output	Test Signal. Register access strobe	PA_2
SCLK0	output	SIM clock Port 0. Gated by SCEN0 and controlled by SCSP0	PB_7
SRST0	output	SIM reset line Port 0	PB_6
SVEN0	output	SIM vcc enable line Port 0	PB_8
DATA0_TX_OUT	output	SIM Transmit data line Port 0	PB_4
DATA0_TX_GZ	output	Transmit tri-state control signal Port 0	PB_4
SCLK1	output	SIM clock Port 1. Gated by SCEN1 and controlled by SCSP1.	PB_13
SRST1	output	SIM reset line Port 1	PB_12
SVEN1	output	SIM vcc enable line Port 1	PB_14
DATA1_TX_OUT	output	SIM Transmit data line Port 1	PB_10
DATA1_TX_GZ	output	Transmit tri-state control signal Port 1	PB_10
RCVD0_IN	input	Receive data line Port 0	PB_5
SIMPD0	input	SIM presence detect line Port 0	PB_9
RCVD1_IN	input	Receive data line Port 1	PB_11
SIMPD1	input	SIM presence detect line Port 1	PB_15

Table 51-11. Secondary SIM Module Interface Signals

Signal	I/O	Description
ARCVD_IN	output	Alternate Receive data line. Output to secondary SIM module
ASIMPD	output	Alternate SIM presence detect line. Output to secondary SIM module
ACLK	input	Alternate SIM clock. Input from secondary SIM module
ARST	input	Alternate SIM reset line. Input from secondary SIM module
ADATA_TX_OUT	input	Alternate transmit data line. Input from secondary SIM module
ADATA_TX_GZ	input	Alternate transmit tri-state control signal. Input from secondary SIM module
AVEN	input	Alternate SIM vcc enable line. Input from secondary SIM module

51.5 Module Architecture

To best describe the organization of the SIM module from a user's point of view, it is instructive to view the SIM at a number of different levels of hierarchy. The port diagram of Figure 51-2 on page 51-18 gives a detailed description of the connectivity of the SIM to the Neptune LTE IC. Figure 51-3 below illustrates how the SIM is organized, and how the signals can be brought out onto the two available serial ports.

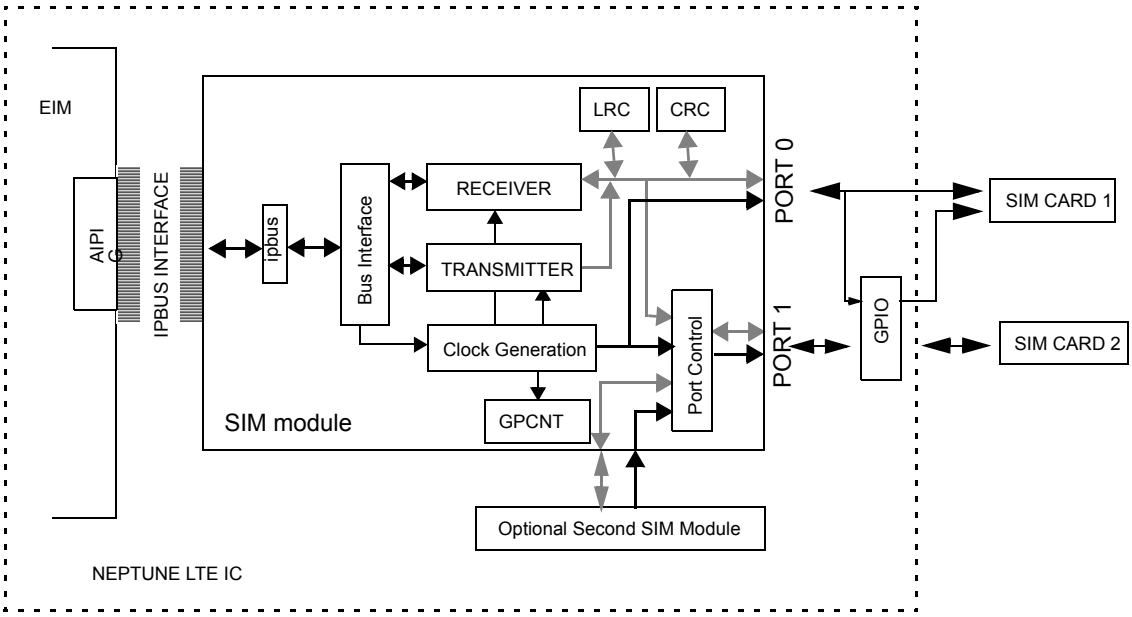


Figure 51-3. SIM Block Diagram

The SIM module is essentially a standard UART with some special provisions made for SIM card communication. The SIM consists of nine main parts: ipbus interface, bus interface, clock generator, transmitter, receiver, port controller, general purpose counter, and LRC and CRC blocks. The details of each of these sub-functions is discussed in the functional descriptions of Section 51.6 starting on page 51-23.

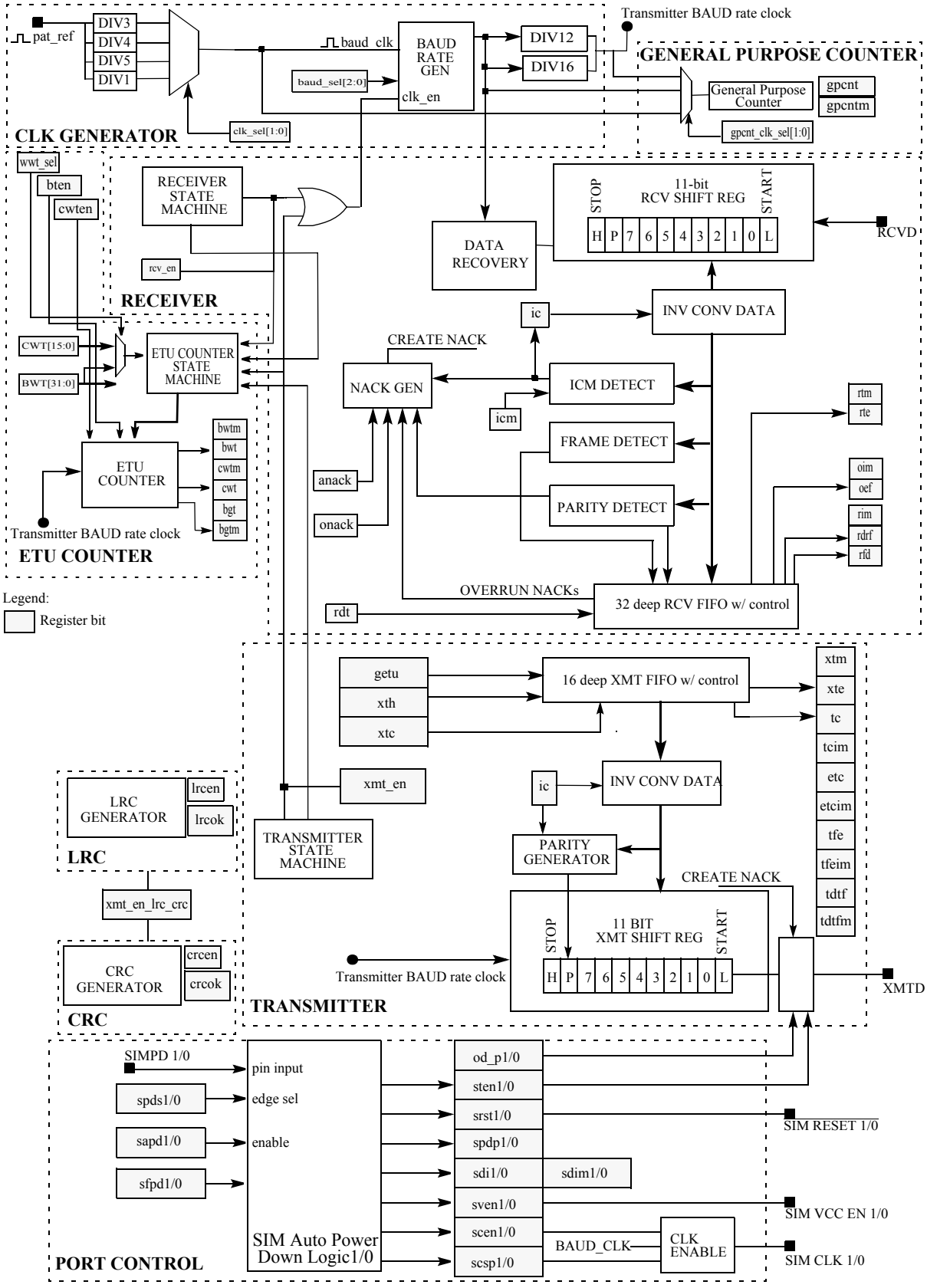


Figure 51-4. SIM Block Diagram

51.6 Functional Description

The SIM module can be used to interface to SIM cards located on either port 1 or port 0. The following subsections describe the functionality of the SIM module.

51.6.1 IPBUS Gasket ()

The ipbus is instantiated at the top-level of the SIM module. The main goal of the ipbus is to provide an IP compliant interface between the module and the AIPI controller. The ipbus interface essentially gates the *IPG_CLK* and the input and output databus to zero when the module is not selected. It also detects a non 16 bit address access to the SIM.

For more information on the ipbus interface, refer to the ***aipi specification***.

51.6.2 SIM Bus Interface

The SIM Bus interface block has been designed to enable the SIM module to be easily ported to other MCU cores. The bus interface block contains all of the logic that uses the bus interface signals. The bus interface block architecture is shown in Figure 51-5.

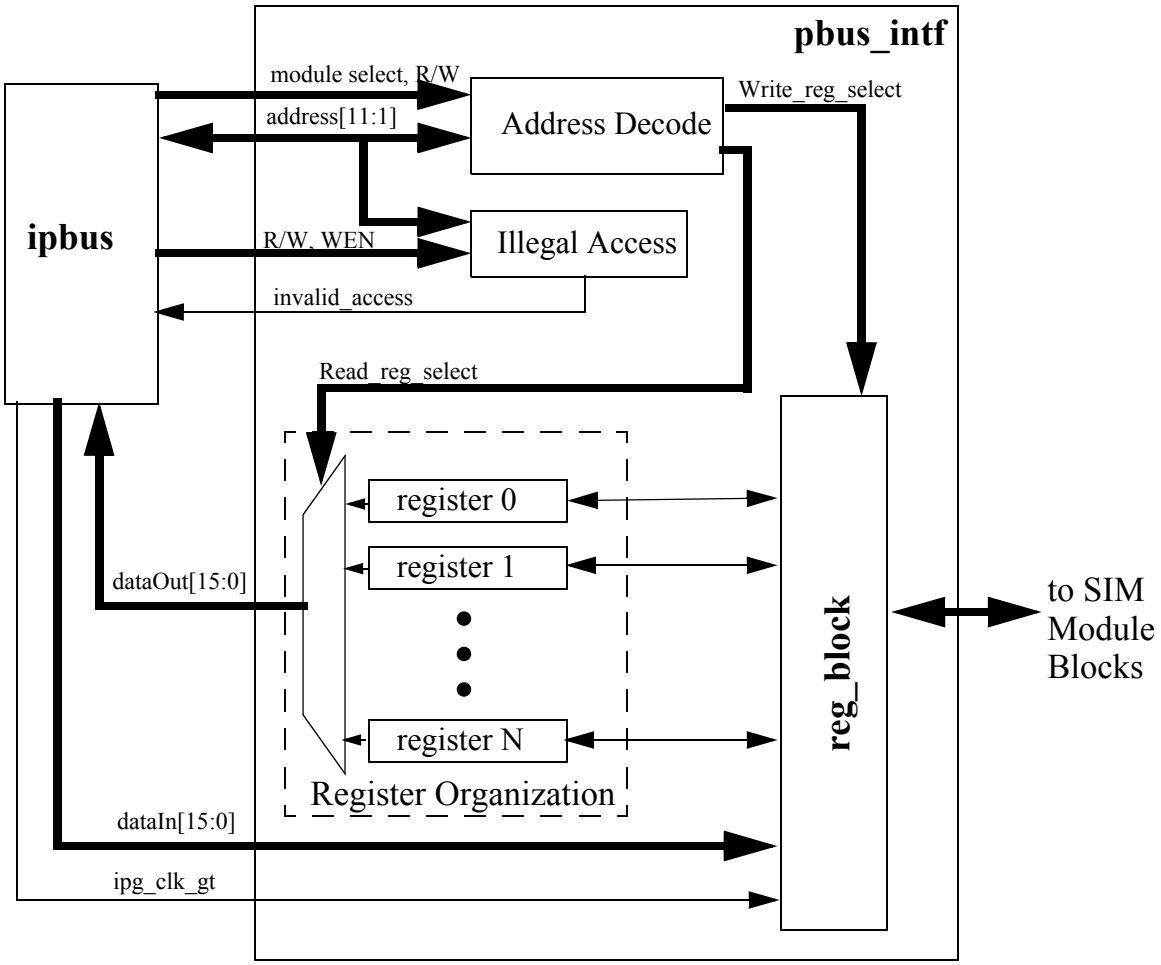


Figure 51-5. SIM Bus Interface Port Diagram

The bus interface block is responsible for peripheral bus address decoding, illegal access detection, dynamic wait-state requests, register organization, and register bit implementations. The address decoding uses PIG module address bus input along with the module select output of the ipbus to decode if the SIM module is being addressed. The output of the address decode is used with the read/write signal from the PIG to determine the action requested by the bus master. If the read signal is active, the data_out bus will be driven with the register selected by the address decoding. If the write signal is active, the data_in bus will be latched into the register selected by the address decoder at the rising edge of the data_strobe signal.

The illegal access detection is implemented similarly to the address decoding. If the ips_module_en signal becomes active while the address inputs are pointing to an unimplemented address, the ips_xfr_err signal is asserted asynchronously to the bus master.

The ips_xfr_err signal can also be generated under certain register access conditions such as writing to a read-only register. This signal is connected to the **ipbus** which asserts **ips_xfr_err** to the aipi for generating and access exception to the core.

The dynamic wait state generation is not implemented in the current version of the SIM module. However, since the SIM operates in a different clock domain than the MCore, the number of cycles required for the MCore to access a SIM register can vary from 4 to 6 cycles.

The register organization provides the mapping of the register bits to a given register address in the SIM module register address space. All of the register bits are brought into the register organization section so that they can be added to any register location in any order. This eases re-use of the SIM module when porting to a different MCU interface.

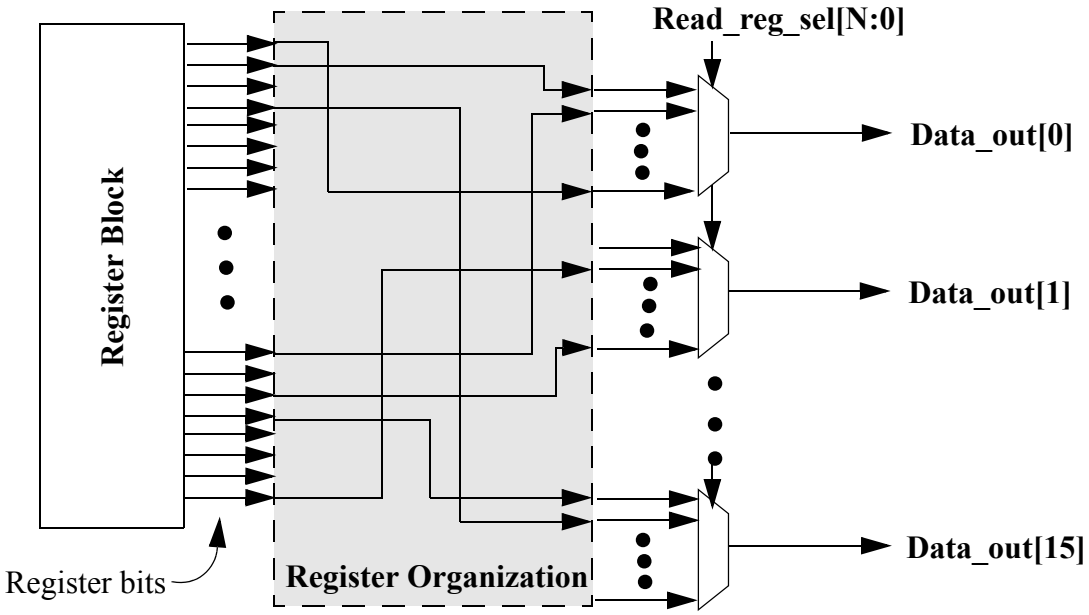


Figure 51-6. Register Organization Diagram

The register bit implementations are done inside the reg_block sub-module. This block accepts the decoded read and write signals for the register bits generated in the address decoder. The ipg_clk_gt output of the ipbus is used as the clock input to the register bits in order to clock in the new data during a write access. The write_reg_sel signals are used to gate the write action with the address decoding. The clearing mechanisms for status bits are implemented as write-one-to-clear.

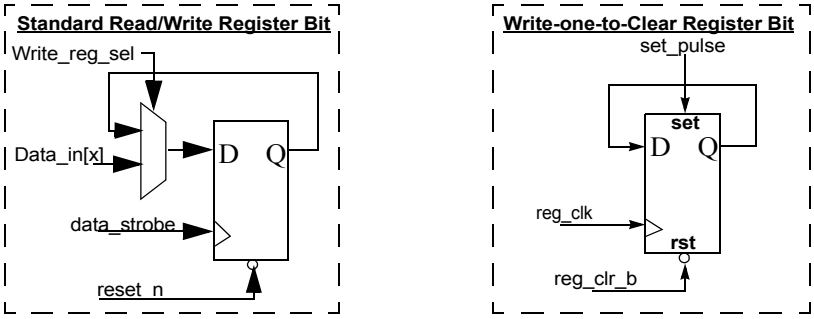


Figure 51-7. Register Bit Implementations Diagram

In the “write-one-to-clear” register bit example, “set_pulse” represents a pulsed signal generated in the data_strobe clock domain to set the register output.

51.6.3 SIM Clock Generator

The clock generator is responsible for implementing clock tree synthesis constructs, scan clock muxing, baud rate clock (baud_clk) generation, and providing clocks to the transmitter, receiver, and port controller sections of the SIM module. Figure 51-8 shows an architecture diagram of the Clock generator block. The dividers outlined in bold generate a pulsed clock of the desired frequency. The pulse is equal in duration to one half the CKIH (ref_clk) period.

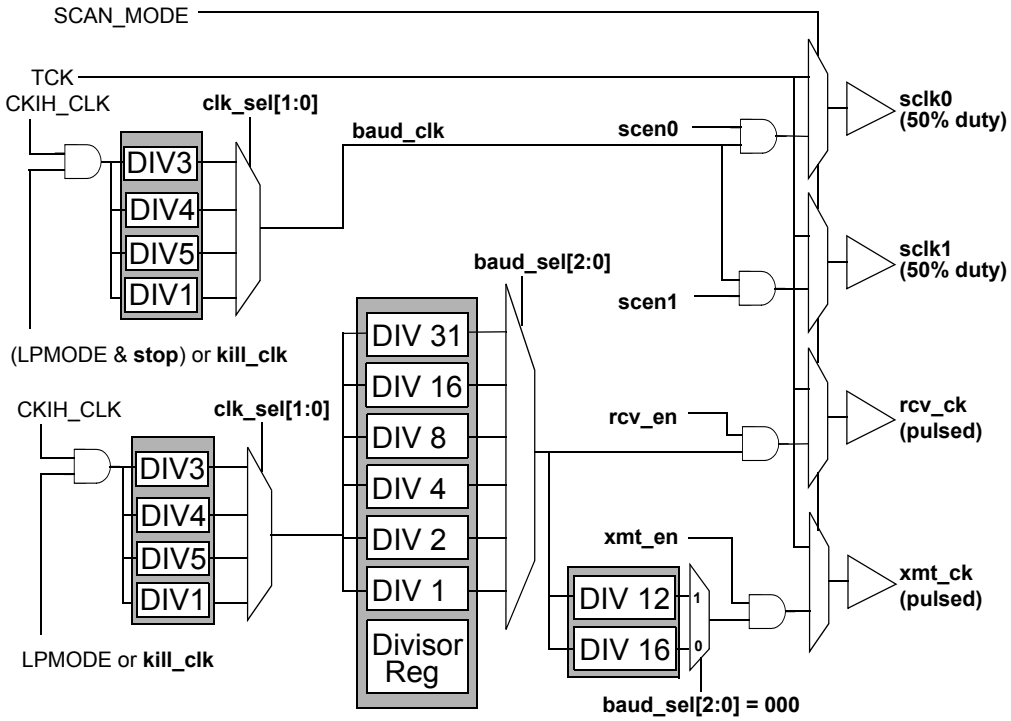


Figure 51-8. SIM Clock Generator Diagram

51.6.3.1 Clock Tree Synthesis

The clock tree synthesis constructs used in the current implementation follow the Neptune LTE clock tree synthesis methodology. This cell can then be used to control the clock skew during the layout process.

51.6.3.2 SCAN Test

The scan clock muxing is implemented according to the design for testability rules described in the *Neptune LTS Design for Test methodology*. These rules specify that clock muxing is done at the CCM level for all clocks, module asynchronous reset is used as scan_reset, and scan_mode to is used to explicitly mux-out any internally generated clocks or Flop resets during scan testing.

51.6.3.3 Baud Clock Generation

The baud rate clock generation performed by the clock generator results in one of four different frequencies. The default frequency is a divide by four of the CKIH_CLK input. The baud rate can be programmed to be CKIH_CLK divided by 1, 3, 4, or 5 using the **clk_sel[1:0]** bits as shown in Table 51-20 on page 51-56. The baud rate clock is generated in two forms in the design. The baud_clk that is used by the SIM cards (sclk0, sclk1) is generated so that it is approximately 50% duty at all divide values. This is an internally generated clock that is used to go out of the SIM. This is necessary to meet the requirements of the ISO 7816 specification. The baud_clk that is used internal to the SIM module is generated as a gated version of the CKIH_CLK clock. The resultant clock will be a pulse of one-half CKIH_CLK period in width with the expected frequency.

51.6.3.4 Transmitter Clock Generation

The transmitter clock (xmt_clk) is generated by the clock generator based on the values passed to it for the baud rate select (**baud_sel[2:0]**). The transmit clock is gated by the transmit enable (**xmt_en**) register bit. When the transmitter is enabled, the clock generator counts the appropriate number of receive clock (rcv_clk) positive edges to determine when to toggle the transmitter clock output. The transmitter clock is always based upon the receive clock. The ratio between transmit clock period and receive clock period is typically 16:1, except for the slowest baud rate (/372 mode, **baud_sel[2:0]** set to 000) where the ratio is changed to 12:1.

51.6.3.5 Receiver Clock Generation

The receiver clock (rcv_clk) is generated by the clock generator based on the value passed to it for the baud rate select (**baud_sel[2:0]**). The receiver clock is gated by the receiver enable (**rcv_en**) register bit. When the receiver is enabled, the clock generator counts the appropriate number of baud_clk positive edges to determine when to toggle the receiver clock output. The number of baud_clk positive edges is determined by the value of the baud rate select input (**baud_sel[2:0]**) and is programmable to these divisors: 372 (slowest), 256, 128, 64, 32, 16, and **divisor[6:0]**. The programmable divisor (**divisor[6:0]**) is configurable via the DIVISOR_REG register. There are 16 receive clock periods during each of the divisor settings except for the slowest (/372) in which there are 12 receive clock cycles.

51.6.3.6 Port Control Clock Generation

The port controller clocks are provided by the clock generator for use on the SIM card ports. These clocks are equivalent in frequency to the baud_clk and are gated by the SIM clock enable (**scen0** and **scen1**) signals. The level at which the card clocks (sclk0, sclk1) are stopped when disabled is determined by the SIM clock select polarity (**scsp0**, **scsp1**) inputs to the clock generator. Synchronizers are implemented to ensure glitch free operation of the card clocks when enabling/disabling, or changing the clock stopped polarity.

51.6.3.7 Low Power Mode Clock Control

The clock generator block is responsible for gating the clocks to the SIM module appropriately whenever a low power mode instruction is decoded. The AIPI interface provides three signals that encode the three available low-power states of the Processor. These states are: STOP, WAIT, and DOZE. The STOP instruction is supported through the use of the STOP bit in the RESET_CNTL register. If the STOP bit in the RESET_CNTL register is clear, the clock generator will halt all clocks in the SIM module. If the STOP bit is set, the clock generator will halt all clocks except for the baud_clk that is used to generate the SIM card clocks (sclk0, sclk1). For WAIT instructions, the clock generator block does not affect the clocking of the SIM module. The DOZE instruction is supported through the use of the DOZE bit in the RESET_CNTL register. If the DOZE bit in the RESET_CNTL register is set, the clock generator will shut down all module clocks after the current transmission is complete. If the SIM module is not transmitting when the DOZE instruction is executed, the SIM module clocks will be shut down immediately. Otherwise, the DOZE instruction has no effect (i.e. just like WAIT).

51.6.4 SIM Transmitter

The SIM Transmitter block has been designed to localize all transmit related circuitry into one section of hierarchy. The Transmitter block is comprised of the following sections of logic: transmit state machine, transmit shift register, transmit FIFO, guard time generator, transmit NACK control, and transmit data convention.

51.6.4.1 Transmit State Machine

The Transmit state machine is the heart of the transmitter block. The state machine is responsible for sequencing through a transmit operation while reacting to inputs from the receiver, the transmit FIFO, and the guard time circuit. Figure 51-9 below depicts the transmit state machine operation. The functions performed by each state are:

- IDLE
  - This is the initial state. The state machine waits here until it detects **xmt_en** is set and a write to the transmit FIFO has occurred. The data pointed to by the transmit read pointer is loaded into the shift register, and the state machine transitions to the MAIN_XMIT state. Any time **xmt_en** is cleared, the state machine will return to this state.
- MAIN_XMIT
  - The transmitter is operating normally in this state. The data in the shift register is shifting once every transmit clock cycle. When the second to last bit of the current transmission is about to be sent, the state machine transitions to the LAST_XMIT state.
- LAST_XMIT
  - This state transmits the last bit of the current transmission and determines the next operation. One of the following will occur.
    - If **getu[7:0]** is non-zero, jump to the GUARD_WAIT state.
    - If a transmit NACK error occurred, with a zero in **getu[7:0]**, jump to MAIN_XMIT state to retransmit the current byte.
    - If no transmit NACK, and **getu[7:0]** is zero, load the shift register, jump to MAIN_XMIT to transmit the next byte
    - If no transmit NACK, **getu[7:0]** is zero, and the FIFO is empty, jump to IDLE state; set the transmit complete (**tc**) flag.
- GUARD_WAIT
  - The state machine remains in this state until the Guard time counter has expired.
    - If a transmit NACK error occurred on last transmission, jump to RTX_MAIN_XMIT and re-transmit
    - If no transmit NACK and the FIFO is not empty, load the shift register, jump to MAIN_XMIT to transmit the next byte.
    - If no transmit NACK and the FIFO is empty, return to the IDLE state.
    - If transmit NACK threshold is detected, stop transmitter, set **xte** flag, and jump to the IDLE state.
- RTX_MAIN_XMIT
  - The transmitter is operating normally in this state. The data in the shift register is shifting once every transmit clock cycle. When the second to last bit of the current transmission is about to be sent, the state machine transitions to the RTX_LAST_XMIT state. This state is identical to the MAIN_XMIT state except that it retransmits the previously NACKed byte.
- RTX_LAST_XMIT
  - This state transmits the last bit of the current re-transmission and determines the next operation.

One of the following will occur.

— If **getu[7:0]** is non-zero, jump to the GUARD_WAIT state.

— If a transmit NACK error occurred, with a zero in **getu[7:0]**, jump to GUARD_WAIT state to check Transmit NACK threshold.

— If no transmit NACK, **getu[7:0]** is zero, and the FIFO is not empty, load the shift register, jump to MAIN_XMIT to transmit the next byte

— If no transmit NACK, **getu[7:0]** is zero, and the FIFO is empty, jump to IDLE state; set the transmit complete (**tc**) flag.

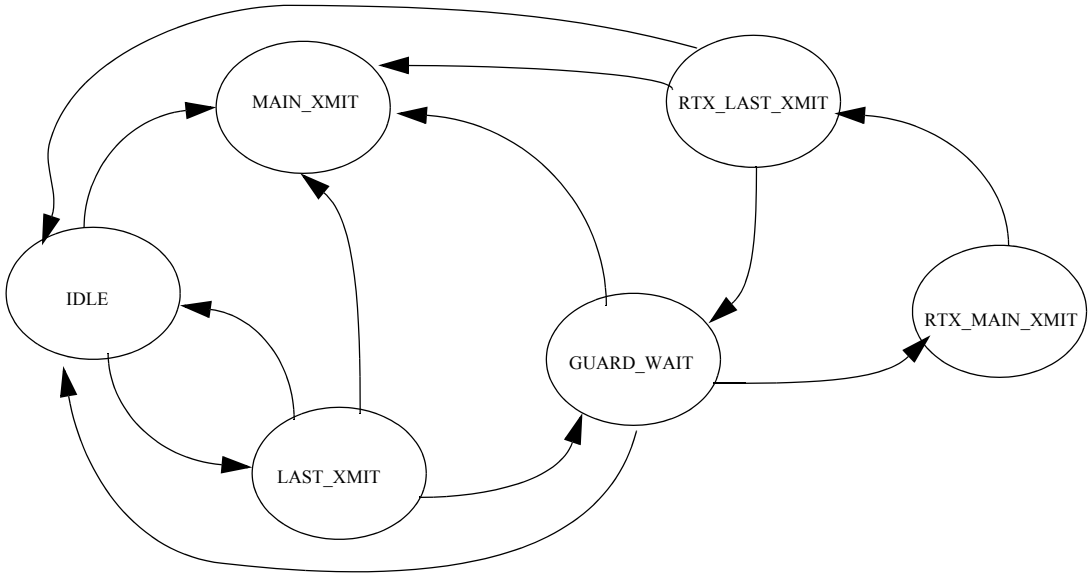


Figure 51-9. Transmit State Machine Diagram

51.6.4.2 Transmit Shift Register

The transmit shift register is 11 bits wide and controlled by the transmit state machine described previously. The shift register shifts out data at the transmit clock frequency (xmt_ck).

51.6.4.3 Transmit FIFO

The transmit FIFO is implemented inside the transmitter block. The FIFO depth is 16 bytes for Patriot. The FIFO block is shared by both SIM module ports. The transmit FIFO cannot be accessed by the alternate SIM module through the alternate port. Each write to the transmit FIFO increments the transmit FIFO write pointer. Each time the transmit shift register is loaded from the transmit FIFO, the transmit FIFO read pointer is incremented. When the read and write pointers are equal, the transmit FIFO empty flag (**tfe**) will be set. Software has no visibility of the transmit FIFO pointers, but a transmit FIFO threshold value can be set to alert the software when the number of bytes in the FIFO has reached a specified level. A read of the transmit FIFO register (PORTx_XMT_BUF) will return the last byte written to the FIFO. Writes to the transmit FIFO can occur at any time.

The transmit FIFO can be flushed by setting the **xmt_flush** bit in the RESET_CNTL register. A transmit NACK threshold error (**xte**) will halt the transmitter, and flush the transmit FIFO. The flush operation resets the transmit read and write pointers to equal values. Everything in the transmitter block is reset by the transmit flush operation. This does not include the control registers associated with the transmitter. The Transmit data threshold flag (tdtf) does not get cleared by the xmt_flush operation.

51.6.4.4 Transmit Guard Time Generator

The guard time generator is simply a counter that is clocked by the transmit clock in order to delay the beginning of the next transmission and the setting of the transmit complete interrupt flag (**tc**) by a programmable amount of transmit bit widths (ETU's). The duration of the count is controlled by the GUARD_CNTL register (**getu[7:0]**). Figure 51-10 depicts three transmit operations in order to show the effect of the guard time generator logic.

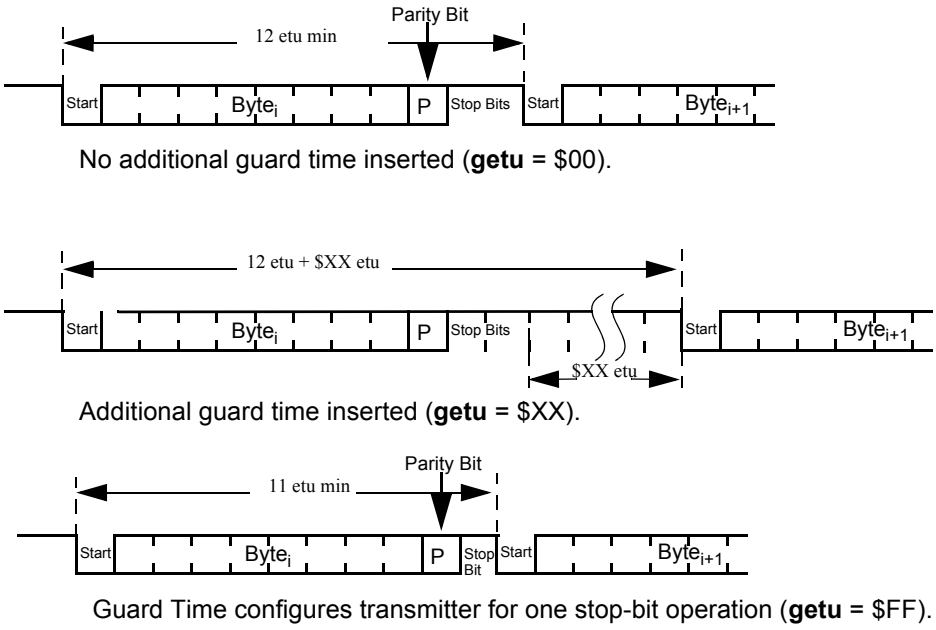


Figure 51-10. Transmit Guard Time Diagram

51.6.4.5 Transmit NACK Generator

The transmit NACK generator is responsible for driving the transmitter output low during the STOP bit time to signify an error was detected in the received data from the SIM card. This logic responds to a NACK request generated by the receiver block. Figure 51-11 shows a typical SIM transaction with the NACK pulse inserted.

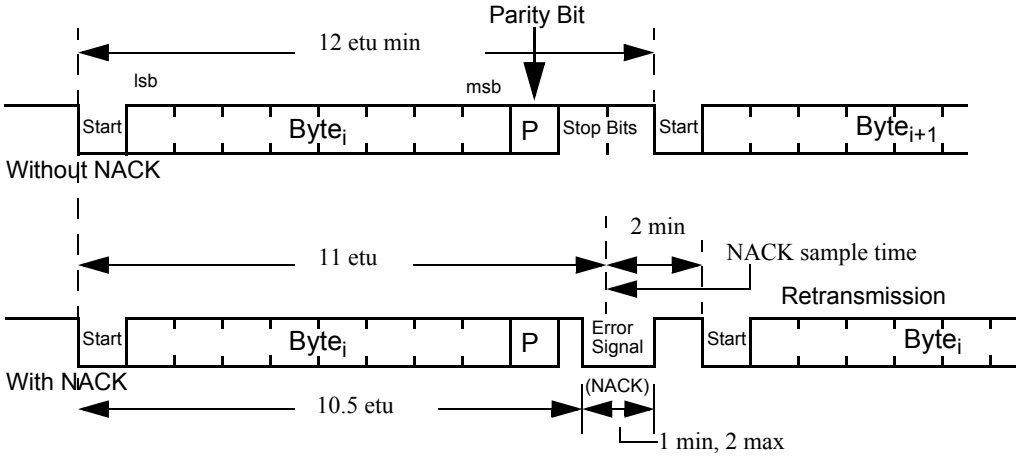
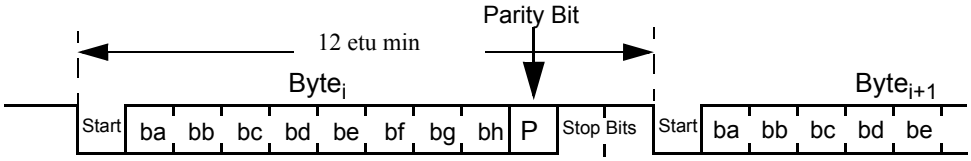


Figure 51-11. Transmit NACK Operation

The transmit NACK generator is also responsible for keeping track of the number of NACKs received during a transmit operation. The SIM receive state machine detects NACKs generated by the SIM card, and reports them to the transmit NACK logic. Once the number of detected NACKs has reached the programmed threshold, an interrupt flag is generated, the transmit FIFO is flushed, and the transmitter is disabled. Table 51-30 on page 51-72 describes the NACK threshold operation based on the programmed value of the **xth[3:0]** bits in the XMT_THRESHOLD register.

51.6.4.6 Transmit Data Convention Logic

The transmit data convention logic provides the support for the two different data conventions available in SIM cards. The data conventions are depicted in Figure 51-12.



Parity Bit: if configured for even parity, total number of logic ones in the 9 bits (8 bits data, 1 parity bit) will be even
if configured for odd parity, total number of logic ones in the 9 bits (8 bits data, 1 parity bit) will be odd
When configured for inverse convention, the parity bit is inverted by SIM before being transmitted.

Direct Convention: *ba* is lsb of data byte to be sent. *bh* is msb.
Neither the data bits nor parity bit is logically inverted.

Inverse Convention: *ba* is msb of data byte to be sent. *bh* is lsb.
Both the data bits and parity bit are logically inverted by hardware.

Figure 51-12. SIM Data Conventions

The direct data convention is the default. If the inverse convention bit (**ic**) in the DATA_FORMAT register is set, then the transmit data convention logic will convert the output of the transmit FIFO to the inverse convention before sending it to the transmit shift register.

51.6.5 SIM Receiver

The SIM Receiver block has been designed to localize all receive related circuitry into one section of hierarchy. The receiver block is comprised of the following functions: receive state machine and the receive FIFO.

51.6.5.1 Receive State Machine

The receive state machine is responsible for sampling the receive data pin and capturing the bit value into the receive shift register. Additionally, the receive state machine can detect the start bit, parity errors, framing errors, and initial character when operating in initial character mode.

Once enabled by the **rcv_en** bit in the ENABLE register, the receive state machine sequences through the states as shown in Figure 51-13. The states identified with a “16x” are used when operating in a 16X oversampling mode. The states identified with a “12x” are used when operating in a 12X oversampling mode. This mode is only used when **baud_sel[2:0]** is set to “000”. The number following the oversampling mode identifier represents the state number in the current mode. There are 12 states in “12x” mode, and 16 states in “16x” mode. Some states simply implement a one rcv_ck delay. States that perform additional functions are identified in the following bullets.

- RCV16x_1, RCV12x_1
  - This is the initial state of the receive state machine. If the first bit has not been received, the state machine remains in this state until a valid start bit is detected. For every subsequent bit, this state is simply a one rck_ck cycle delay.
- RCV16x_3, RCV12x_2
  - This state captures the first sample of the current receive data input.
- RCV16x_7, RCV12x_4
  - This state captures the second sample of the current receive data input.
- RCV16x_8, RCV12x_5
  - This state captures the third sample of the current receive data input.
- RCV16x_9, RCV12x_6
  - This state captures the fourth sample of the current receive data input. If the current bit is the 11th bit of the transfer, this state checks the for valid parity and valid initial character when appropriate. If an error is detected, a NACK is generated in this state when enabled.
  - When the transmitter is active, the sample captured during this state is the first sample of the NACK window. This is 0.5 ETU into the 11th bit of the transfer.
- RCV16x_10, RCV12x_7
  - If the current bit is the first bit of a transfer, this state will check the validity of the previous four samples and perform a majority vote on whether to accept the data as a valid start bit. If the start bit is valid, the data is shifted into the receive shift register.
  - If this is not the first bit of a transfer, this state does a majority vote on the previous four samples and places the result in the receive shift register.
- RCV16x_11, RCV12x_8
  - If the current bit is the last bit of the transfer (first stop bit), this state will signal the shift register to place its contents in the receive FIFO. Parity and framing errors are also detected in this state.
- RCV16x_13, RCV12x_9
  - If the current bit is the last bit of the transfer (first stop bit), this state takes the second sample

of the NACK window.

- RCV16x_14, RCV12x_10
  - If the current bit is the last bit of the transfer (first stop bit), this state takes the third sample of the NACK window.
- RCV16x_15, RCV12x_11
  - If the current bit is the last bit of the transfer (first stop bit), this state takes the fourth sample of the NACK window.
- RCV16x_16, RCV12x_12
  - This state represents the end of the current receive input bit. Several operations occur during this state, including:
    - Increment bit counter
    - Perform a majority vote on the NACK samples and notify the transmitter if a NACK pulse was detected.

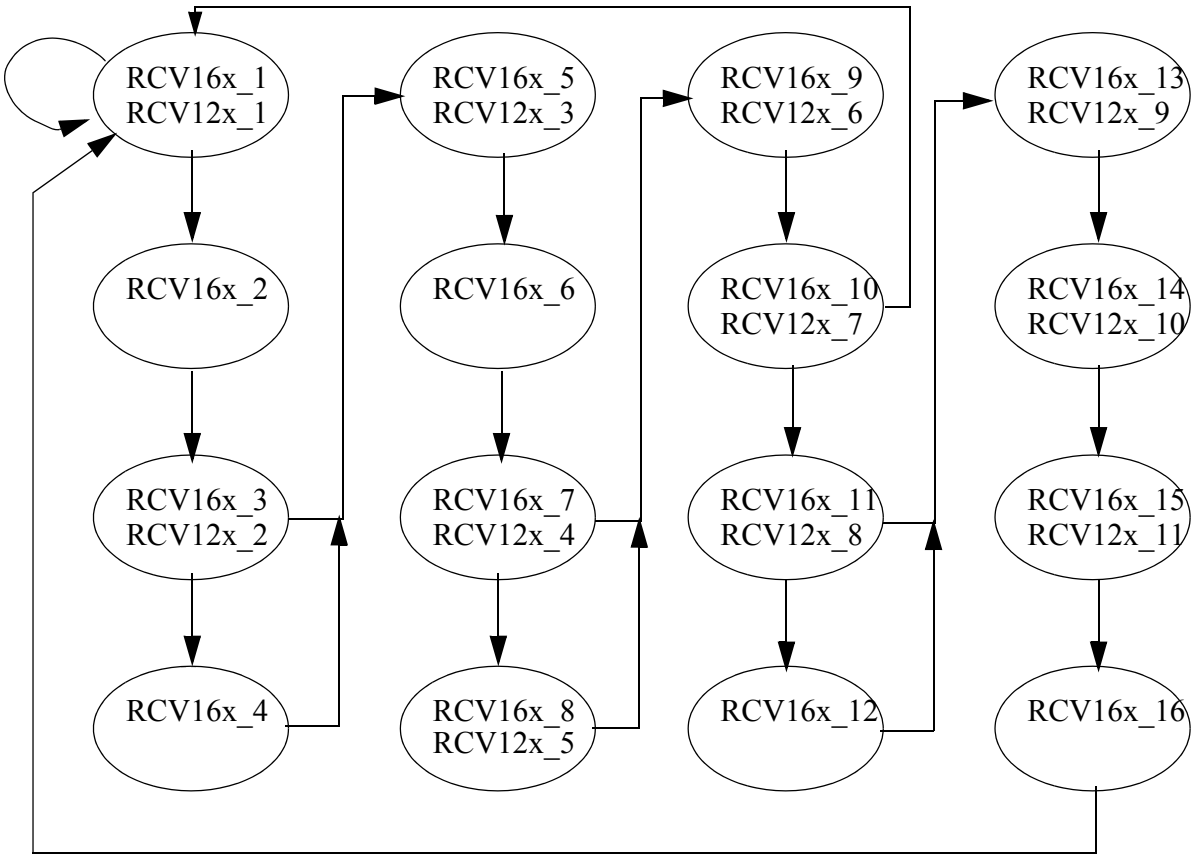


Figure 51-13. Receive State Machine Diagram

51.6.5.1.1 Data Sampling/Voting

The receive state machine runs at the receive clock rate (rcv_ck). This clock is used to oversample the received data at either a 16X or 12X sample rate. For each input bit, the receive state machine captures 3 samples. A majority voting algorithm is then applied to determine the value of the bit received. The value common to 2 or more of the samples is determined to be the bit value in the receive shift register.

51.6.5.1.2 Start Bit Detection

The SIM receive input is defined to be high when not active. The data transmission is defined to begin with a low pulse for a bit duration. This is called the start bit. The receive state machine is responsible for detecting and validating the start bit. The receive state machine samples the start bit three times using a majority voting scheme to determine if the start bit is valid. This will effectively filter out any low receive inputs shorter than one rcv_ck period. Figure 51-14 shows a typical SIM data transaction with the start bit identified.

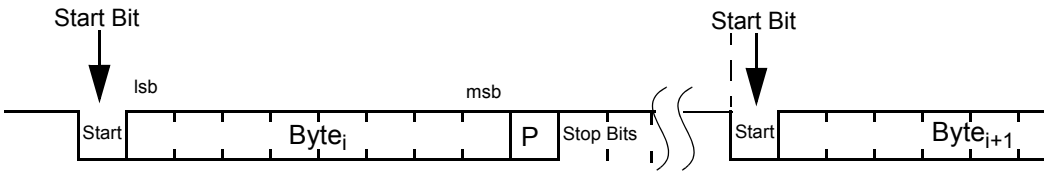


Figure 51-14. Start Bit Diagram

51.6.5.1.3 Parity Error Detection

The receive state machine is responsible for detecting parity errors in the received data. Data is always transmitted with even parity, except when in inverse convention mode. In inverse convention mode, all data bits and the parity bit are complemented making the data appear to be odd parity. The parity bit is defined as the 10th bit of the received data. The parity of the 2nd through 10th received bits is calculated by the receiver parity logic. This logic determines if the parity of the 9 received bits is correct.
Figure 51-15 shows a typical SIM data transaction with the parity bit identified.

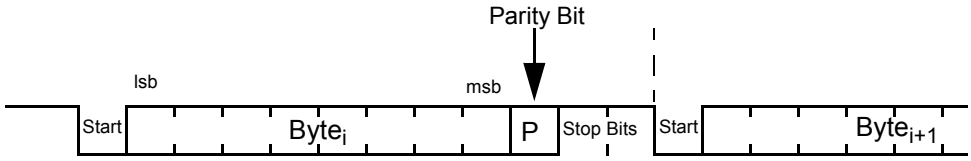


Figure 51-15. Parity Bit Diagram

When a parity error is detected on a given byte, the **rcv_pf** bit for that byte is set in the receive FIFO. A parity error cannot cause an interrupt, but it can signal the SIM transmitter to create a NACK pulse to the SIM card asking for a retransmission of the corrupted data. NACK generation upon a parity error is enabled by setting the **anack** bit in the CNTL register.

51.6.5.1.4 Framing Error Detection

The receive state machine is responsible for detecting framing errors in the received data. A SIM data transaction is defined as 11 or 12 bits long consisting of the start bit, 8 data bits, 1 parity bit and 1 or 2 stop bits. A framing error occurs when the stop bit is not detected during the 11th bit time of a data transaction.

The stop bit is generally defined as two bit times (ETU's) of a high pulse following the parity bit. When the GUARD_CNTL register is programed to $FF, the stop bit is defined as one bit time. A framing error can only occur when the parity bit of the current byte is low, and the STOP bit arrives late. Figure 51-16 shows a typical SIM data transaction with the stop bits identified. Also shown is a SIM data transaction with a late arriving STOP bit indicating a framing error.

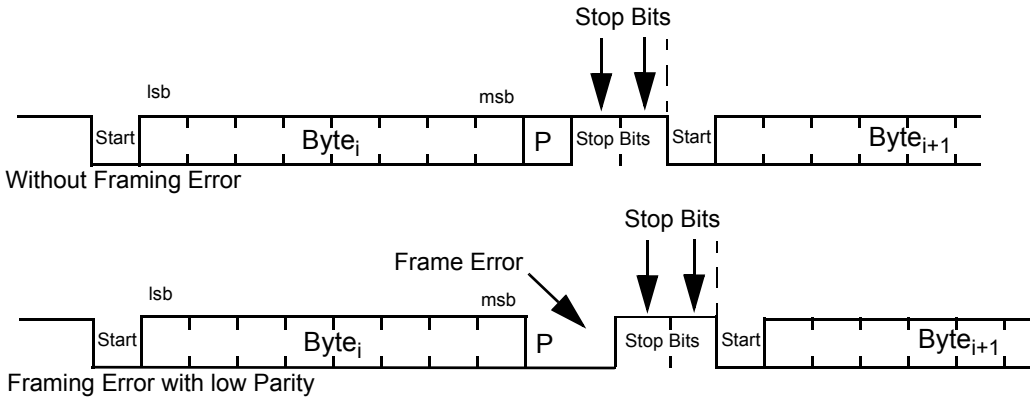


Figure 51-16. Framing Error Diagram

When a framing error is detected on a given byte, the **rcv_fe** bit for that byte is set in the receive FIFO. A framing error cannot cause an interrupt, nor can it create a NACK pulse to the SIM card asking for a retransmission of the corrupted data.

51.6.5.1.5 NACK Detection

The existence of the NACK pulse is sampled by the receive state machine at 11 elementary time units (ETUs) after the falling edge of the start bit. An ETU is equivalent in time to 1 transmit clock period. Once the receiver detects a NACK, it signals the transmitter that an error occurred. The transmitter will not initiate retransmission for at least another two ETU times as required by the ISO 7816 specification.

51.6.5.1.6 Initial Character Detection

The SIM receive state machine supports the detection of special characters that allow it to determine what data format is being used by the connected SIM card. When placed in initial character mode, the SIM expects to receive one of two potential characters that it will use to set the data format control bit, **ic**, in the DATA_FORMAT register. Valid initial characters are shown in Figure 51-17.

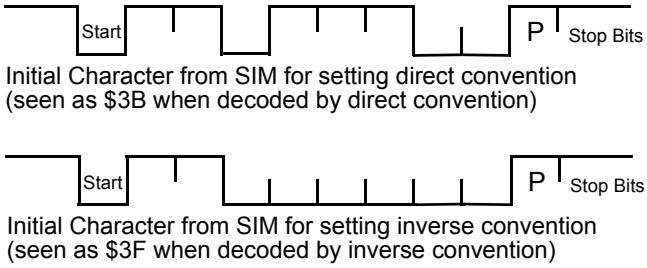
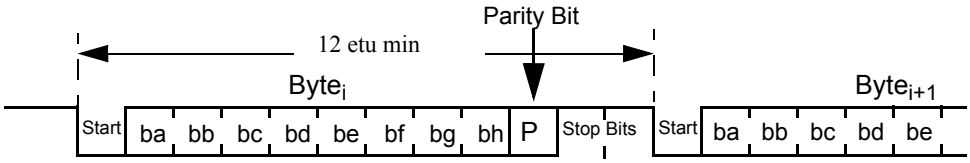


Figure 51-17. Valid Initial Characters

The two possible data formats are inverse convention and direct convention. Figure 51-18 illustrates the differences between the two formats. Essentially, inverse convention differs from direct convention in that the order of the data is flipped msb for lsb, and the data bits and parity bit are logically inverted. When receiving inverse convention data, the transformation of the data back to direct convention format is done in hardware, including the inversion of the data and parity bits.



Parity Bit: if configured for even parity, total number of logic ones in the 9 bits (8 bits data, 1 parity bit) will be even
if configured for odd parity, total number of logic ones in the 9 bits (8 bits data, 1 parity bit) will be odd
When configured for inverse convention, the parity bit is inverted by SIM card before being transmitted.

Direct Convention: *ba* is lsb of data byte to be sent. *bh* is msb.
Neither the data bits nor parity bit is logically inverted.

Inverse Convention: *ba* is msb of data byte to be sent. *bh* is lsb.
Both the data bits and parity bit are logically inverted by SIM card.

Figure 51-18. Inverse Convention vs. Direct Convention

51.6.5.2 Receive FIFO

The receive FIFO is implemented inside a sub-block of the receiver block. The FIFO depth is 32 bytes for Patriot. The FIFO block is shared by both SIM module ports. The receive FIFO cannot be accessed by another SIM module through the alternate port. The receive FIFO is accessed through the PORTx_RCV_BUF registers.

The receive FIFO is loaded from the receive shift register after the final bit of the current SIM card transmission has been received. The FIFO contains 10 bits per transmission. The lower eight bits contain the received data byte. Bits 8 and 9 contain the parity and framing status for the received byte.

Each read from the receive FIFO increments the receive FIFO read pointer. Each time the receive shift register is transferred to the receive FIFO, the receive FIFO write pointer is incremented. When the difference between the read and write pointers equals the programmed threshold value (**rdt[4:0]**), the receive data register full flag (**rdrf**) will be set. An interrupt can be generated by the **rdrf** flag if the **rim** bit in the INT_MASK register is cleared. Software has no visibility of the receive FIFO pointers. A write to the receive FIFO register (PORTx_XMT_BUF) will generate an invalid access exception to the MCore.

The receive FIFO can be flushed by setting the **rcv_flush** bit in the RESET_CNTL register. The flush operation resets the receive read and write pointers to equal values. All logic associated with the receiver will be reset by the flush operation except for receiver control registers.

51.6.5.2.1 Overrun Detection

The receive FIFO logic is responsible for detecting an overrun condition. When a received byte is transferred from the receive shift register to a receive FIFO that contains 32 unread bytes, the SIM receiver will flag an overrun condition. This condition will always set the overrun error flag, **oef** in the RCV_STATUS register. The received byte will be discarded leaving the 32 unread bytes in the FIFO unaltered. The SIM module will generate a NACK to the SIM card on an overrun condition if the **onack** bit in the CNTL register is set. The SIM module will continually NACK SIM card transmissions until a read of the receive FIFO occurs.

51.6.6 SIM Port Control

The SIM Port Control block has been designed to localize all port related circuitry into one section of hierarchy. The port control block is comprised of the following functions: SIM card interface, SIM Card presence detect, and SIM card auto-power down.

51.6.6.1 SIM Card Interface

The SIM module allows for direct control of two separate SIM cards. The SIM module does not support simultaneous communication with two SIM cards. In order to achieve simultaneous communication with two SIM cards, a second SIM module must be used. The SIM module can relinquish control of the inactive port to a secondary SIM module by setting the **amode** bit in the SETUP register.

The SIM card clock is generated in the SIM clock generator. The SIM card reset and SIM card voltage enable are controlled by software through the PORTX_CNTL registers.

Figure 51-19 below illustrates a example SIM module hookup to two SIM cards. The GCAP chip shown is used to provide Vcc for the SIM card, and level translators for the remaining signals when interfacing to a 5 volt card. The SIM module can directly access a 3 Volt SIM card if the Patriot I/O voltage is configured for 2.7V operation. The **3voltX** bit in the PORTX_CNTL registers configures the SIM Port transmit output as bi-directional. This frees up the SIM port receive pin to be used as general purpose I/O.

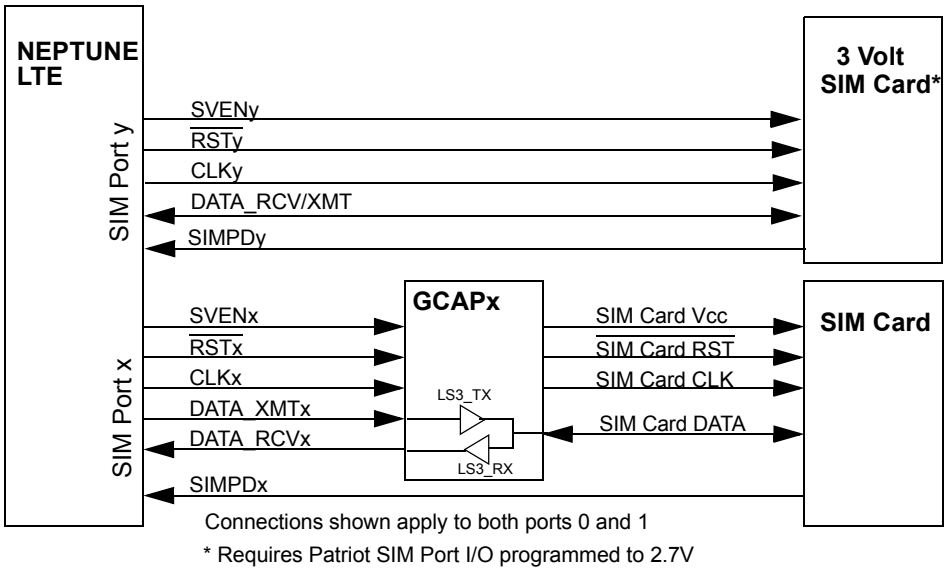


Figure 51-19. SIM Card Hookup to Patriot SIM Ports

51.6.6.2 SIM Card Presence Detect

The SIMPDx input allows for detection of the insertion or removal of a SIM card. The **spdsX** control bit in the PORTX_DETECT register allows the software to configure which edge of the SIMPDx pin will cause a presence detect event. A maskable interrupt can be generated when a SIMPDx event occurs.

An internal pulldown device is present on the SIMPDx pins for Neptune LTE. This will provide for a high to low transition on the SIMPDx pin when a SIM card is removed.

51.6.6.3 SIM Card Automatic Power Down

When interfacing to the SIM cards, it is necessary to follow a particular sequence when powering them up and down. The SIM port control block contains hardware that provides the correct sequence to power down a SIM card (see Figure 51-20). The power up sequence must be done manually by the software using the pin control bits supplied in the PORTx_CNTL registers.

The power down sequence is specified in ISO 7816 as:

1. RST transitions from high to low
2. CLK is turned off to a low
3. I/O transitions from tri-state to low
4. SIM Vcc is turned off

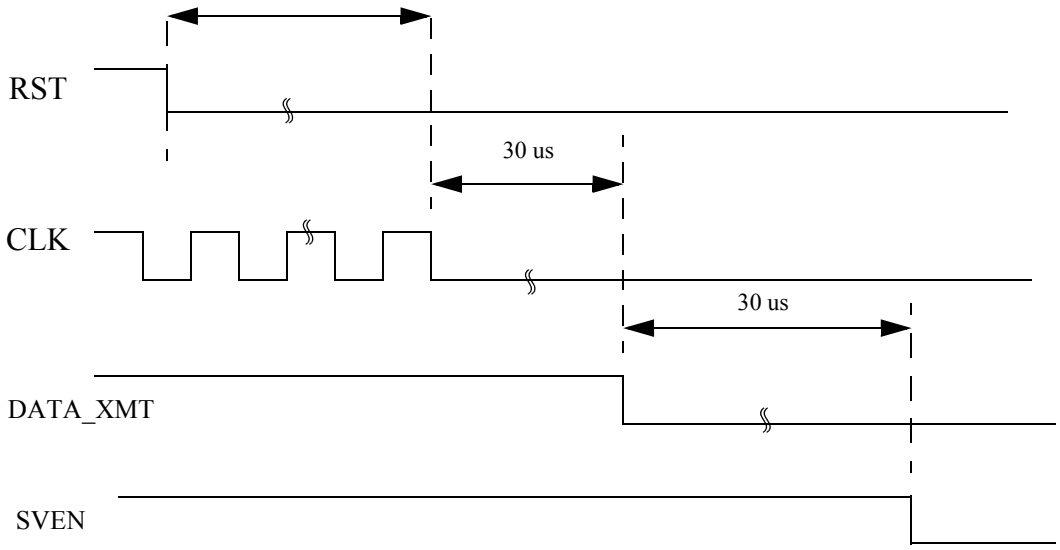


Figure 51-20. Automatic Power Down Sequence

51.6.6.4 SIM Card Forced Power Down

The **sfpdX** bit in each of the PORT_CNTL registers, adds a new capability to the SIM module, by forcing a SIM card power down. It is similar to the auto power down feature, except that it is not dependent upon either the state of the SIM card auto power down enable (**sapdX**) bit, or triggering of a presence detect event (**sdiX**).

51.6.7 SIM General Purpose Counter

The SIM module provides a 16-bit counter for use when timing events during SIM card communication. The clock source for the counter is selectable between three sources: baud_clk, rcv_clk, or xmt_ck (ETU clock). The **gpcnt_clksel[1:0]** bits in the CNTL register are used to select the clock input. The counter is enabled as soon as the input clock is selected. The starting of the counter is immediate once the input clock is running. Software can control the three input clock sources by using the **kill_clk**, **cwten, rcv_en** and **xmt_en** bits provided in the RESET_CNTL, CNTL, and ENABLE registers.

To run the counter from the card clock source the following conditions must be met:

1. **'kill_clk** = 0' in RESET_CNTL register
2. **'gpcnt_clksel[1:0]'** = 01' in CNTL register

The counter will begin to count at the card clock rate as soon as these conditions are met.

To run the counter from the receive clock source the following conditions must be met:

1. **'kill_clk** = 0' in RESET_CNTL register
2. **'rcv_en** = 1' or' **xmt_en** = 1' in ENABLE register
3. **'gpcnt_clksel[1:0]** = 10' in CNTL register

The counter will begin to count at the receive clock rate as soon as these conditions are met.

To run the counter from the ETU (transmit) clock source the following conditions must be met:

1. **kill_clk = 0** in RESET_CNTL register
2. Either one of the following:
   a. **rcv_en** = 1 in ENABLE register and **cwt_en** = 1 in CNTL register
   b. **xmt_en** = 1 in ENABLE register
3. **gpcnt_clksel[1:0]** = 11 in CNTL register

The counter will begin to count at the ETU (transmit) clock rate as soon as these conditions are met.

The counter can be reset by setting **gpcnt_clksel[1:0]** to 00. A 16-bit comparator value is provided that allows the software to select a count value at which an interrupt flag can be set and an interrupt generated if the mask is clear.

51.6.8 SIM LRC Block

The SIM module provides an 8-bit Linear Redundancy Check (LRC) generator / checker. The block is provided for use with T=1 SIM cards that support LRC. This block can be enabled through the **lrcen** bit in the CNTL register. This block performs an 8-bit exclusive-OR on all received or transmitted characters. At the end of the reception of a block of characters, the result is expected to be 00. If so, the **lrcok** bit is set in the RCV_STAT register. During transmission, the LRC block Exclusive-OR's each character that is transmitted with the current value of the LRC. If the **xmt_en_LRC_CRC** bit in the CNTL register is set, the LRC value will automatically be sent by the SIM transmitter as the final character when the transmit FIFO empties.

The LRC value can be reset in multiple ways. Clearing the **lcren** bit in the CNTL register will reset the LRC value. At the end of a transmission (either after the LRC byte is transmitted, or after the last character in the transmit FIFO is sent when **xmt_en_LRC_CRC** is clear), the LRC value is automatically reset by the SIM hardware. Finally, when setting the **xmt_en** bit, the SIM hardware resets the LRC value.

51.6.9 SIM CRC Block

The SIM module provides an 16-bit Cyclic Redundancy Check (CRC) generator / checker. The block is provided for use with T=1 SIM cards that support CRC. This block can be enabled through the **crcen** bit in the CNTL register. This block performs a polynomial based check on all received or transmitted characters. The polynomial description is shown in Figure 51-21.

At the end of the reception of a block of characters, the residual from the CRC calculation is compared to $1D0F. If so, the **crcok** bit is set in the RCV_STAT register. During transmission, the CRC block updates the current value of the CRC residual using each character. If the **xmt_en_LRC_CRC** bit in the CNTL register is set, the CRC value will automatically be inverted and sent by the SIM transmitter as the final two characters when the transmit FIFO empties.

The CRC value can be reset in multiple ways. Clearing the **crcen** bit in the CNTL register will reset the CRC value. At the end of a transmission (either after the CRC characters are transmitted, or after the last character in the transmit FIFO is sent when **xmt_en_LRC_CRC** is clear), the CRC value is automatically reset by the SIM hardware. Finally, when setting the **xmt_en** bit, the SIM hardware resets the CRC value.

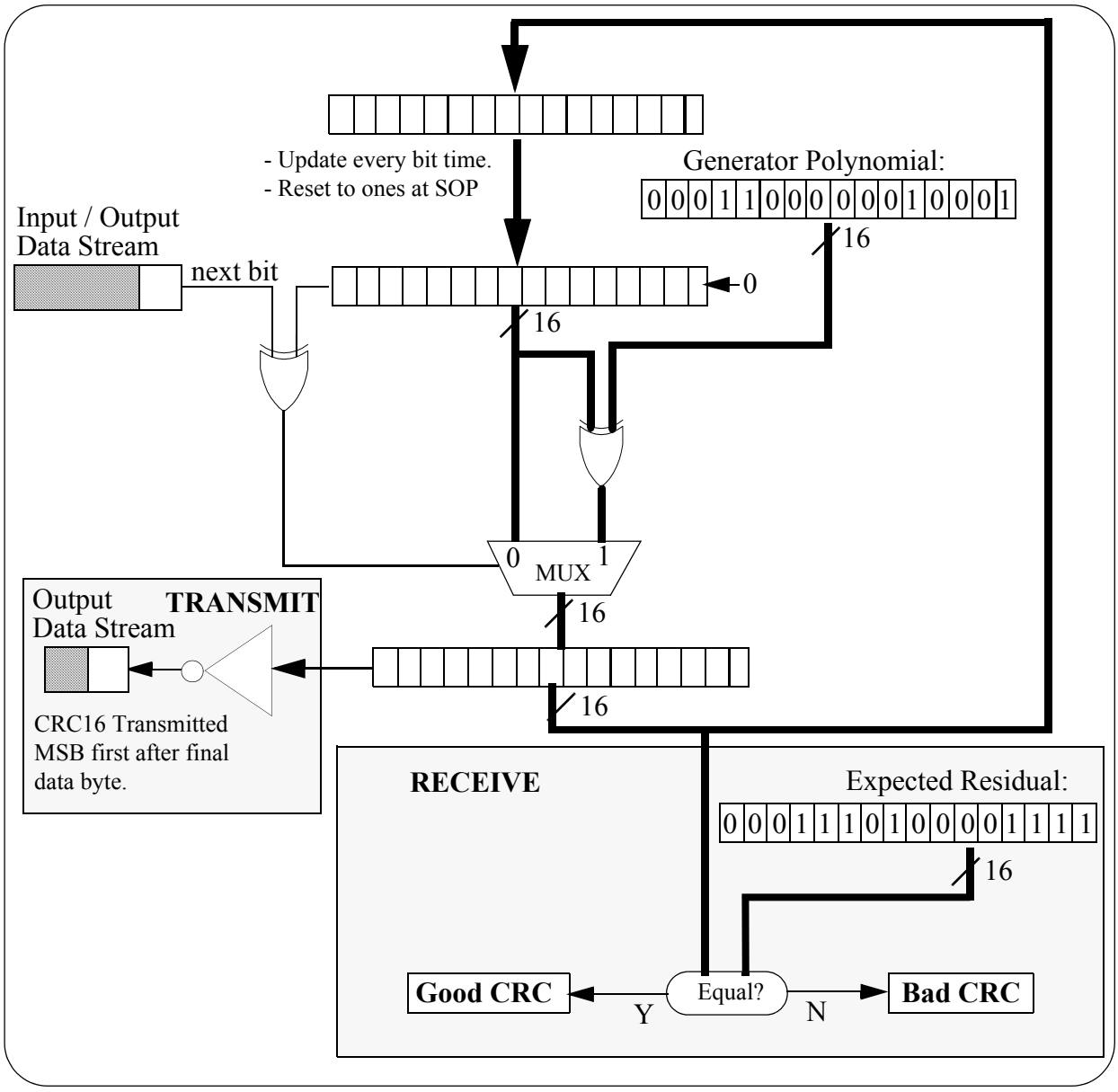


Figure 51-21. Cyclic Redundancy Check Circuit Diagram

51.6.10 SIM T=0 & T=1 features

51.6.10.1 Work Waiting Time / Character Wait Time

ISO/IEC 7816-3 requires that the time between the start bits of consecutively received characters be measured. This (etu) time, is called the Work Waiting Time (WWT) when using T=0 SIM cards, or Character Wait Time (CWT), when using T=1 SIM cards. It also requires the SIM interface to flag an error whenever the measured (etu) time is equal to, or greater than the value programmed in the SIM register.

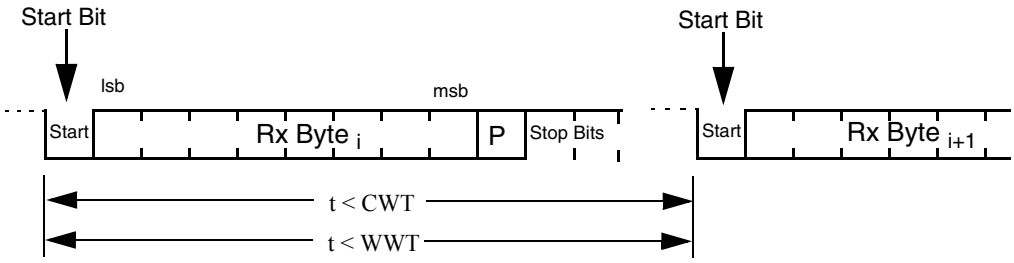


Figure 51-22. Work Waiting / Character Wait Time

Work Waiting Time is also defined as the time between the start bit of the last character transmitted and the first character received by the (SIM) card interface.

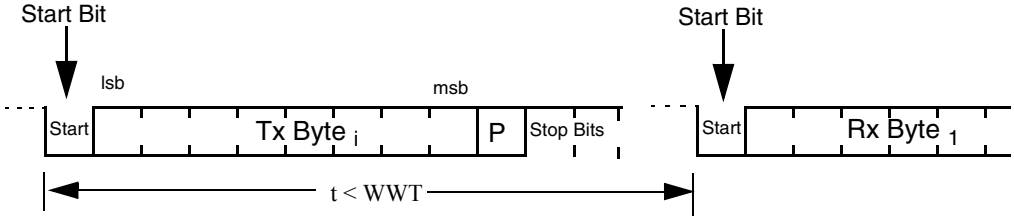


Figure 51-23. Work Waiting Time

51.6.10.2 Block Time

51.6.10.2.1 Block Guard Time (T=1 SIM cards only)

As defined in ISO/IEC 7816-3 block guard time is the minimum delay between the leading edges of two consecutive characters sent in opposite directions. BGT_{min} = 22 etus.

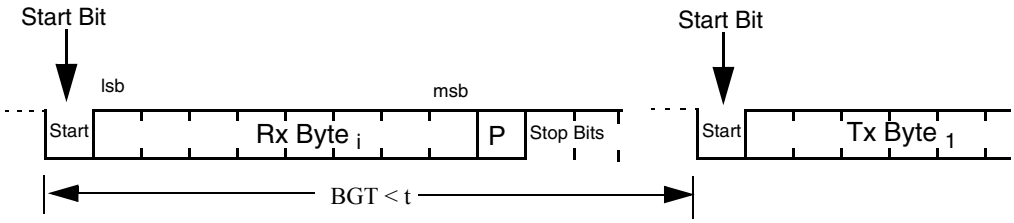


Figure 51-24. Block Guard Time

51.6.10.2.2 Block Wait Time

ISO/IEC 7816-3 defines block wait time as the maximum (etu) time between the leading edge of the last character of the block received by the card and the leading edge of the first character of the next block sent by the card. BWT is used to detect an unresponsive card.

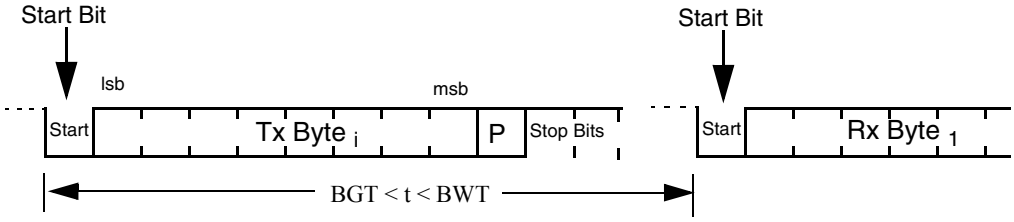


Figure 51-25. Block Wait Time

51.6.10.3 Register Interface

The **wwt_sel** bit in the **SETUP** register controls whether the (etu) time measured is relative to SIM T=0 cards or SIM T=1 cards. When **wwt_sel=1**, cwt_en and bt_en **(CNTL register)** bits, control the ability of the SIM interface to flag WWT violations, with **bt_en** bit enabling the logic to detect violations between the start bit of the last character transmitted and the first character received by the SIM interface, and **cwt_en** bit enabling the logic to detect violations between the start bits of two consecutively received characters.

Similarly, when **wwt_sel=0**, bt_en controls the logic to detect violations between the start bits of the last character transmitted out and the first character received by the SIM interface, while the **cwt_en** has the same function as described (above), when **wwt_sel=0**.

The only major difference between the **wwt_sel** bit states is that the Block Wait Time register is used to store/compare the value of the desired WWT, when **wwt_sel=1**, with CWT not being used. However, when **wwt_sel=0**, Character Wait Time register and the Block Wait Time Registers are used for their respective functions.

Note: Since Block Guard Time is only applicable to T=1 SIM cards, it is only enabled when wwt_sel=0.

51.6.10.4 Backward Software Compatibility

To guarantee backwards software compatibility when using an IC with this “new” design, some modifications were done, which would ensure a smooth and transparent software port onto the new design. Due to bug in the Character Wait Time logic, the CHAR_WAIT register had to be programmed to a value that subtracts 11 etus (1 start, 8 data, 1 parity, 2 stop bits), and adds 2 etus to compensate for the cwt counter clock error. In order to accommodate this error compensation methodology, whenever software would write to CHAR_WAIT register, a value of 11 etus would be added to the actual value use for comparison. Similarly, 11 etus would be subtracted when reading this register value. This will affect the out-of-reset read value of the CHAR_WAIT register, which will return a value of 0xFFF4.

For transitional to the new logic, a CHAR_WAIT_ACC (character wait time accurate) register is added. When software writes to this register, the actual value is used for comparison. Similarly, on a read access, actual value is output on the bus.

51.6.10.5 ETU Counter Architecture

51.6.10.5.1 ETU counter State machine

All types of measurements would be performed in bit time units (etu). ETU state machine controls the etu_counter block, which counts the number of etu cycles between characters, and flags any type of bit time violations. The receiver and the transmitter state machines provide the necessary control signals to the etu state machine such as the occurrence of a start bit or a transmission of a character from the SIM interface. The state description is as follows:

Table 4: ETU counter States

IDLE	This is the initial state when either receiver or the transmitter is enabled. The FSM is going to stay in this state until the either a valid start bit encountered or the character is loaded in the transmit shift register.
ETU_RX2TX_1:	FSM enters this state when a start bit is detected on the receive line. ETU counter clock is also going to be enabled in this state. FSM is going to remain in this state a unless a byte is loaded into the transmitter shift register, which is going to transfer control to ETU_TX. If the receiver is disabled, the FSM would go to IDLE state. If another start bit is detected, FSM will transfer to ETU_RX state.
ETU_RX:	This state acts as a reset state for the etu clock. On the next clock after entering this state the control is transferred to ETU_RX2TX_1, which again waits for a signal from either the receive or the transmit state machines.
ETU_TX2RX_1:	FSM enters this state when a new byte is loaded into the transmit shift register. ETU counter clock is also going to be enabled in this state. FSM is going to remain in this state a unless a start bit is detected, which is going to transfer control to ETU_RX. If the receiver is disabled, the FSM would go to IDLE state. If another byte is loaded into the shift register, FSM will transfer to ETU_TX state.
ETU_TX:	This state acts as a reset state for the etu clock. On the next clock after entering this state the control is transferred to ETU_TX2RX_1, which again waits for a signal from either the receive or the transmit state machines.

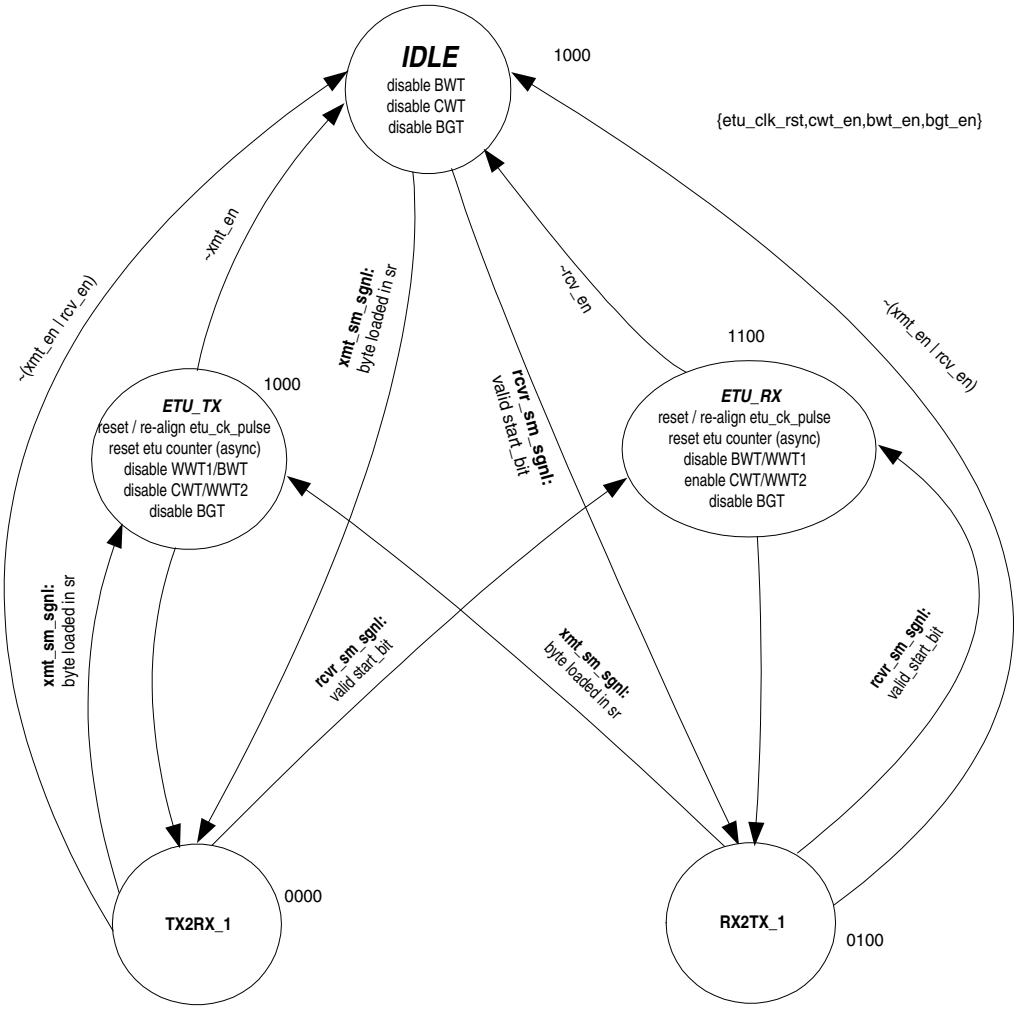


Figure 51-26. ETU Counter State Diagram

51.7 Module Interrupts

The following is a listing of all possible interrupt sources and their corresponding mask bits. All SIM interrupts are logically OR'ed to create the active low irq_n and data_irq_n signals that go to the ITC module. All mask bits are such that a logic 1 implies that the corresponding interrupt is disabled (masked), whereas a 0 implies that the interrupt enabled (unmasked). All of these mask bits are logic 1 out of reset (all interrupts are masked).

Table 51-12. SIM Module Interrupts

Flag	Flag register	Mask	Mask register	Description
tc	XMT_STATUS (page 51-61)	tcim	INT_MASK (page 51-66)	Transmit complete
etc		etcim		Early Transmit complete
tfe		tfeim		Transmit FIFO Empty
xte		xtm		Transmit threshold error
tfo		tfom		Transmit FIFO Overfill Error
tdtf		tdtfm		Transmit Data Threshold Flag
gpcnt		gpcntm		General Purpose Counter Comparator Flag
rdrf	RCV_STATUS (page 51-63)	rim		Receive data register full (FIFO threshold level reached)
oef		oim		Overrun error flag
cwt		cwtm		Character Wait Time Counter Comparator Flag
bwt		bwtm		Block Wait Time Counter Comparator Flag
bgt		bgtm		Block Guard Time Counter Comparator Flag
rte		rtm		Receive NACK Threshold Error Flag
sdi1	PORT1_DETECT (page 51-63)	sdim1	PORT1_DETECT (page 51-51)	SIM detect interrupt for port 1
sdi0	PORT0_DETECT (page 51-63)	sdim0	PORT0_DETECT (page 51-70)	SIM detect interrupt for port 0

51.8 Register Map

Figure 0-11.

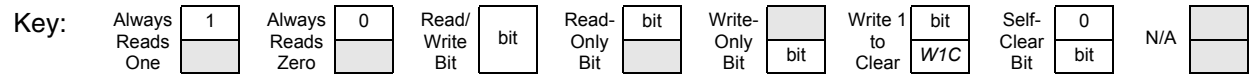


Table 51-13. Register Summary: $2484X000h - $2484XFFFh

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORT1_CNTL SIM1: ($2484_F000)	R	0	0	0	0	0	0	0	0	0	3volt1	SCSP1	SCEN1	SRST1	STEN1	SVEN1	SAPD1
	W									SFPD							
SETUP SIM1: ($2484_F002)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	wwt_sel	SPS	AMODE
	W																
PORT1_DETECT SIM1: ($2484_F004)	R	0	0	0	0	0	0	0	0	0	0	0	0	SPDS1	SPDP	SDI1	SDIM1
	W															W1C	
PORT1_XMT_BUF SIM1: ($2484_F00C)	R	0	0	0	0	0	0	0	0	port1_xmt[7:0]							
	W																
PORT1_RCV_BUF ($2484_F00E)	R	0	0	0	0	0	port1_cwt	port1_f	port1_pe	port1_rcv[7:0]							
	W																
PORT0_CNTL SIM1: ($2484_F010)	R	0	0	0	0	0	0	0	0	0	3volt0	SCSP0	SCEN0	SRST0	STEN0	SVEN0	SAPD0
	W									SFPD							
CNTL ($2484_F012)	R	bten	xmt crc	crcen	lrcen	cwten	gpcnt clk sel[1:0]		baud_sel[2:0]			clk sel[1:0]		ONACK	ANACK	ICM	0
	W																
RCV_THRESHOLD SIM1: ($2484_F014)	R	0	0	0	0	0	0	0	RTH[3:0]				RDT[4:0]				
	W																
ENABLE SIM1: ($2484_F016)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	xmten	rcven
	W																
XMT_STATUS SIM1: ($2484_F018)	R	0	0	0	0	0	0	0	gpcnt	TDTF	TFO	TC	ETC	TFE	0	0	XTE
	W								W1C	W1C	**	W1C	W1C	W1C			W1C
RCV_STATUS ($2484_F01A)	R	0	0	0	0	bgt	bwt	RTE	cwt	CRC	LRCO	RDRF	RFD	0	0	0	OEF
	W					W1C	W1C	W1C	W1C								W1C
INT_MASK ($2484_F01C)	R	0	0	0	bgtm	bwtm	RTM	cwtm	gpcntm	tdtfm	TFOM	XTM	TFEIM	ETCIM	OIM	TCIM	RIM
	W																
PORT0_XMT_BUF ($2484_F01E)	R	0	0	0	0	0	0	0	0	port0_xmt[7:0]							
	W																
PORT0_RCV_BUF ($2484_F020)	R	0	0	0	0	0	port0_cwt	port0_f	port0_pe	port0_rcv[7:0]							
	W																
PORT0_DETECT SIM1: ($2484_F022)	R	0	0	0	0	0	0	0	0	0	0	0	0	SPDS0	SPDP	SDI0	SDIM0
	W															W1C	
DATA_FORMAT SIM1: ($2484_F024)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	IC
	W																
XMT_THRESHOLD SIM1: ($2484_F026)	R	0	0	0	0	0	0	0	0	XTH[3:0]			TDT3:0]				
	W																
GUARD_CNTL SIM1: ($2484_F028)	R	0	0	0	0	0	0	0	RCVR11	GETU[7:0]							
	W																
OD_CONFIG SIM1: ($2484_F02A)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	OD_P1	OD_P0
	W																
RESET_CNTL SIM1: ($2484_F02C)	R	0	0	0	0	0	0	0	0	0	DBUG	STOP	DOZE	kill clock	0	flush xmt	flush rcv
	W														soft rst		
CHAR_WAIT SIM1: ($2484_F02E)	R	Character Wait Time[15:0]															
	W																
GPCNT SIM1: ($2484_F030)	R	General Purpose Counter[15:0]															
	W																
DIVISOR SIM1: ($2484_F032)	R	0	0	0	0	0	0	0	0	0	Divisor[6:0]						
	W																
BLOCK_WAIT_MSB ($2484_F034)	R	Block Wait Time Counter [31:16]															
	W																
BLOCK_WAIT_LSB ($2484_F036)	R	Block Wait Time Counter [15:0]															
	W																

Table 51-13. Register Summary: $2484X000h - $2484XFFFh

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHAR_WAIT_ACC ($2484_F038)	R	Character Wait Time (Accurate) [15:0]															
	W																
BLOCK_GUARD ($2484_F03A)	R	0	0	0	0	0	0	0	0	Block Guard Time Counter [7:0]							
	W																

Note: ** Refer to bit description in the Detailed Register Description section for clearing information

51.8.1 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various SIM registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

PORT1_CNTL Port 1 Control Register

Addr (SIM1) $2484_F000 (SIM2) $2A8A_0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									sfpd1	3volt1	scsp1	scen1	srst1	sten1	sven1	sapd1
TYPE:	r	r	r	r	r	r	r	r	slfclr	rw	rwm	rwm	rwm	rwm	rwm	rwm
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 51-14. PORT1_CNTL Description

Name	Description	Settings
sfpd1 Bit 7	**SIM card force power down Port 1** - Used to force a power-down on port1, regardless of the state of **sapd1** (bit0) in this register or **sdi1** flag (bit1) in the PORT1_DETECT register. This bit is self-clearing. **Note:** This feature is not available in CDR3 (i.e., Patriot RAM1.x, RAM 2.x, ROM 1.x)	0 = No power-down forced on Port 1 (default) 1 = Force a power-down sequence on Port 1.
3volt1 Bit 6	**3 Volt SIM card Port 1 —** Used to configure the Port 1 transmit pin as bi-directional. This allows the Port 1 Receive pin to be re-used as General Purpose I/O. This operation is restricted to 3 Volt SIM cards only. In addition, the Patriot SIM I/O must operate at 2.7V in this mode. **Note:** This bit should not be changed while the receiver or transmitter is enabled! **Note:** The bit is not readable in Patriot RAM1 due to errata.	0 = Port 1 requires both transmit and receive pins (default). 1 = Port 1 transmit pin is bi-directional. Port 1 receive pin is unused.
scsp1 Bit 5	**SIM card Clock Stop Polarity Port 1 —** Used to control the polarity of the idle SIM clock when the clock is disabled by **scen1**. It will be forced low by hardware during the auto power down sequence. This forces the clock be a logic 0 when stopped by auto power down as required by ISO 7816 spec.	0 = Clock is logic 0 when stopped by **scen1** (default). 1 = Clock is logic 1 when stopped by **scen1**.
scen1 Bit 4	**SIM card Clock Enable Port 1 —**Used to enable/disable the clock to the SIM card. It can be forced low by hardware during the auto power down sequence.	0 = SIM Card Clock Disabled Port 1(default). 1 = SIM Card Clock Enabled Port 1
srst1 Bit 3	**SIM card Reset—**Used to control state of reset line to the SIM card. It can be forced low by hardware during the auto power down sequence. SIM card reset signals are active low.	0 = SIM Card reset Port1 inactive (default). 1 = SIM Card reset Port1 active
sten1 Bit 2	**SIM card Transmit Enable Port 1—**Used to enable/disable the XMT data to the SIM card. It can be forced low by hardware during the auto power down sequence.	0 = Port1 Transmit Data is forced to zero (default). 1 = Port 1 Transmit Data controlled by SIM module.

Table 51-14. PORT1_CNTL Description

Name	Description	Settings
sven1 Bit 1	**SIM card Vcc Enable Port 1—**Used to control the state of the SVEN1 pin on SIM card port 1. The SVEN1 pin controls the SIM card Vcc enable in the GCAP chip. It can be forced low by hardware during the auto power down sequence.	0 = SIM card Voltage Port 1 disabled (default). 1 = SIM card Voltage Port 1 enabled
sapd1 Bit 0	**SIM card Auto Power Down Port 1—**Used to enable/disable the auto power down function for port 1. It will be forced low at the end of an auto power down sequence, including a forced power down triggered by writing a one to **sfpd1** (bit7) this register.	0 = Auto power down Port 1 disabled (default). 1 = Auto power down Port 1 enabled

Note: The PORT1_CNTL register should not be read when the **amode** bit in the SETUP register is set.

SETUP Setup Register

Addr (SIM1) $2484_F002 (SIM2) $2A8A_0002

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														wwt_sel	sps	amode
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 51-15. SETUP Description

Name	Description	Settings
wwt_sel Bit 2	**Work Waiting Time Toggle —** Controls the source of cwt_flag and bwt_flags, when using either T=0 or T=1 SIM cards. **Note:** When wwt_sel is set, **only bwt_msb[15:0]** and **bwt_lsb[15:0]** shall be used to store the 32-bit value needed for wwt measurement. In the default mode **cwt[15:0]** and **bwt_msb[15:0] / bwt_lsb[15:0]** shall be used for two seperate measurements.	0 = character wait time / block wait time active (default) 1 = work waiting time active
sps Bit 1	**SIM card Port Select—**Controls which port the SIM interface uses **Note:** The **amode** bit must be zero when the **sps** bit is set to one.	0 = Port 0 Enabled (default). 1 = Port 1 Enabled
amode Bit 0	**Alternate SIM Card Mode enable—**Enables an slternate SIM module to control SIM card Port 1. **Note:** The **sps** bit must be zero to give the alternate SIM module control.	0 = Alternate Port Disabled (default). 1 = Alternate Port Enabled

PORT1_DETECT Detect Register for Port 1

Addr (SIM1) $2484_F004 (SIM2) $2A8A_0004

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													spds1	spdp1	sdi1	sdim1
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	rw	r	w1c	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 51-16. PORT1_DETECT Description

Name	Description	Settings
spds1 Bit 3	**SIM Presence Detect Select Port 1—**Controls which edge of the SIMPD1 pin is used to generate the sdi1 interrupt.	0 = Falling edge of SIMPD1 Input (default). 1 = Rising edge of SIMPD1 Input
spdp1 Bit 2	**SIMPD1 input pin status—**This bit reflects the state of the SIMPD1 pin. It is not a latched register bit, but instead a synchronized version of the state of the SIMPD1 pin itself. **Note:** Patriot RAM1 requires two reads of this register to get current pin status due to errata.	0 = SIMPD1 pin is logic low 1 = SIMPD1 pin is logic high
sdi1 Bit 1	**SIM Detect Interrupt flag Port 1—**Status flag to indicate the insertion or removal of a SIM card has been detected on port 1. Can create an interrupt to the MCU if **sdim1** is low. Write a 1 to this bit to clear.	0 = No insertion or removal of SIM card detected on Port 1(default). 1 = Insertion or removal of SIM card detected on Port 1
sdim1 Bit 0	**SIM Detect Interrupt Mask Port 1—**Interrupt mask for the **sdi1** interrupt flag.	0 = **sdi1** enabled 1 = **sdi1** masked (default)

Note: In summary, **spds1** determines which edge transition of the SIMPD1 pin is used for SIM card presence detection. Presence detection can be used to determine if the card has been inserted or removed. The occurrence of the SIMPD1 edge specified by **spds1** will cause the following: **sdi1** to be set; if the **sdim1** mask is clear, an interrupt on SIMIRQ_N; and if **sapd1** in the PORT1_CNTL register is set, an auto power down sequence to begin. If SIM card insertion is expected, **sapd1** can be set low to avoid the auto power down sequence. There is no auto power up sequence. The bit **spdp1** can be used to determine the current state of the SIMPD1 pin.

PORT1_XMT_BUF Port 1 Transmit Buffer Register

Addr
(SIM1)
$2484_F00C
(SIM2)
$2A8A_000C

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									port1_xmt[7:0]							
TYPE:	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 51-17. PORT1_XMT_BUF Description

Name	Description	Settings
port1_xmt[7:0] Bits 7–0	**Port1 Transmit Buffer—**Write to the next available location in the transmit buffer. Writes to this register are ignored when the **sps** bit is zero.	N/A

Note: Writing more data to the transmit FIFO than it can hold (16 bytes) will cause a transmit FIFO Overfill error.

Note: Patriot RAM1 has errata for the Transmit FIFO Overfill error. It does not function properly on RAM1.

PORT1_RCV_BUF Port 1 Receive Buffer Register

Addr $2484_F00E

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						port1_cwt	port1_fe	port1_pe	port1_rcv[7:0]							
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 51-18. PORT1_RCV_BUF Description

Name	Description	Settings
port1_cwt Bit 10	**Port1 Character Wait / Work Waiting Time Violation Flag —** This bit indicates whether a character wait time (T=1 SIM cards), or a work waiting time (T=0 SIM cards) violation has occurred, between the last couple of consecutive bytes received. **Note:** This flag may be set on the first character received by the card, since it is comparing against time zero, and there is not previously byte received.	0 = No violation occurred 1 = violation has occurred between the last two consecutive bytes received.
port1_fe Bit 9	**Port1 Frame Error flag—**The **port1_fe** flag indicates whether a frame error was detected during the reception of the corresponding byte read in the **port1__rcv** field. The following table indicates the effect **sps** has on the source of the **port1_fe** flag. The **port1_fe** flag cannot create an interrupt. It need not be cleared since it will be overwritten by the next byte received into that location of the FIFO.	0 = Byte contains no framing error (default). 1 = Byte contains a framing error
port1_pe Bit 8	**Port1 Parity Error flag—**The **port1_pe** flag indicates whether a parity error was detected during the reception of the corresponding byte read in the **port1__rcv** field. The following table indicates the effect **sps** has on the source of the **port1_pe** flag. The **port1_pe** flag cannot create an interrupt. It need not be cleared since it will be overwritten by the next byte received into that location of the FIFO. **Note:** Patriot RAM1 will not show a parity error on invalid initial characters during initial character mode. This errata is fixed for Patriot ROM1.	0 = Byte contains no parity error (default). 1 = Byte contains a parity error
port1_rcv[7:0] Bits 7–0	**Port1 Receive buffer—**Read from the next location in the receive buffer. Reads from this register return zero when the **sps** bit is zero	N/A

PORT0_CNTL Port 0 Control Register **Addr (SIM1) $2484_F010 (SIM2) $2A8A_0010**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									sfpd0	3volt0	scsp0	scen0	srst0	sten0	sven0	sapd0
TYPE:	r	r	r	r	r	r	r	r	slfclr	rw	rwm	rwm	rwm	rwm	rwm	rwm
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 51-19. PORT0_CNTL Description

Name	Description	Settings
sfpd0 Bit 7	**SIM card force power down Port 0**- Used to force a power-down on port0, regardless of the state of **sapd0** (bit0) in this register or **sdi0** flag (bit1) in the PORT0_DETECT register. This bit is self-clearing. **Note:** This feature is not available in CDR3 (i.e., Patriot RAM1.x, RAM 2.x, ROM 1.x)	0 = No power-down forced on Port 0 (default) 1 = Force a power-down sequence on Port 0.
3volt0 Bit 6	**3 Volt SIM card Port 0 —** Used to configure the Port 0 transmit pin as bi-directional. This allows the Port 0 Receive pin to be re-used as General Purpose I/O. This operation is restricted to 3 Volt SIM cards only. In addition, the Patriot SIM I/O must operate at 2.7V in this mode. **Note:** This bit should not be changed while the receiver or transmitter is enabled! **Note:** This bit is not readable in Patriot RAM1 due to errata.	0 = Port 0 requires both transmit and receive pins (default). 1 = Port 0 transmit pin is bi-directional. Port 0 receive pin is unused.
scsp0 Bit 5	**SIM card Clock Stop Polarity Port 0 —** Used to control the polarity of the idle SIM clock when the clock is disabled by **scen0**. It will be forced low by hardware during the auto power down sequence. This forces the clock be a logic 0 when stopped by auto power down as required by ISO 7816 spec.	0 = Clock is logic 0 when stopped by **scen0** (default). 1 = Clock is logic 1 when stopped by **scen0**.
scen0 Bit 4	**SIM card Clock Enable Port 0 —**Used to enable/disable the clock to the SIM card. It can be forced low by hardware during the auto power down sequence.	0 = SIM Card Clock Disabled Port 0(default). 1 = SIM Card Clock Enabled Port 0
srst0 Bit 3	**SIM card Reset—**Used to control state of reset line to the SIM card. It can be forced low by hardware during the auto power down sequence. SIM card reset inputs are active low.	0 = SIM Card reset Port0 inactive (default). 1 = SIM Card reset Port0 active
sten0 Bit 2	**SIM card Transmit Enable Port 0—**Used to enable/disable the XMT data to the SIM card. It can be forced low by hardware during the auto power down sequence.	0 = Port0 Transmit Data is forced to zero (default). 1 = Port 0 Transmit Data controlled by SIM module.

Table 51-19. PORT0_CNTL Description

Name	Description	Settings
sven0 Bit 1	**SIM card Vcc Enable Port 0—**Used to control the state of the SVEN0 pin on SIM card port 0. The SVEN0 pin controls the SIM card Vcc enable in the GCAP chip. It can be forced low by hardware during the auto power down sequence.	0 = SIM card Voltage Port 0 disabled (default). 1 = SIM card Voltage Port 0 enabled
sapd0 Bit 0	**SIM card Auto Power Down Port 0—**Used to enable/disable the auto power down function for port 0. It will be forced low at the end of an auto power down sequence, including a forced power down triggered by writing a one to **sfpd0** (bit7) this register.	0 = Auto power down Port 0 disabled (default). 1 = Auto power down Port 0 enabled

CNTL Control Register **Addr $2484_F012**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	bten	xmt crc / lrc	crcen	lrcen	cwten	gpcnt_clk_sel[1:0]		baud_sel[2:0]			clk_sel[1:0]		onack	anack	icm	
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rwm	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Table 51-20. CNTL Description

Name	Description	Settings
bten Bit 15	Block Time / Work Waiting Time Select — **Selects whether the block guard time / block wait time detection logic (T=1 SIM cards) or the Work Waiting Time (T=0 SIM cards) logic is enabled.** **Note:** If **wwt_sel=1** this bit has no effect.	0 = Block Wait / Block Guard Time disabled. 1 = Block Wait / Block Guard Time enabled
xmt_crc_lrc Bit 14	**Transmit CRC or LRC—** This bit specifies whether or not to transmit the redundancy checking data at the end of a transmission (i.e. when the FIFO becomes empty). **Note:** This feature is not available on Patriot RAM1.	0 = No Redundancy check info transmitted (default). 1 = Transmit LRC or CRC info when FIFO empties (whichever is enabled)
crcen Bit 13	**CRC Enable—** This bit enables the calculation of the 16-bit CRC value for both Receiver and transmitter. The result of the calculation is continuously compared to the expected remainder and reflected in the **crcok** bit in the RCV_STAT register. Clearing this bit resets the current CRC residual value in the SIM hardware. **Note:** This feature is not available on Patriot RAM1.	0 = 16-bit Cyclic Redundancy Checking disabled (default). 1 = 16-bit Cyclic Redundancy Checking enabled
lrcen Bit 12	**LRC Enable —** This bit enables the calculation of the 8-bit LRC value for both Receiver and transmitter. The result of the calculation is continuously compared to zero and reflected in the **lrcok** bit in the RCV_STAT register. Clearing this bit resets the current LRC value in the SIM hardware. **Note:** This feature is not available on Patriot RAM1.	0 = 8-bit Linear Redundancy Checking disabled (default). 1 = 8-bit Linear Redundancy Checking enabled

Table 51-20. CNTL Description

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>cwten
Bit 11</td><td>Character Wait / Work Waiting Time Select — Enables the character wait time (T=1 cards) or the work waiting time (T=0 cards) detection logic. Clearing this bit would disable the respective functionality of the Internal ETU counter.
Note: This feature is not available on Patriot RAM1.
Note: If wwt_sel=1 this bit has no effect.</td><td>0 = Violation would not be flagged (default).
1 = Violation would be flagged.</td></tr>
<tr><td>gpcnt clk_sel[1:0]
Bits 10-9</td><td>General Purpose Counter Clock Select—Selects which clock source is used by SIM Module general purpose counter. The only way to reset the counter is to set these bits to zero. The counter will begin counting as soon as the clock input is selected and the clocks are enabled (see Section 51.6.6.4). These input clocks are enabled through other register bits of the SIM module (kill_clock, rcv_en, and xmt_en respectively)
Note: This feature is not available on Patriot RAM1.</td><td><table><tr><th>gpcnt clk_sel[1:0]</th><th>Frequency</th></tr><tr><td>00</td><td>Disabled / Reset</td></tr><tr><td>01</td><td>Card Clock</td></tr><tr><td>10</td><td>Receive Clock</td></tr><tr><td>11</td><td>ETU Clock (transmit clock)</td></tr></table></td></tr>
<tr><td>baud_sel[2:0]
Bits 8–6</td><td>SIM Baud Rate Select—Selects the asynchronous baud rate divisor of the clock When set to “111”, the divisor is set to the value programmed in the DIVISOR register. This allows for more flexible baud rate determination. Also sets the corresponding sample rate which is the number of times a bit being received is sampled.
Note: The “DIVISOR Reg” option is not available on Patriot RAM1.</td><td><table><tr><th>baud_sel[2:0]</th><th>Divisor</th><th>Sample Rate</th></tr><tr><td>000</td><td>372</td><td>12</td></tr><tr><td>001</td><td>256</td><td>16</td></tr><tr><td>010</td><td>128</td><td>16</td></tr><tr><td>011</td><td>64</td><td>16</td></tr><tr><td>10x</td><td>32</td><td>16</td></tr><tr><td>110</td><td>16</td><td>16</td></tr><tr><td>111</td><td>DIVISOR Reg</td><td>16</td></tr></table></td></tr>
<tr><td>clk_sel[1:0]
Bits 5–4</td><td>SIM card Clock Select—Selects which clock source is used by baud rate generator and SIM cards.
Note: The “ref” option is not valid for use with SIM cards. This is provided when using the SIM module as a simple UART.
Note: The “ref” option is not available on Patriot RAM1.</td><td><table><tr><th>clk_sel[1:0]</th><th>Frequency</th></tr><tr><td>00</td><td>ref / 4</td></tr><tr><td>01</td><td>ref / 3</td></tr><tr><td>10</td><td>ref / 5</td></tr><tr><td>11</td><td>ref</td></tr></table></td></tr>
<tr><td>onack
Bit 3</td><td>Overrun NACK Enable—Enables overrun NACK generation.</td><td>0 = NACK generation on overrun is disabled (default).
1 = NACK generation on overrun is enabled</td></tr>
</table>

Table 51-20. CNTL Description

Name	Description	Settings
anack Bit 2	**Automatic NACK Enable—**Enables nack generation for parity errors or invalid initial characters when in ICM mode.	0 = NACK generation on errors disabled (default). 1 = NACK generation on errors enabled.
icm Bit 1	**Initial Character Mode—**Enables initial character mode. Will be automatically cleared by hardware once a valid initial character is received.	0 = Initial Character Mode disabled (default). 1 = Initial Character Mode enabled
reserved Bit 0	**Reserved**	Always reads zero.

Note: Changing **clk_sel[1:0]** not only affects the transmit baud rate, but it also determines the frequency of the SIM card clock driven on the SIM card port clock pins. The card clock is disabled when **clk_sel[1:0]** is set to 11.

Note: The CNTL register should not be written while the SIM transmitter or receiver is enabled!

RCV_THRESHOLD Receive Threshold Register **Addr (SIM1) $2484_F014 (SIM2) $2A8A_0014**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								rth[3:0]				rdt[4:0]				
TYPE:	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 51-21. RCV_THRESHOLD Description

Name	Description	Settings
rth[3:0] Bits 8-5	Receive Nack Threshold—**Used to specify the number of consecutive NACKs transmitted by the SIM module, for a given character, before the receive threshold error (rte) flag is triggered. A value of 0 indicates rte would never be set.** When a valid character is received by the SIM, the internal counter keeping track of the NACK count resets to zero for the subsequent byte being received. **Note:** If **anack** bit is clear in the CNTL register, **rth** has no effect. **Note:** This feature is not available in CDR3 (i.e., Patriot RAM1.x, RAM 2.x, ROM 1.x)	N/A
rdt[4:0] Bit 4–0	**Receive Data Threshold—**Determines the number of unread bytes that must exist in the FIFO to trigger the receive data register full (**rdrf**) interrupt flag. If the number of unread bytes in the receive FIFO is greater than or equal to the value in **rdt**, the **rdrf** flag in the RCV_STATUS register will be set. A value of zero indicates that there must be 32 unread bytes in the FIFO to trigger **rdrf**. The **rdt** value can be altered at any time, and the **rdrf** flag will be updated accordingly.	N/A

ENABLE Transmit/Receive Enable Register

Addr
(SIM1)
$2484_F016
(SIM2)
$2A8A_0016

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															xmt_en	rcv_en
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 51-22. ENABLE Description

Name	Description	Settings
xmt_en Bit 1	**SIM Transmit Enable—**Used to enable/disable the SIM transmit state machine. When the SIM is being used to receive data, **xmt_en** should be deasserted. This bit also enables the xmt_clk and rcv_clk inputs to the General Purpose Counter. **Note:** Setting this bit (transition from 0 to 1) will reset the CRC and LRC values.	0 = SIM Transmitter disabled (default). 1 = SIM Transmitter enabled
rcv_en Bit 0	**SIM Receiver Enable—**Used to enable/disable the SIM receive state machine. The **rcv_en** bit should be left high whenever the SIM module is in use. The SIM module has an *automatic receive mode* operation that disables the reception of characters when the transmitter is operational. Once the transmitter has completed sending the last character, the receiver is automatically enabled. This bit also enables the rcv_clk input to the General Purpose Counter.	0 = SIM Receiver disabled (default). 1 = SIM Receiver enabled

XMT_STATUS Transmit Status Register **Addr (SIM1) $2484_F018 (SIM2) $2A8A_0018**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								gpcnt	tdtf	tfo	tc	etc	tfe			xte
TYPE:	r	r	r	r	r	r	r	w1c	w1c	n/a	w1c	w1c	w1c	r	r	w1c
RESET:	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0

Table 51-23. XMT_STATUS Description

Name	Description	Settings
gpcnt Bit 8	**General purpose Counter Flag—**Used to indicate when the General purpose counter has reached the value in the GPCNT register. **Note:** This feature is not available on Patriot RAM1.	0 = GPCNT time not reached, or bit has been cleared. (default). 1 = General Purpose counter has reached the GPCNT value.
tdtf Bit 7	**Transmit Data Threshold Flag—**Used to indicate that the number of bytes in the transmit FIFO is less than or equal to the value programmed in the **tdt[3:0]** bits in the XMT_THRESHOLD register. **Note:** This feature is not available on Patriot RAM1.	0 = Number of bytes in FIFO is greater than **tdt[3:0]**, or bit has been cleared (default). 1 = Number of bytes in FIFO is less than or equal to **tdt[3:0]**
tfo Bit 6	**Transmit FIFO Overfill Error—**Used to indicate when the Transmit FIFO has been written with more than 16 bytes. The tfo bit can only be cleared by setting the **flush_xmt** or **soft_reset** bit in the RESET_CNTL register. **Note:** This feature does not work on Patriot RAM1 due to errata.	0 = No transmit FIFO overfill error has occurred (default). 1 = A Transmit FIFO overfill error has occurred.
tc Bit 5	**Transmit Complete—**Used to indicate whether the SIM transmitter is ready for a new transmission. The **tc** flag becomes set after the guard time has expired for the last byte in the transmit FIFO. The **tc** flag will create an interrupt if **tcim** in the INT_MASK register is low. The **tc** bit is a write-one-to-clear bit. **Note:** This feature does not work properly on Patriot RAM1 due to errata.	0 = Transmit pending or in progress 1 = Transmit complete (default)
etc Bit 4	**Early Transmit Complete—**Used to indicate that the SIM transmitter has finished sending the current byte and the transmit FIFO is empty. This bit differs from the **tc** bit in that it is set before the guard time of the last byte has elapsed. The **etc** flag will create an interrupt if **etcim** in the INT_MASK register is low. The **etc** bit is a write-one-to-clear bit **Note:** This feature does not work properly on Patriot RAM1 due to errata.	0 = Transmit pending or in progress 1 = Transmit complete (default)

Table 51-23. XMT_STATUS Description

Name	Description	Settings
tfe Bit 3	**Transmit FIFO Empty—**Used to indicate that the SIM transmit FIFO has emptied. This bit will be set when the last byte in the transmit FIFO has been transferred to the SIM transmitter shift register. The **tfe** flag will create an interrupt if tfeim in the INT_MASK register is low. The **tfe** bit is a write-one-to-clear bit **Note:** This bit will not be set when transmit FIFO is cleared via the **flush_xmt** bit in the RESET_CNTL register. **Note:** This feature does not work properly on Patriot RAM1 due to errata.	0 = Transmit shift register contains data 1 = Transmit shift register does not contain data (default)
reserved Bits 2–1	Unused bits	N/A
xte Bit 0	**Transmit Threshold Error—**Used to indicate the transmit NACK threshold has been reached. When **xte** is high, no further transmissions will be done until the **xte** flag is cleared. Any data transmissions still pending in the transmit FIFO will be aborted, and the **tc, etc,** and **tfe** flags will be set. The **xte** flag will create an interrupt if **xtm** in the INT_MASK register is low. The **xte** bit is a write-one-to-clear bit **Note:** This feature does not work properly on Patriot RAM1 due to errata.	0 = Transmit NACK threshold has not been reached (default). 1 = Transmit NACK threshold reached; transmitter frozen.

RCV_STATUS Receive Status Register **Addr $2484_F01A**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					bgt	bwt	rte	cwt	crcok	lrcok	rdrf	rfd				oef
TYPE:	r	r	r	r	w1c	w1c	w1c	w1c	r	r	r	r	r	r	r	w1c
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 51-24. RCV_STATUS Description

Name	Description	Settings
bgt Bit 11	**Block Guard Time Flag (BGT) —** Used to indicate whether a BGT violation has occurred. See Example 51.6.10.2.1 for details about BGT and its operation. **bgt** is a write-one-to-clear bit.	0 = No BGT violation has occurred (default) 1 = BGT violation has occurred
bwt Bit 10	**Block Wait Time (BWT) —** Used to indicate whether a BWT (T=1 cards) or a WWT (T=0 cards) violation has occurred. See Example 51.6.10.2.2 for detail about BWT and its operation. This bit is of a write-one-to-clear type. **Note:** If **wwt_sel =1**, this bit would indicate wwt violation between the last character transmitted by the SIM module and the first character received.	0 = No violation has occurred (default) 1 = Time between the last character transmitted and the first character received is greater than or equal to the value programmed in WORK_WAITING_TIME (combination of BWT_MSB and BWT_LSB) register.
rte Bit 9	Receive NACK Threshold Error Flag**—Used to indicate whether the number of consecutive NACKs generated by the SIM module in response to receive parity errors, for the byte being received, equals the value programmed in rth[3:0] in the RCV_THRESHOLD register.** This bit would never be set unless the **anack** bit is set in the CNTL register. **The sapdX (bit0) in the PORTx_CNTL register must be set to enable the threshold error to trigger the auto power down sequence. rte** is a write-one-to-clear bit. **Note:** Clearing this bit also resets the internal counter for consecutive NACKs being transmitted, for a given byte. **Note:** Software should verify that the **sapdX** is clear before clearing the **rte** flag. **Note:** This feature is not available in CDR3 (i.e., Patriot RAM1.x, RAM 2.x, ROM1.x)	0 = Number of NACKs generated by the receiver is less than the value programmed in rth[3:0] 1 = Number of NACKs generated by the receiver on the current byte is equal to the value programmed into **rth[3:0]** (Receive NACK Threshold register)

Table 51-24. RCV_STATUS Description

Name	Description	Settings
cwt Bit 8	**Character Wait Time (CWT) —**Used to indicate when the time between received characters is equal to or greater than the value programmed in the CHAR_WAIT (T=1 cards) or the WORK_WAITING (T=0 cards) register. **Note:** This feature is not available on Patriot RAM1. **Note:** If wwt_sel=1, this bit would indicate a wwt violation between the last two characters received by the SIM module.	0 = No violation has occurred (default). 1 = Time between two consecutive characters exceeded the value in CHAR_WAIT or the WORK_WATING_TIME register.
crcok Bit 7	**Cyclic Redundancy Check Okay flag—**Used to indicate when the calculated 16-bit CRC value matches the expected value for the current input data stream. The value is calculated across all received characters from the point the **crcen** bit is set in the CNTL register. The current CRC residual can be reset by three mechanisms: • Clear **crcen** bit in CNTL register • Set **xmt_en** bit in ENABLE register • Automatically by hardware when **etc** flag is set at the end of a transmission. **Note:** This feature is not available on Patriot RAM1.	0 = Current CRC value does not match remainder (default). 1 = Current calculated CRC value matches the expected result.
lrcok Bit 6	**Linear Redundancy Check Okay flag—**Used to indicate when the calculated 8-bit LRC value is zero value for the current input data stream. The value is calculated across all received characters from the point the **lrcen** bit is set in the CNTL register. The current LRC residual can be reset by three mechanisms: • Clear **lrcen** bit in CNTL register • Set **xmt_en** bit in ENABLE register • Automatically by hardware when **etc** flag is set at the end of a transmission. **Note:** This feature is not available on Patriot RAM1.	0 = Current LRC value does not match remainder (default). 1 = Current calculated LRC value matches the expected result (i.e. zero)
rdrf Bit 5	**Receive Data Register Full—**Used to indicate whether the SIM receive FIFO has reached the threshold level set by **rdt[4:0]** in the RCV_THRESHOLD register. The **rdrf** flag will be set any time the number of unread bytes in the receive FIFO is equal to or greater than the value set by **rdt[4:0]**. The flag can be cleared by reading enough bytes out of the receive FIFO so as to bring the number of bytes left in the FIFO below the **rdt[4:0]** level. Another way to clear the flag is to set the **rdt[4:0]** level higher than the number of unread bytes currently in the FIFO. The **rdrf** flag will create an interrupt if the **rim** bit in the INT_MASK register is cleared.	0 = Number of unread bytes in receive buffer < value set by **rdt[4:0]** (default). 1 = Number of unread bytes in receive buffer >= value set by **rdt[4:0]**.

Table 51-24. RCV_STATUS Description

Name	Description	Settings
rfd Bit 4	**Receive FIFO has unread Data—**Used to indicate that there is at least one unread byte in the receive data FIFO. Can only be cleared by reading all bytes out of the receive FIFO. The **rfd** bit cannot be used to create an interrupt. Normally, the SIM triggers the interrupt with **rdrf**, and software uses **rfd** to read all of the bytes out of the receive FIFO.	0 = There are no unread bytes in the receive FIFO (default). 1 = There is at least one unread byte in the receiver FIFO.
reserved Bits 3–1	**Unused bits**	N/A
oef Bit 0	**Overrun Error Flag—**Used to indicate that the SIM was unable to store received data due to already having 16 unread bytes in the FIFO. It does not necessarily indicate that data has been lost. If the **onack** control bit in the CNTL register is set, there will be a NACK pulse generated on bytes that would otherwise cause a loss of data due to a full FIFO. These bytes should be retransmitted by the SIM card which implies that no data has actually been lost. In this case, the **oef** flag is just an indicator that this situation has occurred which may be helpful in system debug. For the case where **onack** is not set, a set **oef** flag does indicate a loss of data since all bytes received with the **oef** flag set will indeed be lost (including the byte that caused the bit to be set). The **oef** flag will cause an interrupt if the **oim** bit in the INT_MASK register. The **oef** flag is a write-one-to-clear bit.	0 = No overrun error has occurred (default). 1 = A byte was received when the received FIFO was already full.

INT_MASK Interrupt Mask Register

Addr (SIM1) $2484_F01C (SIM2) $2A8A_001C

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				bgtm	bwtm	rtm	cwtm	gpcntm	tdtfm	tfom	xtm	tfeim	etcim	oim	tcim	rim
TYPE:	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 51-25. INT_MASK Description

Name	Description	Settings
bgtm Bit 12	**Block Guard Time Interrupt Mask — Used to enable / disable the ability of the bgt flag in the RCV_STATUS register to generate a SIM interrupt.**	0 = bgt interrupt enabled 1 = bgt interrupt masked
bwtm Bit 11	**Block Wait Time / Work Waiting Time Interrupt Mask — Used to enable / disable the ability of the bgt / wwt1 flag in the RCV_STATUS register to generate a SIM interrupt.**	0 = bwt / wwt1 interrupt enabled 1 = bwt / wwt1 interrupt masked
rtm Bit 10	Receive NACK Threshold Interrupt Mask—Used to enable/disable the ability of the **rte** flag in the RCV_STATUS register to generate SIM Interrupts. **Note:** This feature is not available in CDR3 (i.e., Patriot RAM1.x, RAM 2.x, ROM 1.x)	0 = rte interrupt enabled 1 = rte interrupt masked (default)
cwtm Bit 9	**Character Wait Time / Work Waiting Interrupt Mask—**Used to enable/disable the ability of the **cwt / wwt2** flag in the RCV_STATUS register to generate SIM interrupts. **Note:** This feature not available on Patriot RAM1.	0 = **cwt** / wwt2 interrupt enabled 1 = **cwt** / wwt2 interrupt masked (default)
gpcntm Bit 8	**General Purpose Counter Interrupt Mask—**Used to enable/disable the ability of the **gpcnt** flag in the XMT_STATUS register to generate SIM interrupts. **Note:** This feature not available on Patriot RAM1.	0 = **gpcnt** interrupt enabled 1 = **gpcnt** interrupt masked (default)
tdtfm Bit 7	**Transmit Data Threshold Interrupt Mask—**Used to enable/disable the ability of the **tdtf** flag in the XMT_STATUS register to generate SIM interrupts. **Note:** This feature not available on Patriot RAM1.	0 = **tdtf** interrupt enabled 1 = **tdtf** interrupt masked (default)

Table 51-25. INT_MASK Description

Name	Description	Settings
tfom Bit 6	**Transmit FIFO Overfill Error Interrupt Mask—**Used to enable/disable the ability of the **tfo** flag in the XMT_STATUS register to generate SIM interrupts. **Note:** Due to a SIM Errata, this bit acts as an enable rather than a mask for Patriot RAM1.	0 = **tfo** interrupt enabled 1 = **tfo** interrupt masked (default)
xtm Bit 5	**Transmit Threshold Interrupt Mask—**Used to enable/disable the ability of the **xte** flag in the XMT_STATUS register to generate SIM interrupts	0 = **xte** interrupt enabled 1 = **xte** interrupt masked (default)
tfeim Bit 4	**Transmit FIFO Empty Interrupt Mask—**Used to enable/disable the ability of the **tfe** flag in the XMT_STATUS register to generate SIM interrupts.	0 = **tfe** interrupt enabled. 1 = **tfe** interrupt masked (default)
etcim Bit 3	**Early Transmit Complete Interrupt Mask—**Used to enable/disable the ability of the **etc** flag in the RCV_STATUS register to generate SIM interrupts.	0 = **etc** interrupt enabled 1 = **etc** interrupt masked (default)
oim Bit 2	**Overrun Interrupt Mask—**Used to enable/disable the ability of the **oef** flag in the RCV_STATUS register to generate SIM interrupts	0 = **oef** interrupt enabled 1 = **oef** interrupt masked (default)
tcim Bit 1	**Transmit Complete Interrupt Mask—**Used to enable/disable the ability of the **tc** flag in the XMT_STATUS register to generate SIM interrupts.	0 = **tc** interrupt enabled. 1 = **tc** interrupt masked (default)
rim Bit 0	**Receive Interrupt Mask—**Used to enable/disable the ability of the **rdrf** flag in the RCV_STATUS register to generate SIM interrupts.	0 = **rdrf** interrupt enabled 1 = **rdrf** interrupt masked (default)

Note: The other interrupt masks in the SIM design are the **sdim0** bit in the PORT0_DETECT register, and the **sdim1** bit in the PORT1_DETECT register.

PORT0_XMT_BUF Port 0 Transmit Buffer Register

Addr (SIM1) $2484_F01E (SIM2) $2A8A_001E

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									port0_xmt[7:0]							
TYPE:	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 51-26. PORT0_XMT_BUF Description

Name	Description	Settings
port0_xmt[7:0] Bits 7–0	**Port 0 Transmit Buffer register—**Register to write when transmitting using port 0. To use this register to initiate transmissions to the SIM, set **sps** = 0. When **sps** = 1, a write to this register has no effect. A read of this register will produce the last thing written, or $00 if when **sps** = 1.	N/A

Note: Writing more data to the transmit FIFO than it can hold (16 bytes) will cause a transmit FIFO Overfill error.

Note: Patriot RAM1 has errata for the Transmit FIFO Overfill error. It does not function properly on RAM1.

PORT0_RCV_BUF Port 0 Receive Buffer Register **Addr $2484_F020**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						port0_cwt	port0_fe	port0_pe	port0_rcv[7:0]							
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 51-27. PORT0_RCV_BUF Description

Name	Description	Settings
port0_cwt Bit 10	**Port0 Character Wait / Work Waiting Time Violation Flag —** This bit indicates whether a character wait time (T=1 SIM cards), or a work waiting time (T=0 SIM cards) violation has occurred, between the last couple of consecutive bytes received.	0 = No violation occurred 1 = violation has occurred between the last two consecutive bytes received.
port0_fe Bit 9	**Port0 Frame Error flag—**The **port0_fe** flag indicates whether a frame error was detected during the reception of the corresponding byte read in the **port0__rcv** field. The **port0_fe** flag cannot create an interrupt. It need not be cleared since it will be overwritten by the next byte received into that location of the FIFO.	0 = Byte contains no framing error (default). 1 = Byte contains a framing error
port0_pe Bit 8	**Port0 Parity Error flag—**The **port0_pe** flag indicates whether a parity error was detected during the reception of the corresponding byte read in the **port0__rcv** field. The **port0_pe** flag cannot create an interrupt. It need not be cleared since it will be overwritten by the next byte received into that location of the FIFO. A parity error can create a NACK pulse. **Note:** Patriot RAM1 will not show a parity error on invalid initial characters during initial character mode. This errata is fixed for Patriot ROM1.	0 = Byte contains no parity error (default). 1 = Byte contains a parity error
port0_rcv[7:0] Bits 7–0	**Port0 Receive buffer—**Read from the next location in the receive buffer. Reads from this register return zero when the **sps** bit is zero	N/A

.

PORT0_DETECT Detect Register for Port 0

Addr (SIM1) $2484_F022 (SIM2) $2A8A_0022

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													spds0	spdp0	sdi0	sdim0
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	rw	r	w1c	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 51-28. PORT0_DETECT Description

Name	Description	Settings
spds0 Bit 3	**SIM Presence Detect Select Port 0—**Controls which edge of the SIMPD0 pin is used to generate the sdi0 interrupt.	0 = Falling edge of SIMPD0 Input (default). 1 = Rising edge of SIMPD0 Input
spdp0 Bit 2	**SIMPD0 input pin status—**This bit reflects the state of the SIMPD0 pin. It is not a latched register bit, but instead a synchronized version of the state of the SIMPD0 pin itself. **Note:** Patriot RAM1 requires two reads of this register to get current pin status due to errata.	0 = SIMPD0 pin is logic low 1 = SIMPD0 pin is logic high
sdi0 Bit 1	**SIM Detect Interrupt flag Port 0—**Status flag to indicate the insertion or removal of a SIM card has been detected on port 0. Can create an interrupt to the MCU if **sdim0** is low. Write a 1 to this bit to clear.	0 = No insertion or removal of SIM card detected on Port 0(default). 1 = Insertion or removal of SIM card detected on Port 0
sdim0 Bit 0	**SIM Detect Interrupt Mask Port 0—**Interrupt mask for the **sdi0** interrupt flag.	0 = **sdi0** enabled (default). 1 = **sdi0** masked

DATA_FORMAT Data Format Register

Addr (SIM1) $2484_F024 (SIM2) $2A8A_0024

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																ic
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rwm
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 51-29. DATA_FORMAT Description

Name	Description	Settings
ic Bit 0	**Inverse Convention—**Used to configure the SIM to use either inverse convention or direct convention for its data format. The **ic** bit can be controlled by software, but it is normally set by hardware as a result of the interpretation of the initial character when in ICM mode. There is a two reference clock cycle delay before the software can read that this bit is set after writing it.	0 = Direction convention transfers enabled (default). 1 = Inverse convention transfers enabled.

XMT_THRESHOLD Transmit Threshold Register **Addr (SIM1) $2484_F026 (SIM2) $2A8A_0026**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									xth[3:0]				tdt[3:0]			
TYPE:	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 51-30. XMT_THRESHOLD Description

Name	Description	Settings
xth[3:0] Bits 7–4	**Transmit NACK Threshold—**Used to set the NACK threshold for the transmitter. Once the threshold number set by **xth** has been reached for the current byte being transmitted, the error flag **xte** in the XMT_STATUS register will be set. Setting of **xte** causes the remaining transmissions queued in the transmit FIFO to be aborted and no more transmissions to occur until software clears **xte**. To trigger **xte**, a given byte being transmitted must reach the **xth** threshold itself. Transmit NACKs accumulated on one byte are not carried over to the next. **Note:** Patriot RAM1 has an errata in the transmit state machine that limits the **xth[3:0]** values to 0 or 1.	**xth** / **Transmitter Operation** $0 — **xte** will never be set; retransmission after NACK reception is disabled $1 — **xte** will be set after 1 nack is received; 0 retransmissions will occur. $2 — **xte** will be set after 2 nacks are received; At most 1 retransmission will occur. $3 — **xte** will be set after 3 nacks are received; At most 2 retransmissions will occur. ... — ... $X — **xte** will be set after X nacks are received; At most (X - 1) retransmissions will occur. ... — ... $F — **xte** will be set after 15 nacks are received; At most 14 retransmissions will occur.
tdt[3:0] Bits 3–0	**Transmit Data Threshold --** Used to set the threshold value for the Transmit FIFO at which the **tdtf** bit in the XMT_STATUS register will be set. When the number of bytes in the Transmit FIFO is less than or equal to **tdt[3:0]**, **tdtf** will be set. **Note:** This feature is not available on Patriot RAM1.	N/A

GUARD_CNTL Transmit Guard Control Register

Addr
(SIM1)
$2484_F028
(SIM2)
$2A8A_0028

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								RCVR11	getu[7:0]							
TYPE:	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 51-31. GUARD_CNTL Description

Name	Description	Settings
RCVR11 Bit 8	**Receiver use 11 ETUs—**Used to configure the SIM module receiver for 11 ETU operation (i.e. 1 Stop bit). This bit is provided for support of T=1 cards. Refer to Section 51.9.2.6 on page 51-87 for details on configuring the SIM receiver for T=1 operation.	0 = Receiver configured for 12 ETU operation (default). 1 = Receiver configured for 11 ETU operation.
getu[7:0] Bits 7–0	**Transmit Guard ETUs—**Used to control the number of additional elementary time units (ETUs) inserted between bytes transmitted by the SIM transmitter. An ETU is equivalent to one bit time at the given baud rate (e.g. the length of a start bit). The guard time has no effect on the SIM receiver. A value of $00 inserts no additional ETUs, while a value of $FE inserts 254 additional ETUs. A value of $FF subtracts one ETU by reducing the number of STOP bits from two to one. **Note:** Patriot RAM1 has an errata that does not allow for 11 ETU transmission. When **getu[7:0]** is set to $FF, 2 stop bit will be sent.	N/A

OD_CONFIG — Open-Drain Configuration Control Register — **Addr (SIM1) $2484_F02A (SIM2) $2A8A_002A**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															od_p1	od_p0
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 51-32. OD_CONFIG Description

Name	Description	Settings
od_p1 Bit 1	**Open Drain control for Port 1—**Used to control whether the XMT data line on port 1 is open-drain. If **amode** bit in SETUP register is set, this bit will have no effect. **Note:** Patriot RAM1 has an errata in that this bit is inverted. Port 1 Transmitter defaults to open-drain.	0 = XMT pin on port 1 is push-pull (default). 1 = XMT pin on port 1 is open-drain.
od_p0 Bit 0	**Open Drain control for Port 0—**Used to control whether the XMT data line on port 0 is open-drain	0 = XMT pin on port 0 is push-pull (default). 1 = XMT pin on port 0 is open-drain.

RESET_CNTL Reset Control Register **Addr (SIM1) $2484_F02C (SIM2) $2A8A_002C**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										DBUG	STOP	DOZE	kill clock	soft reset	flush xmt	flush rcv
TYPE:	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	slfclr	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 51-33. RESET_CNTL Description

Name	Description	Settings
DBUG Bit 6	**DEBUG —**Used to configure the operation of the SIM module when a debug event occurs. When set, a debug event (caused by OnCE module, external control, etc.) will cause the receive FIFO read pointer to be frozen. **Note:** This feature is not available on Patriot RAM1.	0 = Debug event has no affect on SIM module (default). 1 = Debug event prohibits read pointer changes for receive FIFO
STOP Bit 5	**STOP —**Used to configure the operation of the SIM module when a MCore STOP instruction is executed. This bit is added to provide support for SIM cards that do not allow the SIM Card clock to be stopped while power is applied. **Note:** This feature is not available on Patriot RAM1.	0 = STOP instruction shuts down all SIM clocks (default). 1 = STOP instruction shuts down all clocks except for the baud_clk (clock provided to SIM Card)
DOZE Bit 4	**DOZE —**Used to configure the operation of the SIM module when a MCore DOZE instruction is executed.	0 = DOZE instruction has no effect on SIM module (default). 1 = DOZE instruction will cause SIM module to gate SIM clocks when the transmit FIFO is empty.
kill_clock Bit 3	**Kill SIM Clock—**Used to enable/disable the SIM clock input to the SIM module. This bit will gate all SIM clocks including the SIM card clock regardless of the state of the STOP bit described above. **Errata**: See Section 51.1.3 for details.	0 = SIM input clock enabled (default). 1 = SIM input clock disabled.
soft_reset Bit 2	**Software Reset—**Used to reset the entire SIM module. This acts the same as a hardware reset for the SIM module. This bit is self-clearing. **Note:** Software should allow a minimum of 4 reference clock cycles (CKIH) before attempting to access the SIM module after a software reset.	0 = SIM Normal operation (default). 1 = SIM held in Reset.

Table 51-33. RESET_CNTL Description

Name	Description	Settings
flush_xmt Bit 1	**Flush Transmitter—**This bit operates as a SIM transmitter reset. The receive portion of the SIM module is not affected. The software must clear this bit before the SIM transmitter can operate.	0 = SIM Transmitter normal operation (default). 1 = SIM Transmitter held in Reset
flush_rcv Bit 0	**Flush Receiver—**This bit operates as a SIM receiver reset. The transmit portion of the SIM module is not affected. The software must clear this bit before the SIM receiver can operate.	0 = SIM Receiver normal operation (default). 1 = SIM Receiver held in Reset

CHAR_WAIT Character Wait Time Register

Addr (SIM1) $2484_F02E (SIM2) $2A8A_002E

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Character Wait Time [15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 51-34. CHAR_WAIT Description

Name	Description	Settings
CWT Bits 15-0	**Character Wait Time—**The value written to this register will specify the number of ETU times allowed between characters. **Note:** This feature is not available on Patriot RAM1. **Note:** This register should not be used to write new code; CHAR_WAIT_ACC register should be used instead. This register could only be used to overcome software incompatibilities with the previous SIM designs, by programming this register to the actual value minus 11(etus).	N/A

GPCNT General Purpose Counter Register

Addr (SIM1) $2484_F030 (SIM2) $2A8A_0030

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	General Purpose Counter [15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 51-35. GPCNT Description

Name	Description	Settings
GPCNT Bits 15-0	**General Purpose Counter—**The value written to this register will be used to compare to the general purpose counter in the SIM module. Once the General purpose counter reaches this value, the **gpcnt** flag in the XMT_STATUS register will be set. This counter is intended to be used for any events that must be monitored for duration based on the card clock, receiver sample rate, or ETU rate (transmit clock). (ex. ATR arrival time, ATR duration, etc....) Refer to Section 51.6.6.4 for more details on how the General Purpose Counter is enabled. **Note:** This feature is not available on Patriot RAM1.	N/A

DIVISOR Programmable Divisor Register **Addr (SIM1) $2484_F032 (SIM2) $2A8A_0032**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE:	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Table 51-36. DIVISOR Description

Name	Description	Settings
DIVISOR Bits 6-0	**Divisor Register—**The value written to this register will be used to generate the SIM Receive Clock. The **baud_sel[2:0]** bits in the CNTL register must be set to '111' in order to control the divisor value using the DIVISOR register. In this mode, the SIM Receive clock is divided by 16 to generate the SIM Transmit clock (i.e. Baud rate clock). There is no support for 12X oversampling when using the DIVISOR register. The oversampling is fixed at 16X. **Note:** This feature is not available on Patriot RAM1.	N/A

BLOCK_WAIT_MSB Block Wait Time Register [31:16] **Addr (SIM1) $2484_F034 (SIM2) $2A8A_0034**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Block Wait Time [31:16]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 51-37. BLOCK_WAIT_MSB Description

Name	Description	Settings
BWT Bits 31-16	**Block Wait Time—**The value written to this register will specify the number of ETU times allowed between the last character transmitted and the first character received by the SIM module. See Example 51.6.10.2.2 for details. **Note:** This feature is not available on Patriot RAM1.	N/A

BLOCK_WAIT_LSB — Block Wait Time Register [15:0]

Addr (SIM1) $2484_F036 (SIM2) $2A8A_0036

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Block Wait Time [15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 51-38. BLOCK_WAIT_LSB Description

Name	Description	Settings
BWT Bits 15-0	**Block Wait Time —** The value written to this register will specify the number of ETU times allowed between the last character transmitted and the first character received by the SIM module. See Example 51.6.10.2.2 for details. **Note:** This feature is not available on Patriot RAM1.	N/A

CHAR_WAIT_ACC — Character Wait Time [15:0]

Addr (SIM1) $2484_F038 (SIM2) $2A8A_0038

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Block Wait Time [15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Name	Description	Settings
CWT Bits 15-0	**Character Wait Time —**The value written to this register will specify the number of ETU times allowed between consecutively received characters. **Note:** This feature is not available on Patriot RAM1. **Note:** This register should be used for all new code. Do not use **CHAR_WAIT** register when writing new software	N/A

BLOCK_GUARD Block Guard Time [7:0] **Addr (SIM1) $2484_F03A (SIM2) $2A8A_003A**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									Block Guard Time [7:0]							
TYPE:	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Name	Description	Settings
BGT Bits 7-0	**Block Guard Time —** The value written to this register will specify the number of ETU times allowed between characters sent in opposite directions. This feature is not available on Patriot RAM1.	N/A

51.9 Programming Model

This section describes the intended programming model for using the SIM module. The section describes how to begin a typical mode of operation using the register descriptions provided in the Section 51.8.

51.9.1 Configuring the SIM for Operation

The following is a list of items that need to be performed in order to configure the SIM module for operation:

1. Port selection in the SETUP register (see page 51-50).
   a. Select which SIM module port is active by writing the **sps** bit.
   b. Select whether the second SIM module port is active at the same time under the control of an alternate SIM module by writing the **amode** bit.
2. Enable the selected port (for port 1, see PORT1_CNTL register on page 51-48; for port 0, see PORT0_CNTL on page 51-54) following the SIM power up procedure specified in ISO 7816.
   a. Enable power to the SIM card using the **sven1** or **sven0** bit.
   b. Enable SIM module transmit data output using the **sten1** or **sten0** bit. This is required to allow the SIM receiver to create NACK pulses.
   c. Enable SIM card clock using the **scen1** or **scen0** bit**.**
   d. Release the SIM card reset using the **srst1** or **srst0** bit (assert srstX high)**.**
3. Select XMT port driver type (open-drain or push-pull) in the OD_CONFIG register (see page 51-74).

a. Configure the selected port to be open drain or push-pull by using the **od_p1** or **od_p0** bit.

4. Choose the port Baud rate in the CNTL register (see page 51-56).
   a. Select the SIM card clock rate by using the **clk_sel[1:0]** bits
   b. Select the SIM card baud rate by using the **baud_sel[2:0]** bits
   c. If **baud_sel[2:0]** is set to '111', program the divisor to use for Receive clock generation in the DIVISOR register (see page 51-79).

Note: Follow the ISO 7816 spec with regards to card clock frequencies to ensure the maximum frequency specification is not violated.

5. Select Data format type, or place SIM receiver in initial character mode.
   a. Select inverse convention or direct convention by using the **ic** bit in the DATA_FORMAT register (see page 51-71), or
   b. Enable initial character mode by using the icm bit in the CNTL register (see page 51-56)

51.9.1.1 Configuring the SIM Receiver

The following is a list of items that need to be performed in order to configure the SIM receiver for operation:

1. Enable NACK capability in CNTL register (see page 51-56).
   a. Select NACK generation on parity errors, or invalid initial character by using the **anack** bit.
   b. Select NACK generation on overrun conditions by using the **onack** bit.
2. Select the desired Receive NACK Threshold and Receive FIFO threshold values in RCV_THRESHOLD register (see page 51-59)
   a. Program the threshold at which the **rte** flag will be set by using the **rth[3:0]** bits. If an auto power down after **rte** flag is desired, enable the SIM card auto power down by using **sapdX** bit in the PORTx_CNTL register.
   b. Program the threshold at which the **rdrf** flag will be set by using the **rdt[4:0]** bits
3. Configure the Character Wait Time Counter
   a. Program Character Wait Time interval in CHAR_WAIT register
   b. Enable Character Wait Time Counter by setting **cwten** bit in CNTL register
4. Enable interrupts in the INT_MASK register (see page 51-66).
   a. Enable the receive data register full interrupt by using the **rim** bit

b. Enable the receive threshold error interrupt by using the **rtm** bit.
c. Enable the overrun condition interrupt by using the **oim** bit
d. Enable the Character Wait Time interrupt by using the **cwtm** bit

51.9.1.2 Configuring the SIM Transmitter

The following is a list of items that need to be performed in order to configure the SIM transmitter for operation:

1. Select desired re-transmission threshold for NACKed characters in XMT_THRESHOLD register (see page 51-72)
   a. Program the threshold at which the **xte** flag will be set by using the **xth[3:0]** bits
2. Select the guard time between transmissions in the GUARD_CNTL register (see page 51-73)
   a. Program the desired guard time between characters transmitted by the SIM module by using the **getu[7:0]** bits
3. Select the desired Transmit FIFO threshold level in the XMT_THRESHOLD register (see page 51-72)
   a. Program the desired threshold using the **tdt[3:0]** bits
4. Enable interrupts in the INT_MASK register (see page 51-66)
   a. Enable the transmit complete interrupt by using the **tcim** bit
   b. Enable the early transmit complete interrupt by using the **etcim** bit
   c. Enable the transmit FIFO empty interrupt by using the **tfeim** bit
   d. Enable the transmit threshold error interrupt by using the **xtm** bit
   e. Enable the transmit FIFO threshold interrupt by using the **tdtfm** bit
   f. Enable the transmit FIFO overfill interrupt by using the **tfom** bit

51.9.1.3 Configuring the SIM General Purpose Counter

The following is a list of items that need to be performed in order to configure the SIM General Purpose Counter for operation:

1. Enable the selected clock source for the General Purpose Counter using either the RESET_CNTL (see page 51-75) or CNTL and ENABLE registers (see page 51-56 and page 51-60)
   a. If the GP Counter will be configured for the card clock, enable the clock by clearing the **kill_clock** bit in the RESET_CNTL register.
   b. If the GP Counter will be configured for the receive oversample clock, enable this clock by setting the **rcv_en** bit or **xmt_en** bit in the ENABLE register.
   c. If the GP Counter will be configured for the ETU (transmit) clock, enable this clock by setting the **xmt_en** bit in the ENABLE register, or setting the **rcv_en** bit in the ENABLE register and the **cwten** bit in the CNTL register.
2. Program counter comparator using the GPCNT register (see page 51-78)

a. Use the **general_purpose_counter[15:0]** bits to select the desired count value at which the **gpcnt** interrupt flag will be set.

3. Enable interrupts in the INT_MASK register (see page 51-66)
   a. Enable the general purpose counter interrupt by using the **gpcntm** bit
4. Select desired clock source for the General Purpose Counter using the CNTL register (see page 51-56)
   a. Use the **gpcnt_clksel[1:0]** bits to select the desired clock source for the counter

51.9.1.4 Configuring the SIM Linear Redundancy Check (LRC) Block

The following is a list of items that need to be performed in order to configure the SIM Linear Redundancy Check Block for operation:

1. Enable the LRC block by using the CNTL register (see page 51-56)
   a. Use the **lrcen** bit to enable the LRC block
   b. Use the **xmt_en_LRC_CRC** bit to enable the transmission of the LRC Character after the last character in the Transmit FIFO is sent. Refer to the T=1 programming model for more details (Section 51.9.3.5.4 on page 51-94).

51.9.1.5 Configuring the SIM Cyclic Redundancy Check (CRC) Block

The following is a list of items that need to be performed in order to configure the SIM Cyclic Redundancy Check Block for operation:

1. Enable the CRC block by using the CNTL register (see page 51-56)
   a. Use the **crcen** bit to enable the CRC block
   b. Use the **xmt_en_LRC_CRC** bit to enable the transmission of the CRC Characters after the last character in the Transmit FIFO is sent. Refer to the T=1 Programming model for more details (Section 51.9.3.5 on page 51-90).

51.9.2 Using the SIM Receiver

Once the SIM has been properly configured (correct baud rate, correct data format, etc.), SIM receptions can be enabled by setting the receive enable bit, **rcv_en**, in the ENABLE register (see page 51-60). As bytes are received, they will be placed in the 32 byte deep receive data FIFO. Unread bytes can be accessed from this FIFO at any time. There is no need to disable the receiver to access the FIFO. The FIFO should only be read when the receive FIFO data flag, **rfd**, in the RCV_STATUS register (see page 51-63) is set. The **rfd** flag, which cannot create an interrupt, is high any time there is at least one unread byte in the receive FIFO. If the receive FIFO is read when **rfd** is low, it will simply produce the last byte read.

The correct address to read the data contained in the receive FIFO depends on which SIM port is selected. The **sps** control bit in the SETUP register (see page 51-50) controls port selection. If **sps** = 0, read the data from the PORT0_RCV_BUF register; if **sps** = 1, read the data from the PORT1_RCV_BUF register.

The receive data register full flag, **rdrf**, in the RCV_STATUS register (see page 51-63) can be used to determine when the receive FIFO has reached a given threshold value. This flag will create an interrupt if the **rim** bit in the INT_MASK register (see page 51-66) is clear. To control at which point **rdrf** is set,

program the receive data threshold, **rdt[4:0],** in the RCV_THRESHOLD register (see page 51-59). If the number of unread bytes in the receive FIFO is equal to or greater than the value set by **rdt[4:0]**, **rdrf** will be set.

Note: A value of $0 in **rdt[4:0]** implies that there must be 32 unread bytes in the receive FIFO to trigger **rdrf**.

The value in **rdt[4:0]** can be changed at any time to alter this threshold level. The comparison between the number of unread bytes in the FIFO and the value set by **rdt[4:0]** is continuously updated so that any change in either will be immediately reflected in the state of **rdrf**. For instance, if **rdt[4:0]** is set to 5, and there are 3 unread bytes in the FIFO, changing **rdt[4:0]** to 2 will immediately cause **rdrf** to be set. Likewise, setting **rdt[4:0]** back to 5 will cause **rdrf** to clear. Similarly, if there are 5 unread bytes in the receive FIFO and **rdt[4:0]** is set to 3, **rdrf** will remain high until 3 reads are complete, assuming **rdt[4:0]** is left as a constant and no new data is received.

The standard flow for receiving bytes from the SIM card is to set **rdt[4:0]** to the appropriate value, wait for **rdrf** to cause an interrupt (**rim** clear), and then read bytes out of the receive FIFO as long as **rfd** is high. In addition to checking **rfd** between every byte, it is also recommended that software check for the existence of a set **oef** flag as well.

51.9.2.1 Receive Parity Errors and Parity NACK Generation

The SIM receiver checks every byte received for proper parity. The **ic** control bit in the DATA_FORMAT register (see page 51-71) controls whether it checks for odd parity or even parity. When checking for odd parity, the number of logic ones contained in the 9 received bits (8 data bits and 1 parity bit) should be odd. Likewise, when checking for even parity, the number of logic ones contained in the 9 received bits (8 data bits and 1 parity bit) should be even.

When a parity error is detected on a given byte, the **portX_pe** bit for that byte is set in the FIFO. The **portX_pe** flag for each byte is read out of the FIFO when the data itself is read. There is no need to attempt to clear the parity error flag in the FIFO. It is simply overwritten the next time a byte is received into that position of the FIFO. A parity error cannot cause an interrupt.

It is possible for the SIM to automatically request that the SIM card re-send a byte found to have a parity error by generating a NACK pulse on the SIM XMT pin. The SIM will generate a NACK pulse on a byte received with a parity error if the control bit **anack** in the CNTL register (see page 51-56) is set. Bytes with parity errors that cause a NACK pulse are still placed into the FIFO just as bytes that do not cause a NACK pulse are. It is up to the software to discard data bytes with parity errors.

To control NACK generation by the SIM receiver use **rth[3:0]**, Receive NACK Threshold, in the RCV_THRESHOLD register. This set of bits specify the number of consecutive NACKs generated by the SIM module, on a received byte, before generating an **rte** (receive threshold interrupt flag) in the RCV_STATUS register. The **rte** flag would also force the SIM port to power down the card if the **sapdX** bit is set in the PORTx_CNTL register.

Note: The **anack** bit must be set in the CNTL register to enable this feature. This bit is also used in initial character mode to enable the retransmission of initial characters in the event that an invalid initial character is received.

When a valid character has been received by the SIM, the internal counter keeping track of the number of NACKs transmitted on the current byte resets to zero. Clearing the receive threshold error (**rte**) bit would also clear that counter.

When generating a NACK pulse, the SIM will generate the low pulse starting at 10.5 ETUs and lasting for 1 ETU (see Figure 51-11 on page 51-31).

51.9.2.2 Receive Frame Errors

The SIM receiver checks every byte received for a proper stop bit. A stop bit should exist during at least the first half of the 11th ETU time after the start of the character. If this is not true, a frame error is flagged. When a frame error is detected on a given byte, the **portX_fe** bit for that byte is set in the FIFO. The **portX_fe** flag for each byte is read out of the FIFO when the data itself is read. There is no need to clear the frame error flag in the FIFO. It is simply overwritten the next time a byte is received into that position of the FIFO. A frame error cannot cause an interrupt, nor can it create a NACK pulse to the receiver asking for a retransmission of the corrupted data.

51.9.2.3 Receive Overrun Errors and Overrun NACK Generation

When there already exists 32 unread bytes in the FIFO, a received character will cause the SIM receiver to flag an overrun condition. This condition will always set the overrun error flag, **oef** in the RCV_STATUS register (see page 51-63). The received byte will be discarded leaving the 32 unread bytes in the FIFO unaltered.

It is possible for the SIM to automatically request that the SIM card re-send the byte that caused the overrun condition by generating a NACK pulse on the SIM XMT pin. The SIM will generate a NACK pulse for the byte that caused the overrun condition if the control bit **onack** in the CNTL register (see page 51-56) is set. In this case, the existence of an **oef** flag does not indicate the loss of data, but rather a NACK (i.e. retransmission request) due to a full receive FIFO. As opposed to transmit NACK generation, there is no limit to the number of times an overrun condition will cause a NACK other than to disable **onack** itself. When generating a NACK pulse, the SIM will generate the low pulse starting at 10.5 ETUs and lasting for 1 ETU (see Figure 51-11 on page 51-31).

If the control bit **onack** is clear, the existence of a high **oef** flag indicates the loss of data. The **oef** flag can create an interrupt if the **oim** bit in the INT_MASK register (see page 51-66) is clear.

To clear the **oef** flag, software must simply write a “one” to the **oef** bit position in the RCV_STATUS register. A high **oef** flag has no effect on the operation of the SIM receiver other than to create an interrupt if **oim** is clear.

51.9.2.4 Using Initial Character Mode and Resulting Receive Data Formats

The SIM receiver supports the detection of special characters that allow it to determine what data format is being used by the connected SIM card. When placed in initial character mode, the SIM expects to receive one of two potential values that it will use to set the data format control bit, **ic**, in the DATA_FORMAT register (see page 51-71).

The two possible data formats are inverse convention and direct convention. Figure 51-18 illustrates the differences between the two formats. Essentially, inverse convention differs from direct convention in that the order of the data is flipped msb for lsb, and the data bits and parity bit are logically inverted. When receiving inverse convention data, the transformation of the data back to direct convention format is done in hardware, including the inversion of the data and parity bits.

To place the SIM into initial character mode, set the **icm** bit in the CNTL register (see page 51-56). Once a valid initial character is received, the **ic** bit in the DATA_FORMAT register (see page 51-71) will be set accordingly by the hardware, and the **icm** bit will be cleared. Software can read the state of the **ic** bit to determine which mode the SIM is currently using.

Figure 51-17 on page 51-35 shows the two possible valid initial characters that can be received. The $3B (as decoded by direct convention) with parity bit high will cause direct convention to be used (**ic** set to logic 0), whereas a $3F (as decoded by inverse convention) with parity bit high will cause inverse convention to be used (**ic** set to logic 1).

When the receiver is in initial character mode, all received bytes will continue to be placed into the receive FIFO whether they be valid initial characters or not. If a valid initial character is received that causes the data format being used to change, all subsequent bytes will be decoded with that format before being placed into the FIFO, including the initial character byte itself. That is, if the **ic** bit is low, and the correct initial character for setting inverse convention is received, that character and all subsequently received characters will be stored in the FIFO after having been decoded using inverse convention (e.g. the initial character will be stored as $3F).

If the receiver is in initial character mode (**icm** is high), and an invalid initial character is received, the SIM can be configured to automatically request that the initial character be retransmitted by setting the **anack** control bit in the CNTL register (see page 51-56). When generating a NACK pulse, the SIM will generate the low pulse starting at 10.5 ETUs and lasting for 1 ETUs (see Figure 51-11 on page 51-31). The invalid initial character will be placed into the receive FIFO and marked with a parity error to signify that this is an invalid initial character.

51.9.2.4.1 Initial Character Mode Programming Notes

The usage of the initial character mode requires close attention to the programming model. There is a condition where a parity error in the initial byte for direct convention ($3B) could be decoded as what appears to be a valid initial character for inverse convention (i.e. $3F). The SIM module will not recognize this as a valid initial character for inverse convention and will mark the character by setting the parity error flag. The software must look for the existence of a parity error before recognizing a character as a valid initial character.

Note: Patriot RAM1 has an errata such that the parity flag is not set on invalid initial characters.

51.9.2.5 Automatic Receiver Mode

The SIM module has an automatic receive mode that inhibits the data being transmitted by the SIM module from entering the SIM receive buffer through the feedback path of the SIM data pin. The SIM module receiver should normally be enabled while the transmitter is operational. Automatic receive mode saves the software from having to actively manage the transition from transmitter to receiver. The auto receive mode is always active whenever the receiver is enabled.

51.9.2.6 Using the SIM Receiver with “T=1” SIM Cards

The SIM module provides hardware support for “T=1” type SIM cards. These type of cards present several requirements above and beyond the standard “T=0” cards. The features provided to meet the requirements that pertain to the SIM receiver are as follows:

- 11 ETU Characters
  - “T=1” cards can transmit with character lengths of 11 ETUs (i.e. one STOP bit). The SIM module provides the **rcvr11** bit in the GUARD_CNTL register in order to configure the receiver state machine to accept 11 ETU characters.
- Character Wait Time Counter
  - The character waiting time (CWT) is defined as the time between the start bits of two consecutive characters. The value of CWT can range from 12 ETU to 32779 ETU. The SIM module provides a 16-bit counter with programmable comparator clocked at the ETU bit rate to identify when the CWT has been exceeded by the SIM card.

Note: The CWT counter is not available on Patriot RAM1.

- Block Waiting Time
  - The block waiting time (BWT) is defined as the time between the start bits of the last character of a received block and the first character of the next received block. The value of BWT is always greater than 1800 ETU. The SIM module provides a 16-bit General Purpose Counter that can be used to identify when the BWT has been exceeded by the SIM card. For detecting BWT violations, the General Purpose Counter can be configured to be clocked at the ETU bit rate. A 16-bit comparator is provided to generate an interrupt when the BWT has expired.
- Block Guard Time
  - The block guard time (BGT) is defined as the minimum delay between the leading edges of two consecutive characters sent in opposite directions. The value of BGT is 22 ETU. The SIM module supports the BGT by providing the ability to generate an interrupt when the last byte is received, and transmitting within 2 ETU after the **xmt_en** bit is set. The BGT will be determined by the speed at which the software can react to an interrupt and enable the transmitter. Again, the General Purpose Counter can be used to guarantee that the software does not violate the BGT.
- Error Detection Code
  - "T=1" cards can specify LRC or CRC error detection codes to be used. The SIM module provides hardware support for both the LRC and CRC operations.

51.9.3 Using the SIM Transmitter

Once the SIM has been properly configured (data XMT enabled, correct baud rate, correct data format, etc.), the transmitter can be enabled by setting the **xmt_en** bit in the ENABLE register. If data had been previously written to the transmit FIFO, the transmitter will begin to send the first character. If no data is written to the transmit FIFO before enabling the transmitter, then the transmitter will wait until the first character is written before beginning transmission. Clearing the **xmt_en** bit while the transmitter is in operation, will halt any transmission in progress, flush the transmit FIFO, and reset the transmit state machine.

Data can be written to the transmit FIFO at any time.

The transmit data threshold flag, **tdtf**, in the XMT_STATUS register (see page 51-61) can be used to determine when the transmit FIFO has reached a given threshold value. This flag will create an interrupt if the **tdtfm** bit in the INT_MASK register (see page 51-66) is clear. To control at which point **tdtf** is set, program the transmit data threshold, **tdt[3:0],** in the XMT_THRESHOLD register (see page 51-72). If the number of bytes remaining in the transmit FIFO is equal to or less than the value set by **tdt[4:0]**, **tdtf** will be set.

Note: A value of $0 in **tdt[3:0]** implies that the transmit FIFO must be empty to trigger **tdtf**.

The value in **tdt[3:0]** can be changed at any time to alter this threshold level. The comparison between the number of remaining bytes in the transmit FIFO and the value set by **tdt[3:0]** is continuously updated so that any change in either will be immediately reflected in the state of **tdtf**. Unlike the **rdrf** flag for the receive FIFO, **tdtf** is latched an remains set until the software writes a one to the **tdtf** bit location in the XMT_STATUS register. For instance, if **tdt[3:0]** is set to 5, and there are 6 bytes remaining in the FIFO, changing **tdt[3:0]** to 6 will immediately cause **tdtf** to be set. However, setting **tdt[3:0]** back to 5 will not cause **tdtf** to clear.

The standard flow for transmitting bytes from the SIM card is to set **tdt[3:0]** to the appropriate value, write up to 16 bytes to the transmit FIFO, enable the transmitter, wait for **tdtf** to cause an interrupt (**tdtfm** clear), and then write additional bytes to the transmit FIFO.

NOTE:

For the SIM module to transmit successfully, the transmit pin must be connected to the receive pin of the same port. This connection is necessary for the SIM to decode transmit NACKs sent to it by the SIM card. Without this connection, all transmissions will appear to have a NACK thereby causing the first byte to be transmitted continuously. When operating in 3volt mode (**3volt0** or **3volt1**), this connection is made internal to the Neptune LTE IC.

51.9.3.1 Transmit Data Formats

There are two possible data formats that the SIM module will use when transferring data to the SIM card--inverse convention or direct convention (see Figure 51-12 on page 51-31). The format used depends on the state of inverse convention bit (**ic** in the DATA_FORMAT register). Software can set the **ic** bit, or it can be set automatically by hardware when using the initial character mode.

51.9.3.2 Transmit NACKs

The SIM transmitter can respond to NACKs created by the SIM card. A NACK will be decoded if the SIM card creates a logic low level on the SIM receive pin during the STOP bit time at the end of a transmitted byte (see Figure 51-11 on page 51-31). To prevent a situation where the SIM interface is stalled by an infinite number of NACK pulses on a given byte, the SIM module can be configured to limit the number of times it will respond to NACKs. The **xth[3:0]** control field in the XMT_THRESHOLD register (see page 51-72) allows software to set a threshold on the number of times a given byte will be retransmitted. If the threshold is reached, a transmit threshold error, **xte** in the XMT_STATUS register (see page 51-61), is asserted.

When **xte** is set, the SIM transmitter is halted, and all pending transfers are aborted, and the **tc, etc,** and **tfe** flags are set. All bytes remaining in the transmit FIFO will be lost. There is no way to restart the transmission on the next byte in the FIFO. The transmitter will remain frozen until **xte** is cleared by software. The only way to clear **xte** is to write a 1 to the **xte** bit in the XMT_THRESHOLD register (see page 51-72). The **xte** flag can create an interrupt if the **xtm** mask in the XMT_THRESHOLD register is clear.

It is possible to disable the detection of NACKs from the SIM card by setting **xth[3:0]** to $0. By setting **xth[3:0]** to $1, it is possible to disable all retransmissions while still setting **xte** on the first NACK received. In general, **xte** is set on the NACK that causes the threshold set by **xth[3:0]** to be reached. This final NACK will not cause a retransmission, whereas all previous NACKs will cause a retransmission. Table 51-30 on page 51-72 summarizes the effect that **xth[3:0]** will have on the operation of the SIM transmitter.

51.9.3.3 Transmit Guard Time

The time between data bytes sent from the SIM transmitter can be altered using the transmit guard time control. By default, the minimum time between start bits of successive transmitted bytes is 12 ETUs (Elementary Time Units). An ETU is equivalent in time to 1 bit time. Therefore, the amount of time an ETU consumes is determined by the baud rate chosen. The 12 ETUs that form the default character transmission time consist of a start bit, 8 data bits, a parity bit and two stop bits (see Figure 51-10 on page 51-30). The number of stop bits (idle bits) can be extended by an integer number of ETUs. The number of additional ETUs can be programmed directly into the **getu[7:0]** bits of the GUARD_CNTL register (see page 51-73). Programming the **getu[7:0]** bits to $FF will configure the SIM transmitter to use only one stop bit for each character transmission.

51.9.3.4 Using the SIM Transmit with “T=1” SIM Cards

The SIM module provides hardware support for “T=1” type SIM cards. These type of cards present several requirements above and beyond the standard “T=0” cards. The features provided to meet the requirements that pertain to the SIM transmitter are as follows:

- 11 ETU Characters
  - The SIM module transmitter has a programmable guard time register that allows the programmer to specify the number of ETUs between character transmissions. Programming a value of 255 ($FF) in the **getu[7:0]** bits in the GUARD_CNTL register will set the number of ETUs per character transmitted to 11.
- Character Waiting Time
  - The character waiting time (CWT) is defined as the time between the start bits of two consecutive characters. The value of CWT can range from 12 ETU to 32779 ETU. The time between transmitted characters is controlled by the programmable guard time in the GUARD_CNTL register. However, the time between the last byte in the transmit FIFO, and the next transmitted byte can be largely affected by software response time to the transmit interrupts. The SIM transmitter provides a Transmit FIFO threshold (**tdtf**) interrupt to signal the system when the expected number of characters have been transmitted from the transmit FIFO. The minimum CWT is achieved only if the software can respond to the **tdtf** interrupt and write new data to the transmit FIFO before the last character in the Transmit FIFO has been sent.
- Block Waiting Time
  - See Example 51.6.10.2.2 for Block Waiting Time definition. The value of BWT is always greater than 1800 ETU. The SIM supports the BWT by providing an interrupt when the number of etu cycles become greater than equal to the value in the bwt[31:0] register.
- Block Guard Time
  - The block guard time (BGT) is defined as the minimum delay between the leading edges of two consecutive characters sent in opposite directions. The value of BGT is 22 ETU. The SIM module supports the BGT by providing the ability to generate an interrupt whenever the etu time becomes greater than or equal to or greater than the value programmed in bgt[7:0].
- Error Detection Code
  - “T=1” cards can specify LRC or CRC error detection codes to be used. The SIM module provides hardware support for both the LRC and CRC operation.

51.9.3.5 Suggested “T=1” Compliant Programming Model

This section will describe the suggested programming model for supporting “T=1”, “T=0”, and known “special” cards using the SIM module on the Patriot IC. This should be used as a rough guide for how to configure the SIM module for use with SIM cards specified by ISO 7816-3 and EMV. Some details are not addressed. Other uses for some of the SIM features are not included (ex. GP Counter uses for some ISO timing requirements).

51.9.3.5.1 Answer To Reset (ATR) Detection

The first step to communicating with a SIM card is to provide power and a clock signal to the card. Once the card is detected as present (using the presence detect features, or some other method), the SIM card should be powered up according to the power up sequence specified in the ISO 7816-3 specification.

1. Apply voltage to the SIM card by setting the **sven0** or **sven1** bit in the PORTx_CNTL register
2. Select the appropriate clock frequency for the SIM card by programming the **clk_sel[1:0]** bits in the CNTL register.
3. Enable the clock to the SIM card by setting the **scen0** or **scen1** bit in the PORTx_CNTL register
4. Remove the card from reset by setting the **srst0** or **srst1** bit in the PORTx_CNTL register

The first communication between the SIM card and the SIM module will be a block of data sent from the SIM card to the SIM module after the card is powered and the card reset is removed. This block is call the Answer To Reset (ATR). In order to receive the ATR, the SIM module should be configured for 12 ETU character reception. According to the ISO 7816-3 spec, both “T=0” and “T=1” cards will communicate initially using 12 ETU character durations.

Note: We are aware of some card manufacturers that communicate at 11.5 ETU character durations (Geldkate). This will complicate the initial card detection sequence shown below.

5. Clear **rcvr11** bit in the GUARD_CNTL register

The next item to configure for ATR reception is the NACK capability of the SIM module. The ISO 7816-3 spec allows the SIM module to NACK any communication errors that occur during the initial communication at 12 ETU.

Note: The Europay Mastercard and VISA (EMV) cards are similar to “T=1”, but do not allow the SIM module to NACK during the initial communication. This will again complicate the initial card detection sequence shown below.

6. Set **anack** bit in the CNTL register to enable NACK generation

In order for the SIM module to notify when characters are received, the receive interrupt(s) can be enabled. A threshold can be set for the number of characters to receive before generating an interrupt.

7. Enable **rdrf** and **oef** interrupts by setting **rim** and **oim** bits in INT_MASK register
8. Set desired threshold for received characters by writing **rdt[4:0]** in RCV_THRESHOLD register

The SIM should be setup to perform in initial character mode. This will cause the hardware to identify the first valid character sent during the ATR as an initial character. This character will automatically configure the hardware for the data convention used by the SIM card.

9. Set Initial Character Mode by setting the **icm** bit in the CNTL register

The ISO 7816-3 spec requires that SIM cards meet certain timing restrictions. One of these is the time from the deassertion of the card reset to the beginning of the ATR sequence. The SIM module General Purpose Counter can be used to verify that the SIM card begins its ATR within the 400 to 40,000 clock cycle range.

10. Set General Purpose Counter Comparator to $9C40 using GPCNT register

11. Enable General Purpose Counter Interrupt by clearing **gpcntm** in INT_MASK register
12. Enable General Purpose Counter by programming the **gpcnt_clksel[1:0]** bits to 01 so the card clock is used for counting

The ISO7816-3 spec states that the maximum allowed time between two characters during the ATR is 9600 ETUs (Initial Waiting Time). The Character Wait Time Counter should be setup to detect any errors for this condition.

13. Set Character Wait Time Counter Comparator to $2574 (9600 - 12) using the CHAR_WAIT register
14. Enable the Character Wait Time Counter Interrupt by clearing **cwtm** in INT_MASK register
15. Enable the Character Wait Time Counter by setting the **cwten** bit in the CNTL register

The last step in preparing for ATR reception is to enable the receiver.

16. Set **rcv_en** bit in ENABLE register

The SIM module will generate interrupts once a threshold number of characters is received. The software should react to these interrupts and read the characters from the receive FIFO (PORTx_RCV_BUF) until the complete ATR has been received. If a General Purpose Counter interrupt occurs before the final ATR character is received, then the card should be deactivated according to the ISO 7816-3 spec. Otherwise, once a valid ATR is received, the software will know from the ATR information the specific characteristics for this card (refer to the ISO 7816-3 spec for details).

51.9.3.5.2 Programming Considerations for Geldkate Cards

There is at least one manufacturer of SIM cards (Geldkate) that we know of that does not send the ATR in 12 ETU mode or support NACKs. This creates an issue with detecting a valid ATR from a SIM card. Since any kind of card can be attached to the SIM module, the software and hardware do not know how to begin communication. The suggested flow for this type of card is shown in Figure 51-27 on page 51-93. Basically, if the card fails to send a valid ATR on the first try, disable NACKs, configure the SIM module for 11 ETU and try again. The software should toggle between 12 and 11 ETU modes with and without NACKs enabled until a valid ATR is received, or the number of attempts to communicate passes a predetermined error threshold.

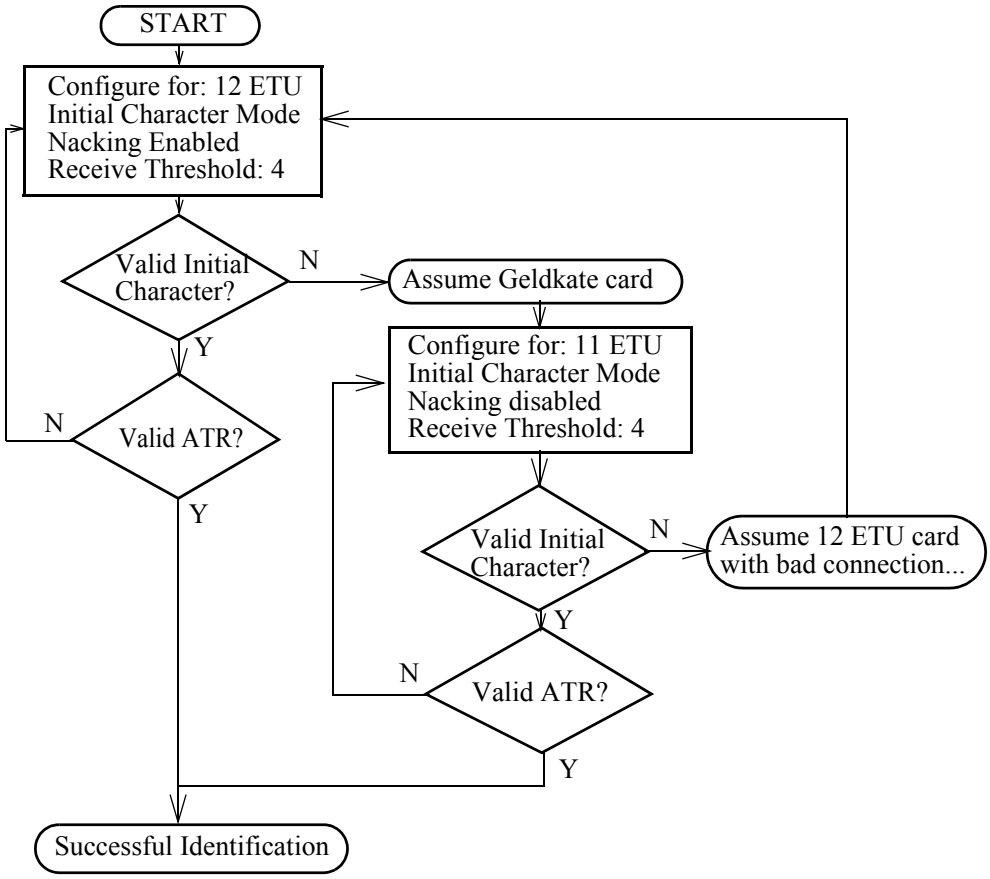


Figure 51-27. Suggested "T=1", EMV, Geldkate Compliant SIM Initialization

51.9.3.5.3 Programming Considerations for T=0 SIM Cards

If the card is of type T=0, the software should adjust the following parameters according to the information in the ATR:

1. Adjust the baud rate by changing the values of **baud_sel[2:0]** in the CNTL register
2. Adjust the guard time between characters by changing the value of **getu[7:0]** in the GUARD_CNTL register
3. Adjust NACK capability by modifying the values of the **onack** and **anack** bits in the CNTL register
4. Adjust the stop clock polarity by modifying the values of the **scsp0** or **scsp1** bits in the PORTx_CNTL register
5. Adjust the level of transmit NACK re-transmissions allowed by modifying the value of the **xth[3:0]** bits in the XMT_THRESHOLD register
6. Adjust the level for the Receive NACK threshold by modifying the **rth[3:0]** bits in the RCV_THRESHOLD register.

If a negotiation with the SIM card is desired, the software will send a PPS response to the SIM card. In order to send the response, the following steps should be performed...

7. Set the desired transmit FIFO threshold level by writing the **tdt[3:0]** bits in the XMT_THRESHOLD register
8. Write the characters to be sent as response (max 16) to the transmit FIFO using the PORTx_XMT_BUF register
9. Clear all transmit interrupt flags in the XMT_STAT register by writing a one to them
10. Enable the transmit interrupts desired by clearing the mask bits in the INT_MASK register. If more than 16 character are to be sent, it is suggested that the **tdtf** interrupt be used to signify when to write more characters to the transmit FIFO. This results in the most efficient transfer times to the SIM card.
11. Enable the transmitter by setting the **xmt_en** bit in the ENABLE register

At this point, the SIM module will transmit the characters in the transmit FIFO. If more than 16 characters are to be sent, the transmit threshold interrupt will be set when the threshold number of characters are remaining in the FIFO. The software can then write an additional number of characters to be sent without interrupting transmission to the SIM card.

Once the transmission is complete, the SIM module should be completely configured for standard operation with the T=0 SIM card. The software can continue to service **rdrf** interrupts for received characters, and **tdtf** interrupts for transmitted characters.

51.9.3.5.4 Programming Considerations for T=1 SIM Cards

If the card is of type T=1, the software should adjust the following parameters according to the information in the ATR:

1. Adjust the baud rate by changing the values of **baud_sel[2:0]** in the CNTL register
2. Adjust the guard time between characters by changing the value of **getu[7:0]** in the GUARD_CNTL register. Setting **getu[7:0]** to $FF configures the SIM transmitter for 11 ETU transmissions
3. Disable NACK capability by clearing the **onack** and **anack** bits in the CNTL register. T=1 cards do not allow NACKs.
4. Adjust the stop clock polarity by modifying the values of the **scsp0** or **scsp1** bits in the PORTx_CNTL register
5. Set Character Wait Time Counter Comparator to value specified in the ATR by using the CHAR_WAIT register
6. Enable the Character Wait Time Counter Interrupt by clearing **cwtm** in INT_MASK register
7. Enable the Character Wait Time Counter by setting the **cwten** bit in the CNTL register

8. Enable CRC or LRC error checking according to the ATR information by setting either the **crcen** or **lrcen** bit in the CNTL register. These bits will never be set at the same time!

For T=1 cards, the ATR is sent using a T=0 type of structure (12 ETU, no LRC or CRC). If a negotiation with the SIM card is desired, the software will send a PPS response to the SIM card. Otherwise, the protocol is initiated with a block transfer from the SIM module. In order to send the response or the first block, the following steps should be performed...

9. Set the desired transmit FIFO threshold level by writing the **tdt[3:0]** bits in the XMT_THRESHOLD register
10. Write the characters to be sent as response (max 16) to the transmit FIFO using the PORTx_XMT_BUF register
11. Clear all transmit interrupt flags in the XMT_STAT register by writing a one to them
12. Enable the transmit interrupts desired by clearing the mask bits in the INT_MASK register. If more than 16 character are to be sent, it is suggested that the **tdtf** interrupt be used to signify when to write more characters to the transmit FIFO. This results in the most efficient transfer times to the SIM card.
13. Enable the transmission of the error checking characters (LRC or CRC) by setting the **xmt_en_LRC_CRC** bit in the CNTL register.

Note: If the card supports PPS, the software may not be allowed to send the LRC/CRC information until the PPS exchange is completed. If so, do not set the **xmt_en_LRC_CRC** bit during the PPS exchange.

14. Enable the transmitter by setting the **xmt_en** bit in the ENABLE register

At this point, the SIM module will transmit the characters in the transmit FIFO. If more than 16 characters are to be sent, the transmit threshold interrupt will be set when the threshold number of characters are remaining in the FIFO. The software can then write an additional number of characters to be sent without interrupting transmission to the SIM card.

Once the transmission is complete, the SIM module should be completely configured for standard operation with the T=1 SIM card. The software can continue to service **rdrf** interrupts for received characters, and **tdtf** interrupts for transmitted characters.

Chapter 52
Deep Sleep Module (DSM)

Revision History Table

Revision	Date	Author	Changes
1.0	10/17/01	Matsuo Marti	Initial Release.
2.0	04/18/02	Anthony Baccala	Initial Neptune_LTS Release. Modified Patriot_Indy specification with requirements for Neptune_LTS. This includes changing Bus Interface to the IPbus and new Memory Map.
2.1	08/09/02	Anthony Baccala	Updated to reflect COUNT32 incrementing on negative edge of CKIL as opposed to positve edge in previous designs (DDTS action item ID# DSPhl15118, DSPhl14953).
2.2	05/07/03		Updated for LTE specification release.

52.1 Radio Platform Architecture

Discussion of the Deep Sleep operation of the Neptune LTE Wireless Communications Processor requires an understanding of the radio platform architecture. As shown in Figure 52-1, the major components of the radio platform are the Wireless Communications Processor (Neptune LTE), the audio interface and power control IC, and the air interface IC. As relates to Deep Sleep operation:

- The Deep Sleep Module (DSM) will control the Deep Sleep and Wake-up functions.
- Deep Sleep refers to the process by which the Layer 1 timer is disabled at a known 32 kHz clock edge, all peripherals clocked by PAT_REF/CKIH are shut down, MUX_CLK is gearshifted from PAT_REF/CKIH to 32 kHz, and PAT_REF/CKIH is turned off.
- Wake Up refers to the process by which PAT_REF/CKIH is turned on, MUX_CLK is gearshifted from 32 kHz/CKIL to PAT_REF, peripherals clocked by PAT_REF are started up, and the Layer 1 timer is enabled at a known 32 kHz clock edge.
- Neptune LTE controls the STBY input to the power control IC. Assertion of STBY typically forces external IC's into low-power standby mode.
- The DSM switches CLKIN between the crystal output (uncorrected) and the Reference DPLL output (corrected) with the CLKSEL pin.
- Ref DPLL initialization is done by the Neptune LTE MQSPI writing an initialization word over the SPI bus. This transfer will be activated by the MCU.
- Deep Sleep is used to conserve power. In Deep Sleep, the power control IC will be put in standby, subsequently powering-down the air interface IC.
- MCU Peripherals which need to be left running during Deep Sleep must be capable of running from the power control IC's 32 kHz clock (CKIL).
- Synchronization between PAT_REF/CKIH and CKIL is handled in the DSM.
- The DSM can wake up Neptune LTE in response to external interrupts.

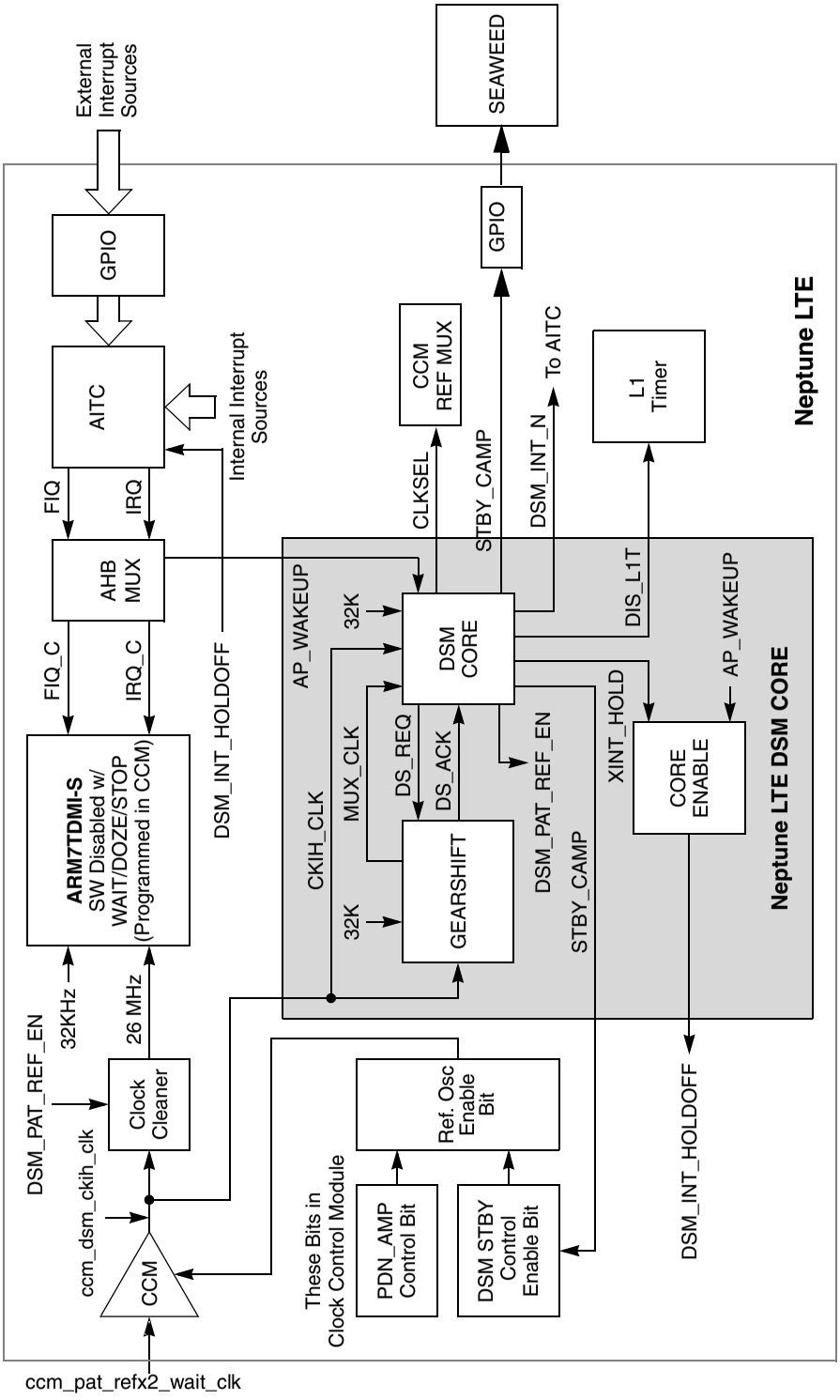


Figure 52-1. Radio platform incorporating Neptune LTE, POWER IC, and RF IC

NOTE:

PAT_REF operates at 13 MHz, CKIL at 32 kHz. The DSM will require the simultaneous use of PAT_REF, CKIL, and the DPLL output. CKIH_CLK = PAT_REF

52.2 Interaction with other MCU Peripherals

52.2.1 Clocks

52.2.1.1 Clock Cleaner

The DSM_PAT_REF_EN signal will be asserted when the system has been shifted to CKIL and PAT_REF is about to be disabled. This signal should be used in the clock cleaner to disable the PAT_REF distribution to the rest of the IC. Two CKIL cycles before WARMTIME, DSM_PAT_REF_EN will be deasserted. By this time the slicer output should be stable. Note that this signal is NOT synchronized to PAT_REF, so the clock cleaner should be implemented in such a way that clock slivers are avoided.

52.2.1.2 Gearshifting

The gearshift logic is responsible for switching the dual clock (**clk1332**) between CKIL and PAT_REF. When the **dsm/ds_req** signal changes state[1], the gearshifting logic will synchronize a changeover 1.5 32 kHz clock cycles later. Once synchronization is complete, the **ds_ack** signal will change state to match **dsm/ds_req** and acknowledge that gearshifting is complete.

52.2.1.3 Simultaneous use of PAT_REF and CKIL

The DSM is the only peripheral requiring simultaneous use of PAT_REF and CKIL. These clocks are routed directly to the DSM without any gating logic. This is done to prevent lockup situations[2]. The DSM will generate state-machine controlled internal clock enables for power conservation.

52.2.2 Interrupt Handler

To allow the DSM to restore clock function when waking up from Deep Sleep due to an external interrupt, the INT_HANDLER should assert the **wakeup** signal to the DSM whenever there is any interrupt pending. The DSM response to external interrupts is programmable, and is described in Section 52.6.1. The **dsm_int_holdoff** signal should prevent interrupts from reaching the MCU when asserted. The DSM will assert **dsm_int_holdoff** when it is programmed to restore the clock before the MCU is enabled, and to prevent race conditions during transitions to and from deep sleep. Normally, the interrupt handler resynchronizes external interrupts to PAT_REF. However, the INT_HANDLER must be able to respond asynchronously to interrupts if its clock is turned off during Deep Sleep.

52.2.3 Layer 1 Timer

The **dis_l1t** signal to the Layer1 Timer allows the DSM to shut down the Layer 1 Timer's Timing Block. The **dis_l1t** signal overrides the timer's L1T/GTC/TBE control bit. Assertion of **dis_l1t** will disable the system clock input to the timer's Timing Block. At the end of the DSM's RESTART procedure, **dis_l1t** will be deasserted. Note that the timer's Event Generator must be asleep by the time Neptune LTE enters Deep Sleep. Otherwise, a fatal SYNC error may be generated by the timer when the DSM executes RESTART.

1. **dsm/ds_ack**: 0 = PAT_REF (13 - 19.44 MHz), 1 = CKIL (32 kHz)
2. Lockup example: Neptune LTE enters Deep Sleep w/ MCU running from CKIL. MCU disables DSM_CKIL_EN. DSM state machine cannot be updated. Neptune LTE stuck running from CKIL.

52.3 Port Information

The ports of the Deep Sleep Module are shown in Figure 52-2. Port definitions are provided in Table 52-1.

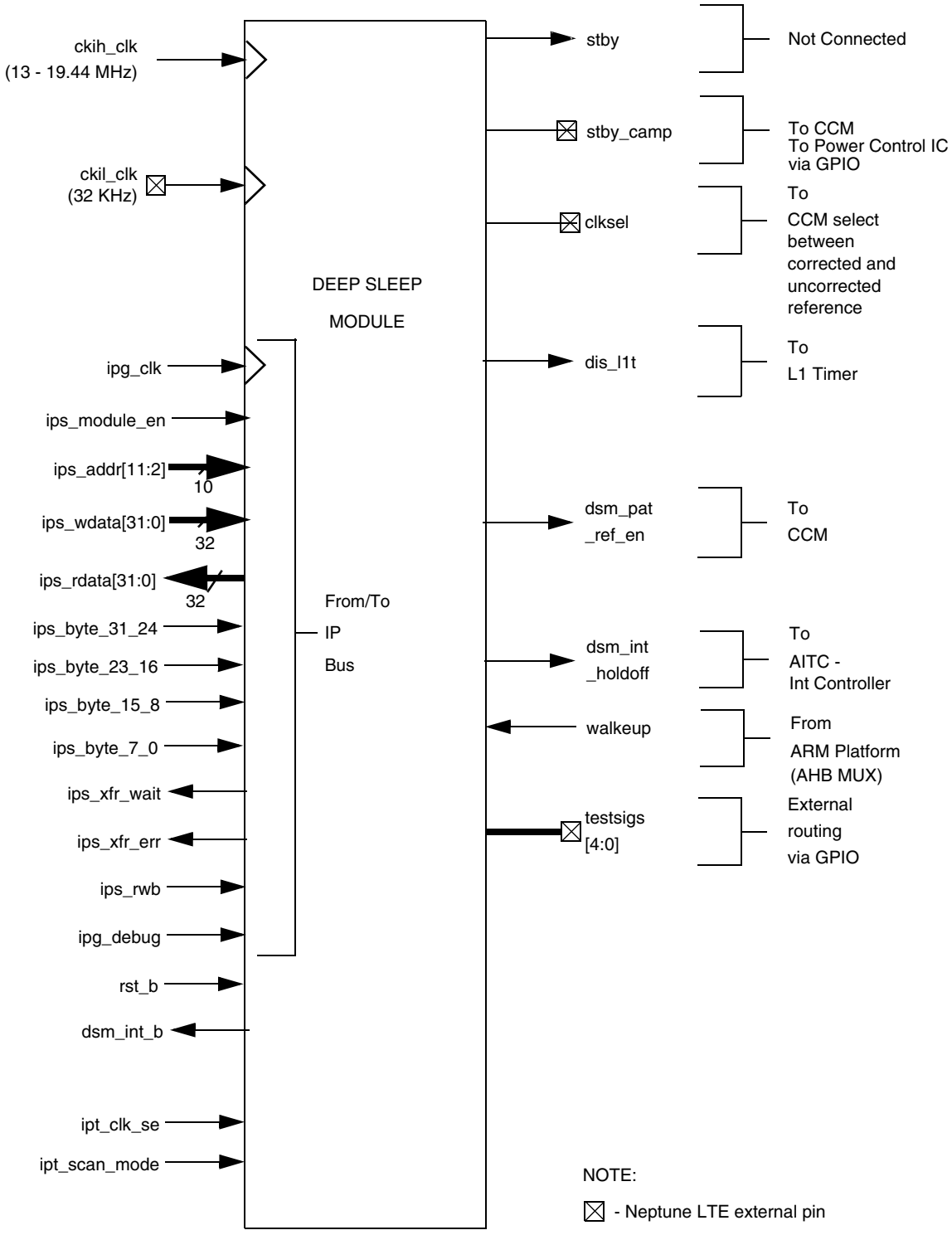


Figure 52-2. Deep Sleep Module Ports

Table 52-1. DSM Module Pin List

Pin Name	Direction	Description	To /From
ckih_clk	I	ckih_clk aka raw_pat_ref (either corrected or uncorrected) from CCM. Also referred to as ccm_dsm_ckih_clk or PAT_REF.	CCM => Neptune LTE/DSM
ckil_clk	I	32 kHz clock	Power Control IC => Neptune LTE/DSM
ipg_clk	I	52 MHz bus clock. Also referred to as MCU_CLK.	CCM => DSM
ips_module_en	I	Active-high chip-select signal	IP => DSM
ips_addr[11:2]	I	Module Address bus	IP => DSM
ips_rdata[31:0]	O	IP readdata bus	DSM => IP
ips_wdata[31:0]	I	IP write data bus	IP => DSM
ips_byte_31_24	I	IP bus byte enable for bits[31:24]	IP => DSM
ips_byte_23_16	I	IP bus byte enable for bits[23:16]	IP => DSM
ips_byte_15_8	I	IP bus byte enable for bits[15:8]	IP => DSM
ips_byte_7_0	I	IP bus byte enable for bits[7:0]	IP => DSM
ips_xfr_wait	O	IP bus transfer wait acknowledge	DSM => IP
ips_xfr_err	O	IP bus transfer error acknowledge	DSM => IP
ips_rwb	I	Read / write control (1 = read, 0 = write)	IP => DSM
ipg_debug	I	Debug control mode (0 = Normal, 1= Debug)	IP => DSM
rst_b	I	Active-low asynchronous RESET	Neptune LTE => DSM
dsm_int_b	O	Active-low Deep Sleep Module interrupt line. This signal is the NAND of all unmasked pending interrupts in the INT_STATUS register.	DSM => INT_HANDLER

Table 52-1. DSM Module Pin List (Continued)

Pin Name	Direction	Description	To /From
stby	O	Power-down control Polarity selectable in CONTROL0	DSM => Not Connected in the Neptune LTE chip
stby_camp	O	Power-down control signal to Clock Control Module and Power Control IC. Polarity is always active-high	DSM => CCM, Power Control IC
clksel	O	Clock select switch to CCM. (0 = uncorrected reference, 1 = corrected reference). This signal is routed internally.	DSM => CCM
dis_l1t	O	Active-high override signal to shut down the timer's Timing Block on a known CKIL clock edge. This signal has precedence over the value of the Timing Block Enable (TBE) bit in the timer's Global Timer Control Register.	DSM => LAYER 1 TIMER
dsm_pat_ref_en	O	Active-low signal to clock cleaner. The cleaner should disable pat_ref to Neptune LTE when DSM_PAT_REF_EN is asserted, and enable pat_ref to Neptune LTE when DSM_PAT_REF_EN is deasserted.	DSM => CLOCK CLEANER (CCM)
dsm_int_holdoff	O	Active-high gating signal to the interrupt handler. This signal is asserted when the DSM wants to prevent external interrupts from waking up the MCU.	DSM => AITC
wakeup	I	This active-low signal is asserted whenever the ARM platform has a pending interrupt.	ARM Platform => DSM
testsigs[1] [4:0]	O	Internal DSM signals useful for debug b3:b0: State Machine state (actual State Machine states which are slightly different than as in STATE register. STATE register is backward compatible to previous DSM designs) 0 - DSM_OFF 1 - MEASURING 2 - MEASDONE 3 - L1TOFF 4 - WAIT4SLEEP 5 - DEEPSLEEP 6 - DEEPSLEEP 7 - WAIT4WARM 8 - WAIT4WARM 9 - WARM 10- LOCKED 11- DSM_OFF 12- DSM_OFF 13 - 15 - INVALID STATES b4: DIS_L1T	Neptune LTE/GPIO => External pin
ipt_clk_se	I	Scan enable to clock gating cells	SCAN

Table 52-1. DSM Module Pin List (Continued)

Pin Name	Direction	Description	To /From
ipt_scan_mode	I	Scan mode to bypass system reset	SCAN

1. note: mapping the testsigs bus to the neptune lte external pins requires configuring the gpio.

52.4 Deep Sleep Module Architecture

The internal architecture of the Deep Sleep Module is shown in Figure 52-3. The Deep Sleep module consists of a Bus Interface, the main State Machine, and a Counter Block. A brief description of the operation of each of these blocks in given in Sections 52.4.1 to 52.4.3.

52.4.1 Bus Interface

The Neptune LTE MCU communicates with the Deep Sleep Module over the IPbus. The important features of the DSM interface to the bus are:

- Big Endian Memory format.
- 32-bit accesses only.
- Addressing on even words only.
- Bus cannot be used during Deep Sleep (32 kHz)
- MCU will have read access to all DSM registers at all times.
- The DSM will assert ips_xfr_err to indicate an invalid address under the following conditions:
  - — Attempting an access in an undefined address space.
  - — Attempting to write to a read-only register.
  - — Attempting a non 32-bit access
  - — Attempting to access REFCOUNT while it is running

The DSM interacts with many different clocks as part of its normal operations. Since data must be resyncronized whenever it must pass from one clock domain to another, there are delays associated with the resyncronizers. The effects of these delays have shown up in several unanticipated ways. To avoid these problems, no DSM operation should be attempted that will modify, enable, disable, or otherwise influence a timeout within four(4) CKIL cycles of the timeout. For example, if it is desired to cause RESTART as soon aspossible, software should read COUNT32, add four (4), and use that value for RESTART_TIME. Using values of 0, 1, 2 or 3 will cause erratic operation. Similarly, if it is desired to enable a state transition, the enable bit must be set no later than four (4) CKIL cycles before the state transition is to occur.

NOTE:

The DSM registers are not accessible during deep sleep. Attempts to access dsm registers while in deep sleep will result in erroneous operation, and possibly cause a lockup condition.

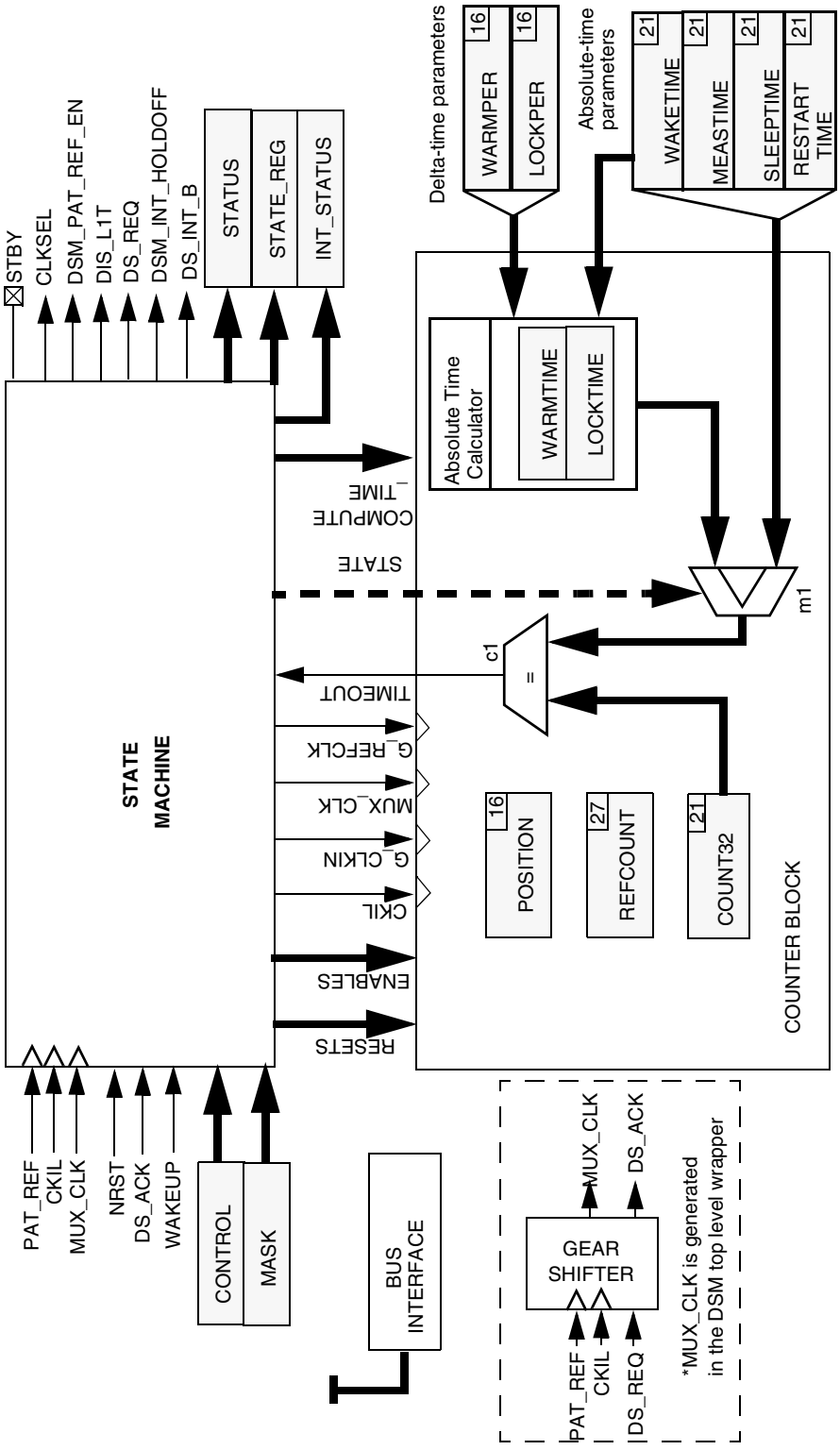


Figure 52-3. Deep Sleep Module: Top-Level Architecture

NOTE:

MCU accessible registers are shaded, vector variables are shown in bold. COUNT32 is a 21 bit counter clocked by CKIL, REFCOUNT is a 27 bit counter clocked by PAT_REF or the MCU_CLK. POSITION is a 16 bit counter indicating the number of PAT_REF counts which have elapsed since the first positive edge of CKIL after POSITION was enabled. Absolute time parameters are compared directly to the value of COUNT32. Delta-time parameters are passed to the Absolute Time Calculator to generate absolute time values to compare to COUNT32. The comparison value for generation of a TIMEOUT is selected by the STATE variable.

52.4.2 Counter Block

Table 52-2. Port Definitions: Counter Block

	I/O	DESCRIPTION
RESETS	I	Resets for POSITION, REFCOUNT, and COUNT32 counters, plus a global reset for the entire Counter Block.
ENABLES	I	Enables for the POSITION, REFCOUNT, and COUNT32 counters.
CKIL	I	CKIL (32 kHz) to Counter Block
G_CLKIN	I	Gated PAT_REF (13 -19.44 MHz) to Counter Block
MUX_CLK	I	MUX_CLK (PAT_REF or CKIL) to Counter Block
G_REFCLK	I	Gated Clock to REFCOUNTER. Can be either PAT_REF or MCU_CLK
STATE[3:0]	I	4-bit DSM state used to select which Absolute Time Value (MEASTIME, SLEEPTIME, RESTART_TIME, WKTIME, WTIME, LTIME) COUNT32 will be compared with to generate a timeout.
COMPUTE_ TIME	I	Control bus from State Machine to tell the Absolute Timer Calculator to recalculate WTIME and LTIME.
TIMEOUT	O	Active-high pulse (width = 1/2 CKIL cycle) indicating equality between COUNT32 and the currently selected Absolute Time value.
WARMPER[15:0]	I	Crystal warm-up period
LOCKPER[15:0]	I	Reference DPLL lock period
WAKETIME[21:0]	I	CKIL cycle on which to start up the power control IC and begin warming up the PAT_REF reference
MEASTIME[21:0]	I	Drift Coefficient measurement time
SLEEPTIME[21:0]	I	CKIL cycle on which to disable the Layer 1 Timer and begin the gearshift of Neptune LTE to CKIL.
RESTART_ TIME[21:0]	I	CKIL cycle on which to enable the Layer 1 Timer

The Counter Block of the Deep Sleep Module contains three counters:

- POSITION - This up-counter keeps track of the number of elapsed ccm_dsm_ckih_clk cycles since the first positive edge of CKIL after it was enabled. This counter is free-running.
- REFCOUNT - This up-counter is used during drift coefficient measurements to measure the number of elapsed ccm_dsm_ckih_clk or MCU_CLK cycles. Note that using the full range of MEASTIME causes this register to roll over. Software must account for possible rollover when computing the drift coefficient.
- COUNT32 - This upcounter is the heart of the DSM. It increments relative to the negative edge of CKIL and is used to generate the timeouts required to mark the end of the clock drift measurement period, the start of Deep Sleep, the end of Deep Sleep, the end of the crystal warm-up period, the end of the crystal lock period, and the Layer 1 Timer restart time. This 21 bit counter has a maximum range of 2,097,152 cycles (65.54 seconds with CKIL @ 32 kHz).

The Absolute Time Calculator is used to convert Delta-time parameters (WARMPER, LOCKPER) into Absolute-time parameters (WARMTIME, LOCKTIME). The Absolute Time Calculator is controlled exclusively by the DSM state machine. However, software can overwrite the results (WARMTIME, LOCKTIME) at any time.

Software is responsible for supplying values for the following Absolute Time parameters:

- MEASTIME - drift coefficient measurement time, 21 bits, range 0 to 65.54 seconds. Note that since REFCOUNT is only a 27 bit counter, use of the full-range of MEASTIME will require software to check for rollover in the REFCOUNT counter.
- SLEEPTIME - COUNT32 count at which begin gearshifting Neptune LTE to 32kHz and shut down the PAT_REF reference.
- WAKETIME - COUNT32 count at which to start up the power control IC and begin warming up the PAT_REF reference.
- WARMTIME - COUNT32 count at which the PAT_REF is usable for executing software but not for channel timing. At this time, Neptune LTE should begin programming the Reference DPLL. The Absolute Time Calculator also updates this register.
- LOCKTIME - COUNT32 count at which the Reference DPLL is locked and PAT_REF is suitable for use as a channel timing reference. The Absolute Time Calculator also updates this register.
- RESTART_TIME - COUNT32 count at which to re-enable the Layer 1 Timer.

The relationship among the parameters is shown in Figure 52-4. Note that software is responsible for ensuring that MEASTIME < SLEEPTIME < WAKETIME < WARMTIME < LOCKTIME < RESTART_TIME.

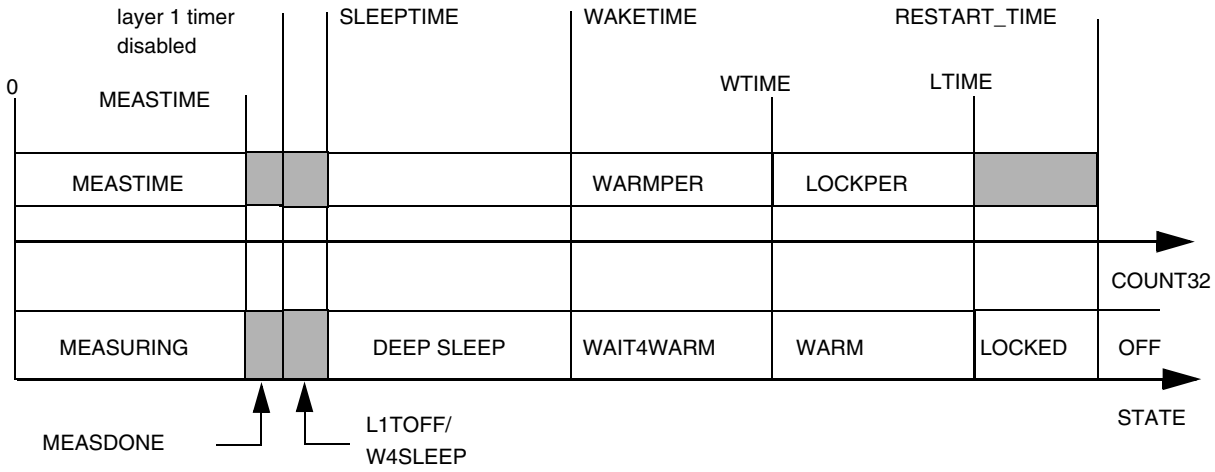


Figure 52-4. Relationship between Delta-Time parameters, Absolute-Time parameters, and the DSM state

NOTE:

Shaded areas represent periods which require assertion of DSM control bits (SLEEP, RESTART) to effect a state-transition.

52.4.3 State Machine

52.4.3.1 Port Definitions

The majority of the State Machine ports, shown in Figure 52-3, are described in either Table 52-1 or Table 52-2. However, the State Machine does receive inputs from the following DSM registers:

- CONTROL1 & 0- Resets, DSM state transition controls (MEAS, STOPL1T, SLEEP), and external interrupt response mode
- MASK - Interrupt masks

The State Machine also outputs data to the following registers:

- STATUS - Pin states
- CTREN - Enable status of REFCOUNT and COUNT32
- STATE - Current DSM state
- INT_STATUS - Interrupt status, Reason-for-wakeup (timeout / external interrupt)
- COUNT32_CAP - COUNT32 count when Layer 1 timer was disabled

52.4.3.2 State Summary

The state transition diagram for the DSM is shown in Figure 52-5. The behavior of the DSM in each state is summarized below.

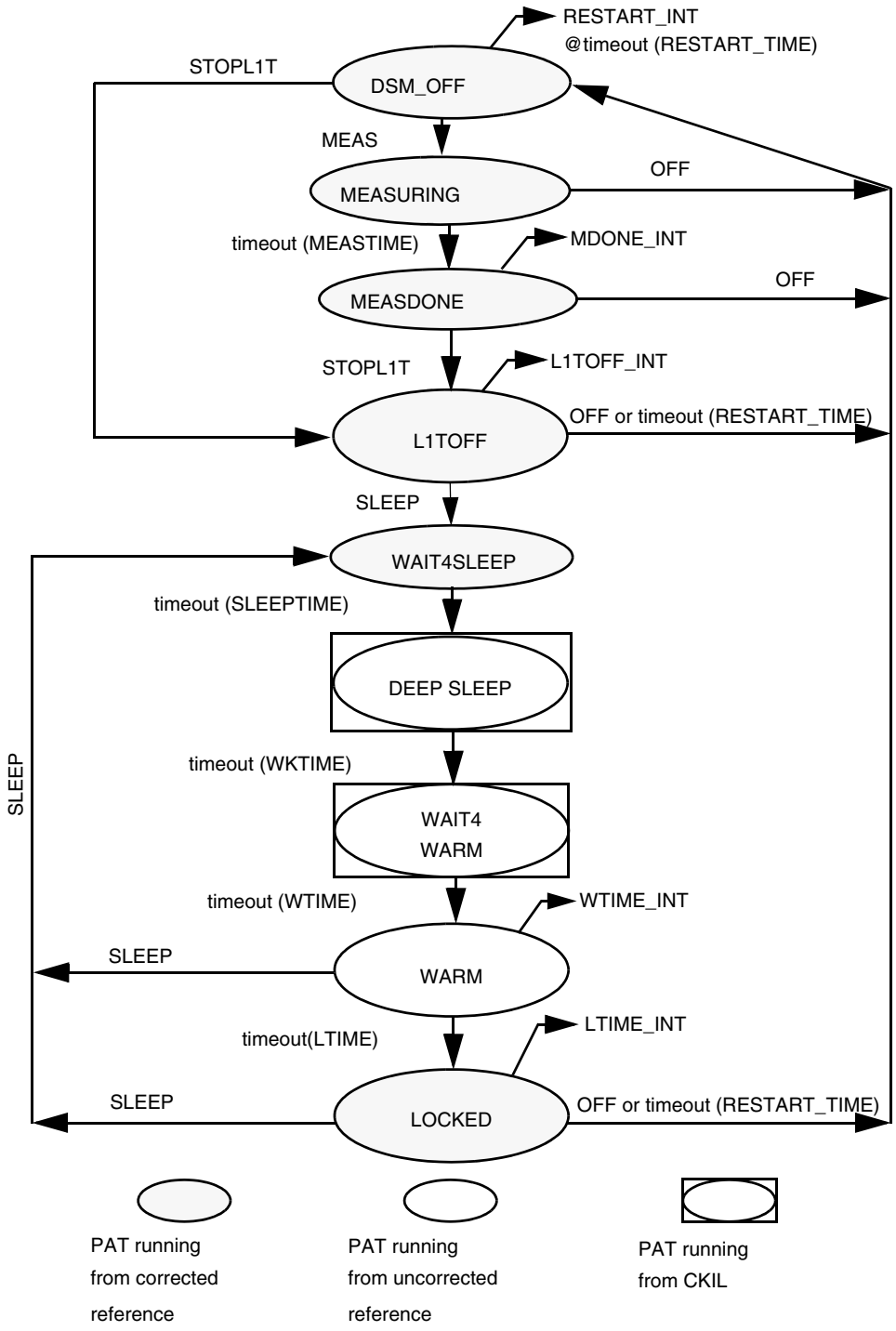


Figure 52-5. DSM state transition diagram

NOTE:

Transition conditions are listed beside each state. The DSM generates (maskable) interrupts when timeouts force a transition into the MEASDONE, WARM, and LOCKED states. The RSTRT_INT interrupt is generated when the DSM transitions from the L1TOFF or LOCKED state to the DSM_OFF state.

```
DSM_OFF: Default state
      Actions on State Entry: Turn off REFCOUNT
            Generate RESTART interrupt if last state was STOPL1T or
            LOCKED
            Deassert DIS_L1T
      MEAS assertion: Reset REFCOUNT, COUNT32
            Transition to MEASURING state
      STOPL1T assertion: Transition to L1TOFF state

MEASURING: Drift coefficient measurement
      Actions on state entry: Deassert MEAS
            Enable COUNT32, REFCOUNT counters
      timeout(MEASTIME): Disable REFCOUNT counter
            Transition to MEASDONE state
      OFF assertion: Disable COUNT32, REFCOUNT counters
            Transition to OFF state

MEASDONE: Measurement complete, DSM idle
      Actions on state entry: Assert measurement done interrupt MDONE_INT
            Disable REFCOUNT counter
      OFF assertion: Disable COUNT32, REFCOUNT counters
            Transition to OFF state
      STOPL1T assertion: Transition to L1TOFF state

L1TOFF: Layer 1 timer disable
      Actions on state entry:  Deassert STOPL1T
            Assert DIS_L1T
            Capture COUNT32 value in COUNT32_CAP register
      SLEEP assertion: Transition to WAIT4SLEEP state
      OFF assertion: Disable COUNT32 counter, if the state machine enabled
COUNT32. Transition to off state. RESTART_INT will be asserted.
            WARNING: this transition will cause the Layer 1 timer to be restart-
ed on the next negative edge of CKIL!!
      timeout(RESTART): Restart Layer 1 timer on next negative edge of CKIL if
RESTART is enabled in control0 register.

WAIT4SLEEP: Waiting for SLEEPTIME timeout
      Actions on state entry: Deassert SLEEP
      timeout (SLEEPTIME): Assert DSM_INT_HOLDOFF (if required)
            Calculate WTIME, LTIME
            Transition to DEEP SLEEP state

DEEP SLEEP: Low power sleep state
      Actions on state entry: Assert DS_REQ to start gearshift-down process
            Pull down CLKSEL to "crystal" state
            Disable POSITION counter (if enabled)
            Assert STBY once DS_ACK is asserted
      External Interrupt: Recalculate WTIME for ASAP wakeup. Recalculate LTIME
            Deassert STBY
            Transition to WAIT4WARM state
      timeout(WKTIME): Deassert STBY
            Transition to WAIT4WARM state

WAIT4WARM: Crystal warmup idle state
      Actions on state entry: None
```

timeout (WTIME): Deassert DS_REQ
Deassert DSM_INT_HOLDOFF
Transition to WARM state

WARM: Crystal warmup complete
Actions on state entry: Assert warmtime interrupt WTIME_INT
timeout(LTIME): Pull up CLKSEL to "VCO" state
Transition to LOCKED state
SLEEP assertion: Transition to WAIT4SLEEP state[1]

LOCKED: Ref DPLL VCO locked, channel-timing accurate clock available
Actions on state entry: Assert lock time interrupt LTIME_INT
timeout(RESTART): Transition to DSM_OFF state if RESTART is enabled in control0 register.
SLEEP assertion: Transition to WAIT4SLEEP state[1]
OFF assertion: Disable COUNT32 counter, if the state machine enabled COUNT32. Transition to off state. RESTART_INT will be asserted.
WARNING: this transition will cause the Layer 1 timer to be restarted on the next negative edge of CKIL!!

1. Note: Typical use of SLEEP during the WARM and LOCKED states is to return to Deep Sleep following the processing of an external interrupt. Note that since the DSM will transition to the WAIT4SLEEP state WARMTIME and LOCKTIME will be recalculated based on the value of WAKETIME, which is **not** overwritten when an external interrupt occurs.

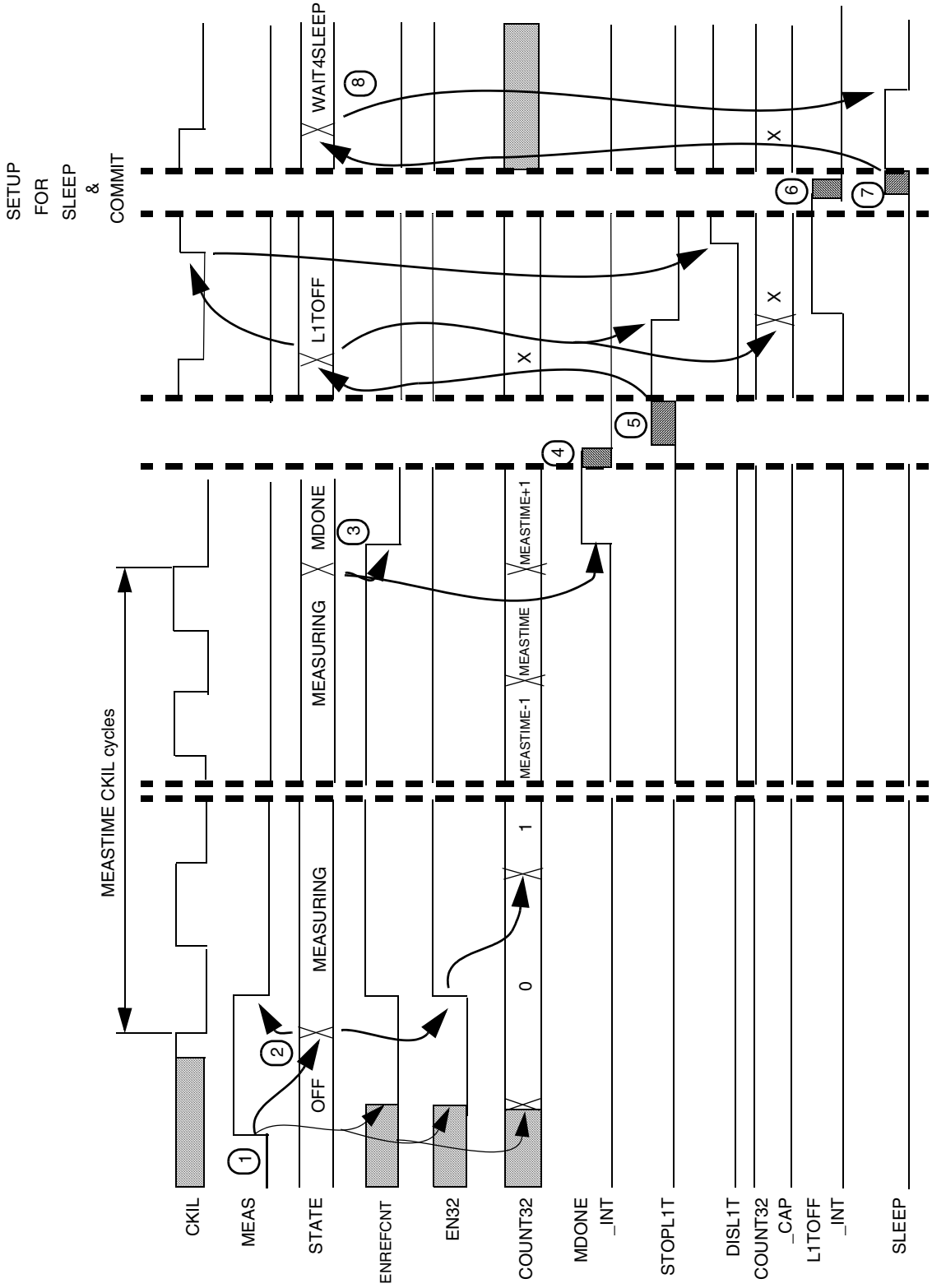


Figure 52-6. SLEEP Sequence: Typical drift measurement and Deep Sleep preparation sequence. Timing is not to scale.

52.5 Sleep Sequence

Please refer to Figure 52-6 and Figure 52-7 for the timing diagrams of a typical SLEEP sequence.

52.5.1 Drift Measurement

The DSM starts in the OFF state. The MCU must program the clock drift measurement period MEASTIME.

1. The measurement process is started by asserting the MEAS bit in the DSM CONTROL1 register. Asserting MEAS will then deassert the ENREFCNT and EN32 clock enables to the REFCOUNT and COUNT32 counters. The RSTREFCNT and RST32 reset lines are also asserted at this time, clearing the REFCOUNT and COUNT32 counters.
2. At the next negative edge of CKIL, the drift measurement process begins when the DSM switches to the MEASURING state. (Note that drift measurement can be cancelled at any time up to this clock edge by asserting the OFF bit in the CONTROL1 register). This state transition causes the EN32 and ENREFCNT signals to assert (enabling COUNT32 and REFCOUNT) and self-clearing of the MEAS bit.
3. When COUNT32 == MEASTIME, drift measurement is complete and the DSM state is updated to MDONE. The REFCOUNT counter is then disabled. The COUNT32 counter is NOT turned off by the DSM. The total time spent in the MEASURING state is (MEASTIME) 32 kHz clock cycles. The state transition asserts the MDONE_INT interrupt flag in the INT_STATUS register. If this interrupt is unmasked in the MASK register, the DS_INT_BDS_INT_B interrupt signal will be asserted.
4. If necessary, clear the MDONE_INT interrupt in the DSM's INT_STATUS register.
5. When ready, the call-processor must assert STOPL1T bit in the CONTROL1 register. On the next negative edge of CKIL, the value of COUNT32 will be captured in the COUNT32_CAP register. On the following rising edge of CKIL, the Layer 1 timer will be disabled. The state transition asserts the L1TOFF_INT in the INT_STATUS register. If this interrupt is unmasked in the MASK register, the DS_INT_B interrupt signal will be asserted.

52.5.2 Setup for Sleep

6. In preparation for Deep Sleep, the call-processor software must:
   - Calculate the drift coefficient by comparing MEASTIME and REFCOUNT.
   - Determine the time to enter sleep (SLEEPTIME).
   - Determine the time to begin the wakeup sequence (WAKETIME).
   - Program the DSM with the desired values of WARMPER and LOCKPER.
   - Program the DSM with the desired external interrupt response (XIRESP[1:0]) in the DSM's CONTROL0 register.
   - If necessary, clear the L1TOFF_INT interrupt in the DSM's INT_STATUS register (6).
   - Determine the time to re-enable the Layer 1 timer (RESTART_TIME).
   - Re-calibrate the Layer 1 timer (See (See **Layer 1 Timer Re-calibration** on page -20.).).

Optionally, the c/p s/w may typically:

- Return the DSM to the OFF state by asserting the OFF bit in the CONTROL1 register NOTE: this will asynchronously re-enable the Layer 1 timer!
- Disable unneeded peripherals
- Mask unwanted interrupts

52.5.3 Commit to Deep Sleep

7. When all setup tasks have been completed, the user asserts the SLEEP bit in the DSM's CONTROL1 register. From this point on, the user is committed to the SLEEP process. Up until the time Deep Sleep is entered, only an external interrupt can cancel the SLEEP request. An external interrupt during this period will:
   — clear the SLEEP bit in the CONTROL1 register
   — set the RWK (Reason-for-Wakeup) bit in the INT_STATUS register
   — set the DSM to the L1TOFF state
8. On the next negative edge of CKIL, the DSM state is updated to WAIT4SLEEP. The state transition deasserts the SLEEP bit in the DSM's CONTROL1 register.

52.5.4 Enter Deep Sleep and Gearshift

See Figure 52-7 for steps 9 - 12.

9. When COUNT32 == SLEEPTIME, the DSM_INT_HOLDOFF line is asserted on the next negatve edge of CKIL. This is done to ensure that the DSM can complete the gearshift process without interference from external interrupts.
10. On the same negative edge of CKIL, the DSM switches to the DEEP SLEEP state. The CLKSEL signal is switched to the CRYSTAL setting (CLKSEL = 0). The DS_REQ line is pulled high to initiate a gearshift to CKIL. Based on the values of WAKETIME, WARMPER, and LOCKPER, the DSM's Absolute Time Calculator computes the values of WARMTIME and LOCKTIME.
11. 1.5 CKIL cycles later, the Gearshifter pulls the DS_ACK signal high. The DSM_PAT_REF_EN signal is asserted, cutting off PAT_REF to Neptune LTE. Based on the selected external interrupt response in the CONTROL0 register, DSM_INT_HOLDOFF is either left on or deasserted.
12. 2 CKIL cycles later, STBY is asserted. The power control IC will enter low-power mode. This will also cut power to the air interface IC.

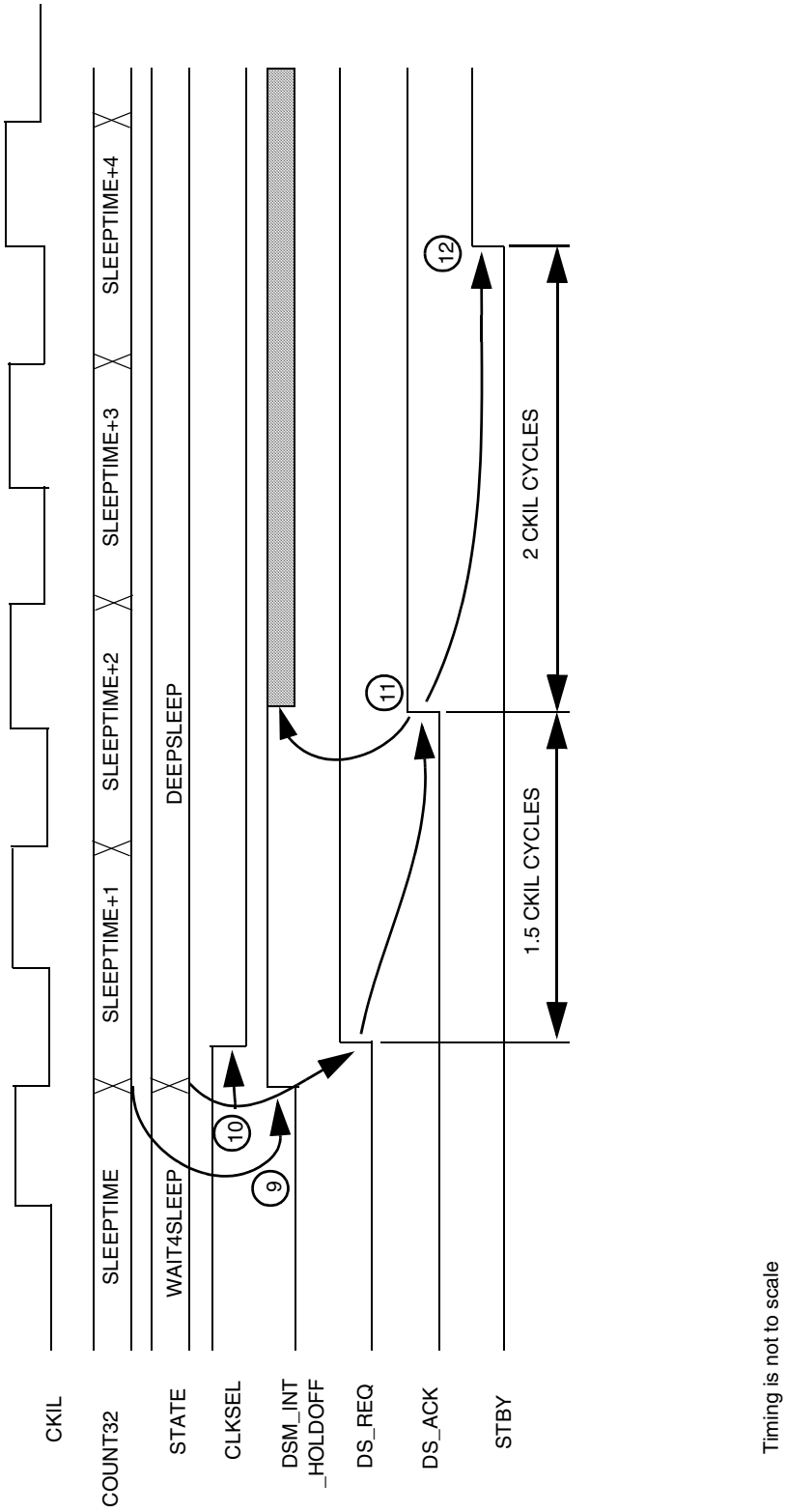


Figure 52-7. DSM entering Deep Sleep and gearshifting PAT_REF from ccm_dsm_ckih_clk frequency to CKIL frequency

52.5.5 Layer 1 Timer Re-calibration

To restart the Layer 1 timer, the timer's Timing Block must be re-calibrated. Re-calibration requires that software:

- — Read the value of COUNT32_CAP
- — Have a drift-coefficient measurement
- — Read the values of PRESCALER, RQBC, RAFN, and CFN. These should be the values in these registers when the Timing Block was shut down by assertion of DIS_L1T.

- Pick a future value of COUNT32 at which to restart the timer. Program this value into the RESTART_TIME register.
- RECALIBRATION EQUATIONS

```
Given: MEASTIME - Drift Measurement time in 32 kHz clock cycles
       REFCNT - ccm_dsm_ckih_clk count during drift measurement time MEASTIME
       RQBC_O - Old L1T Quarter-bit count at time timer was disabled
       RAFN_O - Old L1T Reference Frame Number
       PRE_O - Old L1T Prescaler count
       TRQBCM = Timer Reference Quarter Bit Count Modulus
       TBPREM = Timer Prescaler Modulus

       RESTART_TIME - COUNT32 value at which to re-enable Layer 1 Timer
       COUNT32_CAP - COUNT32 value at which the Layer 1 Timer was disabled

Let:   RAFN_N = New Reference Frame Number
       RQBC_N = New Reference Quarter Bit Count
       PRE_N = New Prescaler count

Then: delta32 = RESTART_TIME - COUNT32_CAP
       deltaREF = rnd[(REFCNT * delta32) / (MEASTIME)]
       delta = deltaREF + (TBPREM+1)*RQBC_O + PRE_O
       frame = (TBPREM + 1) * (TRQBCM + 1)
       nframes = floor [ delta / frame ]
       RAFN_N = RAFN_O + nframes

       delta = delta - frame * nframes
       RQBC_N = floor[delta/(TBPREM+1)]                  [0 <= RQBC_N <= TRQBCM]
       PRE_N = delta - (TBPREM+1)*RQBC_N                 [0 <= PRE_N <= TBPREM]
where
       rnd: round to nearest integer
       floor: round towards zero
```

- Program the new values of RAFN_N, RQBC_N, and PRE_N into the Layer 1 timer.

52.6 Wakeup Sequence

52.6.1 Response to External Interrupts

The DSMs response to external interrupts is programmable via the XIRESP bits in the CONTROL0 register. There are three possible responses to interrupts:

MODE 1: Restore PAT_REF, then enable Neptune LTE

This is the default response. This mode assumes the MCU will be shut down during Deep Sleep.The DSM_INT_HOLDOFF signal remains asserted through Deep Sleep, preventing MCU restart until the DSM has reached the WARM state. The RWK (reason for wakeup) bit in the INT_STATUS register is set, indicating wakeup due to an external interrupt.

MODE 2: Restore PAT_REF and enable Neptune LTE

This mode allows the clock restoration and MCU startup processes to be performed in parallel. The MCU will begin processing the interrupt while running from CKIL, and PAT_REF will be used when the DSM reaches the WARM state. The DSM_INT_HOLDOFF line is deasserted during Deep Sleep in this mode. The MCU can either be enabled or disabled during Deep Sleep in this mode. The RWK bit in the INT_STATUS register is set, indicating wakeup due to an external interrupt.

MODE 3: Enable Neptune LTE

This mode is used when it is desirable to process the interrupt at 32 kHz without restoring PAT_REF. The DSM_INT_HOLDOFF line is deasserted during Deep Sleep in this mode. The MCU can either be enabled or disabled during Deep Sleep in this mode. Note that PAT_REF will be restored if the normally scheduled wakeup (WKTIME timeout) occurs during interrupt processing. The RWK bit in the INT_STATUS register is left low, indicating that the DSM was not woken up due to an external interrupt.

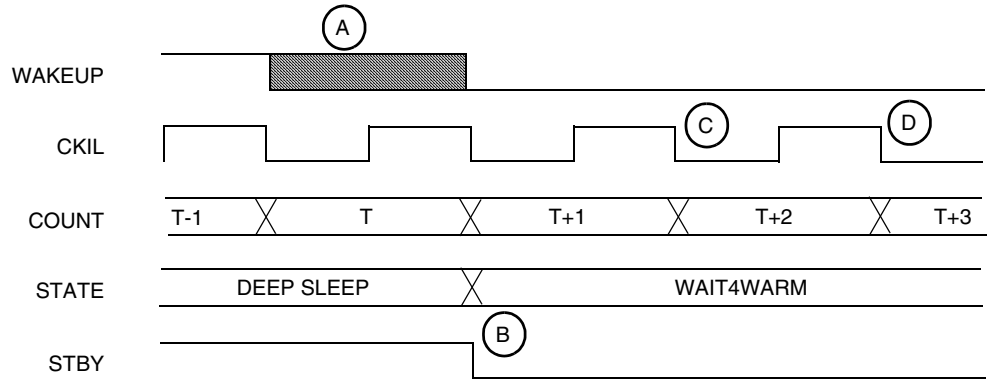


Figure 52-8. Wakeup due to external interrupt for interrupt response modes 1 and 2.

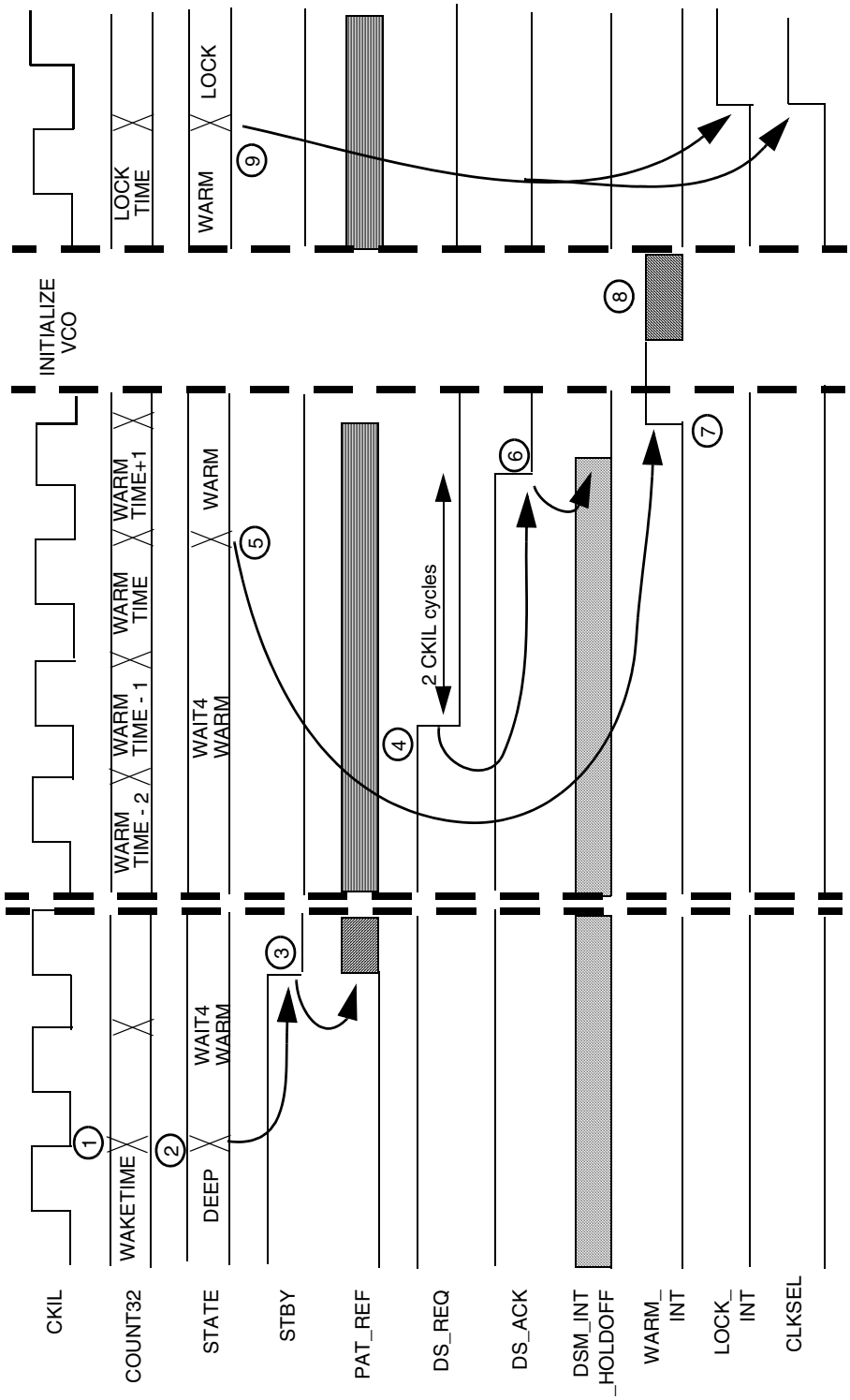


Figure 52-9. WAKEUP sequence with wakeup due to counter timeout. Timing not to scale

For MODE 1 and MODE 2 interrupts, the DSM must wake up as quickly as possible. This process is illustrated in Figure 52-8.

a) An external interrupt causes the ARM Platform to assert the WAKEUP signal. The DSM recognizes the transition on WAKEUP and begins the wakeup process.

b) The DSM deasserts STBY, releasing the power control IC from low-power mode and beginning the power-up of the Reference DPLL. The DSM switches from the DEEP SLEEP state to the WAIT4WARM state and sets the RWK bit in the INT_STATUS register.

NOTE:

The RWK bit is not an interrupt source, just an indicator that the DSM responded to an external interrupt.

c) The Absolute Time Calculator recalculates a new value for WARMTIME.

d) The Absolute Time Calculator recalculates a new value for LOCKTIME.

52.6.2 Wakeup via Counter Timeout

Please refer to the wakeup sequence timing diagram in Figure 52-9.

1. The wakeup sequence begins with a Deep Sleep completion timeout (COUNT32 == WAKETIME).
2. The DSM state changes from DEEP SLEEP to WAIT4WARM.
3. The state transition causes the DSM to begin the ccm_dsm_ckih_clk crystal warm-up process by deasserting the STBY control line. Shortly after the deassertion of STBY, the reference DPLL will begin generating an output, but this clock is not yet stable enough to be usable. Note that there is essentially little difference in the wakeup sequences for timeouts and external interrupts. The only major difference is that for external interrupts, the transition from DEEP to WAIT4WARM is followed by recalculation of WARMTIME and LOCKTIME on the next two CKIL cycles (see Figure 52-8).

52.6.3 Crystal Warm Up Complete

4. 1.5 CKIL cycles before the DSM transitions from the WAIT4WARM to WARM states, the DSM pulls the DS_REQ and DSM_PAT_REF_EN signals low to begin gearshifting PAT_REF from CKIL frequency to ccm_dsm_ckih_clk frequency. At this point, ccm_dsm_ckih_clk should be stable enough for use by Neptune LTE.
5. The DSM transitions from WAIT4WARM to WARM.
6. 2 CKIL cycles after the deassertion of DS_REQ, the gearshifter pulls the DS_ACK line low, signalling that the gearshifting process is complete. If external interrupt response mode 1 was being used (refer to Section 52.6.1), then the DSM deasserts DSM_INT_HOLDOFF. This will allow the MCU enable logic to respond to external interrupts.
7. The DSM signals completion of the crystal warmup process by asserting the WTIME_INT interrupt in the INT_STATUS register. If this interrupt is unmasked in the MASK register, the DSM_INT_N signal will be asserted[1].

1. Note: If external interrupt response mode 1 is being used, the DSM is responsible for waking up the MCU. To do this, the WTIME_INT interrupt must be unmasked in the DSM/INT_STATUS register and the DSM interrupt must be unmasked in the INT_HANDLER/MASK register.

52.6.4 Initialize VCO

Assertion of the DS_INT_B signal will result in the MCU branching to an interrupt service routine (ISR) to handle the DSM interrupt. The first task in this ISR should be for Neptune LTE to initialize the Reference DPLL.

8. At the end of the ISR, the WARM_INT status bit should be cleared in the INT_STATUS register. This will deassert the DSM_INT_N interrupt line.

52.6.5 Reference DPLL Lock

9. The lock period completion timeout causes the DSM to transition from the WARM to the LOCK state Note that this transition is solely due to a blind time-out - there is no status feedback that the Reference DPLL has locked. For this reason, use conservative estimates of the lock period LOCKPER.

The state transition causes the CLKSEL signal to switch from low (to select the uncorrected raw crystal output) to high (to select the corrected Reference DPLL output). The state transitions also causes assertion of the LTIME_INT interrupt. If this interrupt is unmasked in the MASK register, the DSM_INT_N signal will be asserted.

52.6.6 Layer 1 Timer Restart

Please see the timing diagram in Figure 52-10.

10. The RESTART enable bit should be set in the CONTROL0 register. The LOCK_INT interrupt status bit in the INT_STATUS register should be cleared at this time.
11. When COUNT32 == RESTART_TIME, the DSM will transition from the LOCK state to the OFF state. The COUNT32 counter is turned off via deassertion of EN_32, and the restart interrupt (RSTRT_INT in the INT_STATUS register) is asserted. If this interrupt is unmasked in the MASK register, the DS_INT_B signal will be asserted. At the next positive edge of CKIL the layer 1 timer timing block is reenabled. Note that at the next negative edge of CKIL COUNT32 is not incremented.

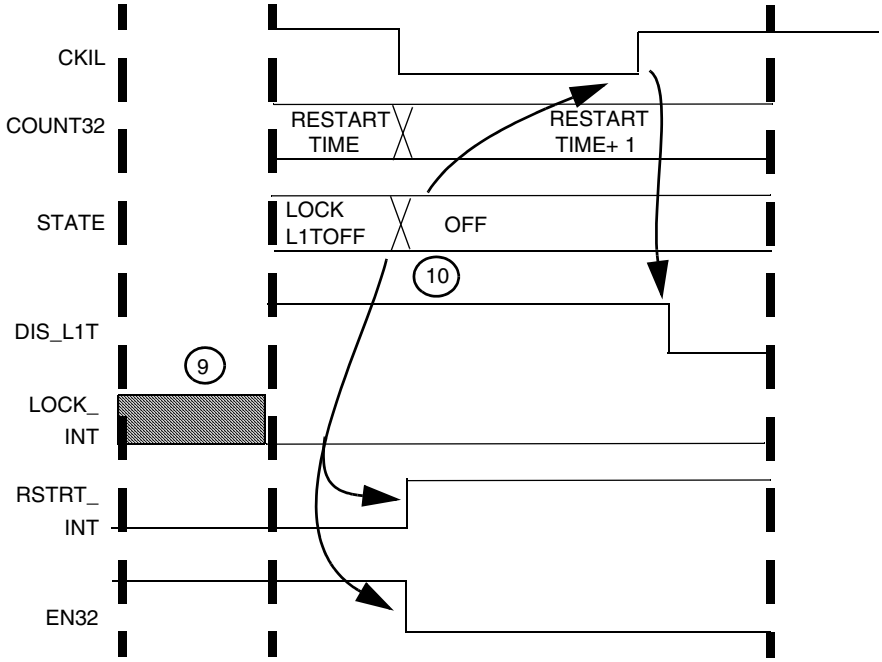


Figure 52-10. WAKEUP Sequence: Restarting the Layer 1 Timer

52.7 Memory Map

Deep Sleep Module registers are divided into:

- Longword counter contents
- Preprogrammed absolute-time parameters
- Derived absolute-time parameters
- Global control and status
- Delta-time parameters
- Word counter contents

Figure 0-12.

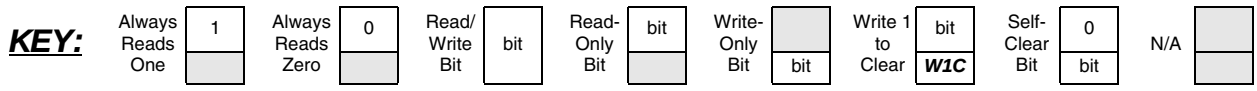


Table 52-3. DSM Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT32 $2484_B000	R	0	0	0	0	0	0	0	0	0	0	0	COUNT32[20:16]				
	W																
	R	COUNT32[15:0]															
	W																
REFCOUNT $2484_B004	R	0	0	0	0	0	REFCOUNT[26:16]										
	W																
	R	REFCOUNT[15:0]															
	W																
MEASTIME $2484_B008	R	0	0	0	0	0	0	0	0	0	0	0	MEASTIME[20:16]				
	W																
	R	MEASTIME[15:0]															
	W																
SLEEPTIME $2484_B00C	R	0	0	0	0	0	0	0	0	0	0	0	SLEEPTIME[20:16]				
	W																
	R	SLEEPTIME[15:0]															
	W																
RESTART_TIME $2484_B010	R	0	0	0	0	0	0	0	0	0	0	0	RESTART_TIME[20:16]				
	W																
	R	RESTART_TIME[15:0]															
	W																
WAKETIME $2484_B014	R	0	0	0	0	0	0	0	0	0	0	0	WKTIME[20:16]				
	W																
	R	WKTIME[15:0]															
	W																
WARMTIME $2484_B018	R	0	0	0	0	0	0	0	0	0	0	0	WTIME[20:16]				
	W																
	R	WTIME[15:0]															
	W																
LOCKTIME $2484_B01C	R	0	0	0	0	0	0	0	0	0	0	0	LTIME[20:16]				
	W																
	R	LTIME[15:0]															
	W																
CONTROL0 $2484_B020	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	DEBUG	STBY_INV	REF_SEL	RESTART	EN_POS	XIRESP[1:0]		MSTR_EN
	W																
CONTROL1 $2484_B024	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	STP **L1T**	SLEEP	MEAS	OFF	RST POS	RSTREFCNT	RST CNT32	SM RST	CB RST	SRST
	W																
CTREN $2484_B028	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EN_REFCNT	EN_CNT32
	W																

Table 52-3. DSM Register Summary (Continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STATUS $2484_B02C	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MUX_CLK	WAKEUP	DS_ACK	DSM_INT_HOLDOFF	DS_REQ	DIS_L1T	CLK_SEL	STBY
	W																
STATE $2484_B030	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	STATE[3:0]			
	W																
INT_STATUS $2484_B034	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	WKTIME_INT	SLEEP_INT	RWK	NGK_INT	PGK_INT	STPL1T_INT	RSTRT_INT	LOCK_INT	WTIME_INT	MDONE_INT
	W							W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
MASK $2484_B038	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	WKTIME_INT_M	SLEEP_INT_M	0	NGK_INT_M	PGK_INT_M	STPL1T_INT_M	RSTRT_INT_M	LOCK_INT_M	WTIME INT_M	MDONE_INT_M
	W																
COUNT32_CAP $2484_B03C	R	0	0	0	0	0	0	0	0	0	0	0	COUNT32_CAP[20:16]				
	W																
	R	COUNT32_CAP[15:0]															
	W																
WARMPER $2484_B040	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	WARMPER[15:0]															
	W																
LOCKPER $2484_B044	R		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	LOCKPER[15:0]															
	W																
POSCOUNT $2484_B048	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	POSCOUNT[15:0]															
	W																
MGPER $2484_B04C	R	EN_MGFX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	MGPER[9:0]								
	W																

52.8 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various DSM registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.

- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

COUNT32 Timebase Counter Register **Addr $2484_B000**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
												COUNT32[20:16]				
TYPE	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	COUNT32[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 52-4. Count32 Description

Name	Description	Settings
Bits 31-21	Reserved	N/A
COUNT32 [20:0] Bits 20-0	**Counter 32—** is an upcounter that serves as the master timebase for the DSM. It increments on the negative edge of CKIL.	COUNT32 may be set to any value by writing the value to COUNT32.

REFCOUNT Reference Counter Register **Addr $2484_B004**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
						REFCOUNT[26:16]										
TYPE	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	REFCOUNT[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 52-5. REFCOUNT Description

Name	Description	Settings
Bits 31-27	Reserved	N/A
REFCOUNT [26:0] Bits 26-0	**Reference Count—** REFCOUNT is an upcounter that keeps track of ccm_dsm_ckih_clk or DPLL clock cycles during measurement. Accessing this register when the counter is enabled (EN_REFCNT bit in CTREN register = 1) will cause a bus error.	REFCOUNT may be set to any value by writing to REFCOUNT.

NOTE:

Careful! Remember using the full measurement time can cause rollover in this counter.

52.8.1 Preprogrammed Absolute-Time Parameters

MEASTIME Measurement Time Register **Addr $2484_B008**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
												MEASTIME[20:16]				
TYPE	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	MEASTIME[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 52-6. MEASTIME Description

Name	Description	Settings
Bits 31-21	Reserved	N/A
MEASTIME [20:0] Bits 20-0	**Measurement Time—** Measurement end time in CKIL clock cycles referenced to COUNT32.	COUNT32 value to end measurement.

SLEEPTIME Sleep Time Register **Addr $2484_B00C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
												SLEEPTIME[20:16]				
TYPE	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	SLEEPTIME[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 52-7. SLEEPTIME Description

Name	Description	Settings
Bits 31-21	Reserved	N/A
SLEEPTIME [20:0] Bits 20-0	Sleep Time— Deep Sleep start time in CKIL clock cycles.referenced to COUNT32.	COUNT32 value to begin deep sleep.

RESTART_TIME — Restart Time Register — Addr $2484_B010

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
												RESTART_TIME[20:16]				
TYPE	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	RESTART_TIME[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 52-8. RESTART_TIME Description

Name	Description	Settings
Bits 31-21	Reserved	N/A
RESTART_TIME [20:0] Bits 20-0	Restart Time— Layer 1 Timer restart time in CKIL clock cycles.referenced to COUNT32.	COUNT32 value to restart the layer 1 timer.

WAKETIME Wake-Up Time Register **Addr $2484_B014**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
												WAKETIME[20:16]				
TYPE	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	WAKETIME[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 52-9. WAKETIME Description

Name	Description	Settings
Bits 31-21	Reserved	N/A
WAKETIME [20:0] Bits 20-0	Wake-Up Time— Wake up time in CKIL clock cycles. Wake-Up Time— Wake up time referenced to COUNT32.	COUNT32 value to re-enable the ccm_dsm_ckih_clk crystal amplifier.

52.8.2 Derived Absolute-time Parameters

WARMTIME Warm-Up Time Register **Addr $2484_B018**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
												WARMTIME[20:16]				
TYPE	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	WARMTIME[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 52-10. WARMTIME Description

Name	Description	Settings
Bits 31-21	Reserved	N/A
WARMTIME [20:0] Bits 20-0	**Warm-Up Time—** Warm Up completion time (i.e. time to enter WARM state) in CKIL clock cycles. **Warm-Up Time—** Warm Up completion time (i.e. time to enter WARM state) referenced to COUNT32. The absolute time calculator updates this value on transition to deep sleep or on wakeup from external interrupt. The calculated value may be overwritten by writing to this register.	COUNT32 value to gearshift from CKIL to ccm_dsm_ckih_clk.

LOCKTIME — Lock Time Register — Addr $2484_B01C

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
												LOCKTIME[20:16]				
TYPE	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	LOCKTIME[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 52-11. LOCKTIME Description

Name	Description	Settings
Bits 31-21	Reserved	N/A
LOCKTIME [20:0] Bits 20-0	**Lock Time—** LOCK completion time (i.e. time to enter LOCKED state) in CKIL clock cycles. **Lock Time—** LOCK completion time (i.e. time to enter LOCKED state) referenced to COUNT32. The absolute time calculator updates this value on transition to deep sleep or on wakeup from external interrupt. The calculated value may be overwritten by writing to this register.	COUNT32 value to switch PAT_REF from uncorrected crystal to corrected Reference DPLL.

52.8.3 Control and Status Registers

CONTROL0 Control 0 Register **Addr $2484_B020**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									DEBUG	STBY_INV	REF_SEL	RESTART	EN_POS	XIRESP[1:0]		MSTR_EN
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 52-12. CONTROL0 Description

Name	Description	Settings
Bits 31-8	Reserved	N/A
DEBUG Bit 7	**Debug Mode —** Commands the DSM to respond to the global DEBUG_B signal from the Core. When debug mode is enabled and the Core's debug signal is active, DSM clocks are disabled.	1 - Debug mode is enabled 0 - Debug mode is disabled
STBY_INV Bit 6	**Standby Inverter—** Inverts the normal (active high) polarity of the STBY signal. **Note:** This is useless in the Neptune LTE chip since the stby signal is no longer routed out of the chip. The stby_camp is the signal that is now routed out of the chip, and it is always active-high.	1 - STBY is active LOW 0 - STBY is active HIGH
REF_SEL Bit 5	**Reference Select—** Selects between PAT_REF and MCU_CLK as the clock source for REFCOUNT.	1 - MCU_CLK is used for REFCOUNT 0 - PAT_REF is used for REFCOUNT
RESTART Bit 4	**Restart—** Enable restarting the Layer 1 Timing Block. If this bit is set and COUNT32 == RESTART_TIME the Layer 1 timer will be restarted on the next positive edge of CKIL. **Note:** In the Whitecap DSM, this bit was a state transition request bit. In the Neptune LTE, it is merely an enable -- it may be enabled any time before RESTART_TIME - 1. There are no interlocks around restart, so it is software's responsibility to make sure restart does not occur during deep sleep!	1 - RESTART is enabled 0 - RESTART is disabled

Table 52-12. CONTROL0 Description (Continued)

Name	Description	Settings
EN_ POS Bit 3	**Enable Position—** Enable the position counter POSCOUNT. **Note:** The value of POSCOUNT will not be valid until the first positive CKIL edge following assertion of this bit.	1 - POSCOUNT is enabled 0 - POSCOUNT is disabled
XIRESP[1:0] Bits 2-1	**External Interrupt Response—** Response to external interrupts.	00 - Restore PAT_REF. Enable MCU only after PAT_REF is available 01 - Begin restoring PAT_REF. Enable MCU, process @ CKIL until PAT_REF ready 10 - Enable MCU. Do not restore PAT_REF prematurely. 11 - Reserved
MSTR_EN Bit 0	DSM Master Enable **Note:** Disabling this bit gates off all clock lines into the DSM, shutting down the module. Note that all registers can be read when the DSM is disabled, but writing to registers may produce undefined results. Only the MSTR_EN bit can be safely written to when the DSM is disabled. Also note if the DSM is turned off without executing a software reset afterwards, it will be in the same state when it is re-enabled. Also, note that when the Master Enable is first enabled it will set CLK_SEL to 1.	0 = DSM disabled 1 = DSM enabled

CONTROL1 Control 1 Register **Addr $2484_B024**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
							STPL1T	SLEEP	MEAS	OFF	RST_POS	RST_REFCNT	RST_CNT32	SM_RST	CB_RST	SRST
TYPE	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 52-13. CONTROL1 Description

Name	Description	Settings
Bits 31-10	Reserved	N/A
STPL1T Bit 9	**Stop Layer 1 Timer—** Stop Layer 1 timer and capture current COUNT32 value in COUNT32_CAP. Can be cancelled by clearing this bit up to the next positive edge of CKIL.	1 - Request {OFF, MEASDONE} -> L1TOFF transition 0 - Nominal
SLEEP Bit 8	**Sleep—** Enter Deep Sleep. Deep Sleep can be cancelled up until the next positive edge of CKIL by clearing this bit. However, at the positive edge of CKIL, the DSM is committed to a sleep/wakeup sequence. This bit self-clears at the subsequent negative edge of CKIL (when the DSM transitions into the WAIT4SLEEP state).	1 - Request L1TOFF -> WAIT4SLEEP transition 0 - Nominal
MEAS Bit 7	**Measurement—** Start clock-clock drift measurement. Measurement can be cancelled up until the next positive edge of CKIL by clearing this bit. This bit self-clears on the subsequent negative edge of CKIL (when the DSM transitions into the MEASURING state).	1 - Request OFF -> MEASURING transition 0 - Nominal
OFF Bit 6	**Off—** Turn off the DSM. The transition to the OFF state can be cancelled up until the next positive edge of CKIL by clearing this bit.This bit self-clears on the subsequent negative edge of CKIL when the DSM transitions into the OFF state.	1 - Request {MEASURING, MEASDONE, L1TOFF, LOCKED} -> OFF transition 0 - Nominal
RST_ POS Bit 5	**POSITION Counter Reset—** This bit self-clears on the next positive edge of MUX_CLK.	1 - Reset POSITION counter 0 - Nominal

Table 52-13. CONTROL1 Description (Continued)

Name	Description	Settings
RST_ REFCNT Bit 4	**REFCOUNT Counter Reset—** This bit self-clears on the next positive edge of MUX_CLK.	1 - Reset REFCOUNT counter 0 - Nominal
RST_ CNT32 Bit 3	**COUNT32 Counter Reset—** This bit self-clears on the next positive edge of MUX_CLK.	1 - Reset COUNT32 counter 0 - Nominal
SM_ RST Bit 2	**State Machine Reset—** This bit self-clears on the next positive edge of MUX_CLK.	1 - Reset state machine to OFF state 0 - Nominal
CB_ RST Bit 1	**Counter Block Reset—** This bit self-clears on the next positive edge of MUX_CLK.	1 - Reset POSITION, REFCOUNT, COUNT32 counters 0 - Nominal
SRST Bit 0	**Software Reset —** Software reset of the entire DSM module. This bit self-clears on the next positive edge of MUX_CLK.	1 - Reset DSM module 0 - Nominal

NOTE:

All bits in this register are self-clearing.

This register contains the enables for the REFCOUNT and COUNT32 register. For these bits,

1= enable, 0 = disable

CTREN Counter Enable Register **Addr $2484_B028**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
															EN_REFCNT	EN_CNT32
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 52-14. CTREN Description

Name	Description	Settings
Bits 31-2	Reserved	N/A
EN_REFCNT Bit 1	**Enable REFCOUNT—** This bit enables the REFCOUNT counter. This bit self-clears when the DSM transitions from the MEASURING state to the MDONE state.	1 - REFCOUNT counter enabled 0 - REFCOUNT counter disabled
EN_CNT32 Bit 0	**Enable COUNT32— T**his bit enables the COUNT32 counter. This bit self-clears when the DSM transitions to the OFF state.	1 - COUNT32 counter enabled 0 - COUNT32 counter disabled

STATUS Status Register **Addr $2484_B02C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									MUX_CLK	WAKEUP	DS_ACK	DSM_INT_HOLDOFF	DS_REQ	DIS_LIT	CLK_SEL	STBY
TYPE	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Table 52-15. STATUS Description

Name	Description	Settings
Bits 31-8	Reserved	N/A
MUX_CLK Bit 7	**MUX_CLK —** mux_clk status register	0 - Mux_clk running at CKIH_CLK frequency 1 - Mux_clk running at CKIL_CLK frequency
WAKEUP Bit 6	**Wake-Up—** ARM interrupt wakeup signal from ARM Platform.	0 - Wakeup, INT_HANDLER has pending interrupt 1 - Nominal
DS_ ACK Bit 5	**Deep Sleep Acknowledge—** gearshifter status	0 - nominal (running from ccm_dsm_ckih_clk frequency) 1 - in Deep Sleep (running from CKIL frequency)
DSM_INT_HOLDOFF Bit 4	**Deep Sleep Module Interrupt Holdoff—** Block interrupts from waking up MCU due to external interrupt.	0 - nominal 1 - block external interrupt
DS_ REQ Bit 3	**Deep Sleep Request—** Request to Clock Manager to enter/leave Deep Sleep.	0 - Run from ccm_dsm_ckih_clk frequency 1 - Run from CKIL frequency
DIS_ LIT Bit 2	**Disable Layer 1 Timer—** Disable Layer 1 Timer timing block.	0 - Layer 1 Timer enabled 1 - Layer 1 Timer disabled
CLK_SEL Bit 1	**Clock Select—** Select line to switch PAT_REF between crystal and Reference DPLL. Note that when the Master Enable is turned on(set to a 1)for the first time, CLK_SEL will become a 1.	0 - crystal 1 - Reference DPLL
STBY Bit 0	**Standby—** Select line to put power control IC into low-power standby mode.	0 - Nominal 1 - Standby

NOTE:

DS_ACK, WAKEUP, and MUX_CLK are read-only bits.

NOTE:

MCU writes to STATUS register bits (except DS_ACK, WAKEUP) override values assigned by DSM.

STATE State Register **Addr $2484_B030**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
													STATE[3:0]			
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 52-16. STATE Description

Name	Description	Settings
Bits 31-4	Reserved	N/A
STATE[3:0] Bits 3-0	**State—** Current state of the DSM.	0 - DSM_OFF 1 - MEASURING 2 - MEASDONE 3 - L1TOFF 4 - WAIT4SLEEP 5 - DEEPSLEEP 6 - WAIT4WARM 7 - WARM 8 - LOCKED 9-15 - INVALID STATES

NOTE:

Unlike the STATUS register, the STATE register is READ ONLY. Only the DSM can cause transitions in the State Machine. The MCU can request transitions via the OFF, MEAS, STPL1T, and SLEEP bits in the control register.

INT_STATUS — Interrupt Status Register — Addr $2484_B034

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
							WKTIME_INT	SLEEP_INT	RWK	NGK_INT	PGK_INT	STPL1T_INT	RSTRT_INT	LOCK_INT	WTIME_INT	MDONE_INT
TYPE	r	r	r	r	r	r	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 52-17. INT_STATUS Description

Name	Description	Settings
Bits 31-10	Reserved	N/A
WKTIME_INT Bit 9	**Waketime Interrupt —** Deepsleep time interval completed (i.e. WAKETIME timed out and entered WARMTIME state). This bit is Write-1-to-Clear.	1 - WKTIME_INT active 0 - Nominal
SLEEP_INT Bit 8	**Sleeptime Interrupt —** Begin clock gearshifting sequence. (ie. entered DEEPSLEEP state). This bit is Write-1-to-Clear.	1 - SLEEPTIME_INT active 0 - Nominal
RWK Bit 7	**Reason for Wakeup—** This bit is set when the DSM responds to an external interrupt, but it is NOT an interrupt source. The interrupt handlers for the STPL1T_INT and WTIME_INT can use this bit to determine if an external interrupt caused a WAIT4SLEEP -> L1TOFF transition or early wakeup, respectively. This bit is Write-1-to-Clear.	1 - DSM woke up due to external interrupt during Deep Sleep 0 - DSM woke up without any external interrupts during Deep Sleep
NGK_INT Bit 6	**Negative Interrupt—** This interrupt is generated at the negative edge of CKIL.	1 - NGK_INT active 0 - Nominal
PGK_ INT Bit 5	**Positive Interrupt—** This interrupt is generated at the positive edge of CKIL.	1 - PGK_INT active 0 - Nominal
STPL1T_INT Bit 4	**Stopped Layer 1 Timer Interrupt—** Layer 1 Timer stopped interrupt (i.e. enter L1TOFF state from MEASDONE).	1 - STPL1T_INT active 0 - Nominal
RSTRT_INT Bit 3	**Restarted Interrupt—** Layer 1 Timer restarted interrupt.	1 - RSTRT_INT active 0 - Nominal
LOCK_ INT Bit 2	**Locked Interrupt—** Ref DPLL locked interrupt from main path (i.e. enter LOCKED state from WARM state).	1 - LOCK_INT active 0 - Nominal

Table 52-17. INT_STATUS Description (Continued)

Name	Description	Settings
WTIME_INT Bit 1	**Warm-Up Time Interrupt—** Warmup interrupt (i.e. leave WAIT4WARM state).	1 - WTIME_INT active 0 - Nominal
MDONE _INT Bit 0	**Measurement Done Interrupt—** Measurement complete interrupt (i.e. leave MEASURING state).	1 - MDONE_INT active 0 - Nominal

NOTE:

All interrupt status bits are Write-1-to-Clear. All interrupt sources are masked then OR'd to form the DS_INT_B interrupt line to the AITC.

The bits in this register are the masks for the corresponding interrupts in the INT_STATUS register. 0 = masked, 1 = unmasked.

MASK Mask Register **Addr $2484_B038**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
							WKTIME_INT_M	SLEEP_INT_M		NGK_INT_M	PGK_INT_M	STPL1T_INT_M	RSTRT_INT_M	LOCK_INT_M	WTIME_INT_M	MDONE_INT_M
TYPE	r	r	r	r	r	r	rw	rw	r	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 52-18. MASK Description

Name	Description	Settings
Bits 31-10	Reserved	N/A
WKTIME_INT_M Bit 9	WKTIME_INT Mask— Determines if an interrupt is generated on the entry of the WARMTIME state.	1 - WKTIME_INT enabled 0 - WKTIME_INT disabled
SLEEP_INT_M Bit 8	SLEEP_INT Mask— Determines if an interrupt is generated on entry of the DEEPSLEEP state.	1 - SLEEP_INT enabled 0 - SLEEP_INT disabled
Bit 7	Reserved	N/A
NGK_INT_M Bit 6	NGK_INT Mask— Determines if an interrupt is generated on the negative edge of CKIL.	1 - NGK_INT enabled 0 - NGK_INT disabled
PGK_ INT_M Bit 5	**PGK_INT Mask—** Determines if an interrupt is generated on the positive edge of CKIL.	1 - PGK_INT enabled 0 - PGK_INT disabled
STPL1T_INT_M Bit 4	**STPL1T_INT Mask—** Determines if an interrupt is generated on entry of the L1TOFF state.	1 - STPL1T_INT enabled 0 - STPL1T_INT disabled
RSTRT_INT_M Bit 3	**RSTRT_INT Mask—** Determines if an interrupt is generated on entry of the DSM_OFF state upon a restart timeout.	1 - RSTRT_INT enabled 0 - RSTRT_INT disabled

Table 52-18. MASK Description

Name	Description	Settings
LOCK_ INT_M Bit 2	**LOCK_INT Mask—** Determines if an interrupt is generated on entry of the LOCKED state.	1 - LOCK_INT enabled 0 - LOCK_INT disabled
WTIME_INT_ M Bit 1	**WTIME_INT Mask—** Determines if an interrupt is generated on entry of the WARM state.	1 - WTIME_INT enabled 0 - WTIME_INT disabled
MDONE _INT_M Bit 0	**MDONE_INT Mask—** Determines if an interrupt is generated on entry of the WARM state.	1 - MDONE_INT enabled 0 - MDONE_INT disabled

52.8.4 Delta-time Parameters

Value of COUNT32 when the Layer 1 timer was disabled

COUNT32_CAP COUNT32 Capture Register **Addr $2484_B03C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
												COUNT32_CAP[20:16]				
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	COUNT32_CAP[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 52-19. COUNT32_CAP Description

Name	Description	Settings
Bits 31-21	Reserved	N/A
COUNT32_CAP[20:0] Bits 20-0	**COUNT32_CAP—** Is a register that captures the value of COUNT32 when the layer 1 timer is stopped.	N/A

Warm Up period in CKIL cycles.

WARMPER Warm-Up Period Register **Addr $2484_B040**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	WARMPER[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 52-20. WARMPER Description

Name	Description	Settings
Bits 31-16	Reserved	N/A
WARMPER [15:0] Bits 15-0	**WARMPER—** Is a register that contains the number of COUNT32 counts to warm up the ccm_dsm_ckih_clk crystal before gearshifting PAT_REF from CKIL frequency to ccm_dsm_ckih_clk frequency on wakeup. This value is added to WAKETIME and the sum written to WARMTIME upon transition to deep sleep. On wakeup from external interrupt, this value is added to the current value of COUNT32 and written to WARMTIME.	ccm_dsm_ckih_clk warm up period in COUNT32 counts (Must be greater than 1)

Reference DPLL lock period in CKIL cycles.

LOCKPER Lock Period Register **Addr $2484_B044**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	LOCKPER[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 52-21. LOCKPER Description

Name	Description	Settings
Bits 31-16	Reserved	N/A
LOCKPER [15:0] Bits 15-0	**LOCKPER—** Is a register that contains the number of COUNT32 counts to allow the Reference DPLL to lock after WARMPER has elapsed. The DSM will assert CLKSEL at the end of LOCKPER. This value is added to WAKETIME+WARMPER and the sum written to LOCKTIME upon transition to deep sleep. On wakeup from external interrupt, this value is added to the current value of COUNT32+WARMPER and written to LOCKTIME.	Reference DPLL lock period in COUNT32 counts (Must be greater than 1)

52.8.5 Word Counter Contents

This counter gives the number of ccm_dsm_ckih_clk cycles elapsed since the first positive edge of CKIL after POSCOUNT was enabled. This counter is enabled by the EN_POS bit in the CONTROL0 register. This counter is free-running when enabled.

NOTE:

POSCOUNT is not valid until the first positive edge of CKIL following assertion of EN_POS, also POSCOUNT stops counting at the first positive edge of CKIL following the deassertion of EN_POS.

POSCOUNT Position Counter Register **Addr $2484_B048**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	POSCOUNT[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 52-22. POSCOUNT Description

Name	Description	Settings
Bits 31-16	Reserved	N/A
POSCOUNT [15:0] Bits 15-0	**POSCOUNT—**This counter gives the number of ccm_dsm_ckih_clk cycles elapsed since the first positive edge of CKIL after POSCOUNT was enabled. This counter is enabled by the EN_POS bit in the CONTROL0 register. This counter is free-running when enabled	N/A

52.8.6 MAGIC Minimum Standby time bugfix

The MAGIC IC used with whitecap 1.0 does not start up correctly if not kept in standby for some minimum period. Hardware has been added to the DSM to disable interrupts for a programmable time after entering deep sleep to prevent an interrupt starting up MAGIC before the minimum standby time has elapsed. The interrupt 'blackout' time is programmable in CKIL clock cycles and can generate a maximum blackout period of 30ms (CKIL @ 32 kHz). One register is added, as detailed below.

MGPER Magic Period Register **Addr $2484_B04C**

BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
EN_MGFX															
TYPE rw	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
						MGPER[9:0]									
TYPE r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 52-23. MGPER Description

Name	Description	Settings
EN_MGFX Bit 31		1 - Prevent MCU interrupts after transitioning into Deep Sleep for MGPER CKIL cycles. 0 - Disable bugfix.
Bits 30-10	Reserved	N/A
MGPER[9:0] Bits 9-0	**Magic Period—** The number of CKIL clock cycles to 'black out' external interrupts after deep sleep starts. The value in MGPER is compared to an internal CKIL counter that starts counting from zero when DS_REQ is asserted. DS_REQ is asserted 3.5 CKIL cycles before STBY, thus the actual period interrupts are disabled after standby starts is MGPER-3.5 CKIL cycles. NOTE: it is the users responsibility to ensure that MGPER is less than (WAKETIME-SLEEPTIME)!	

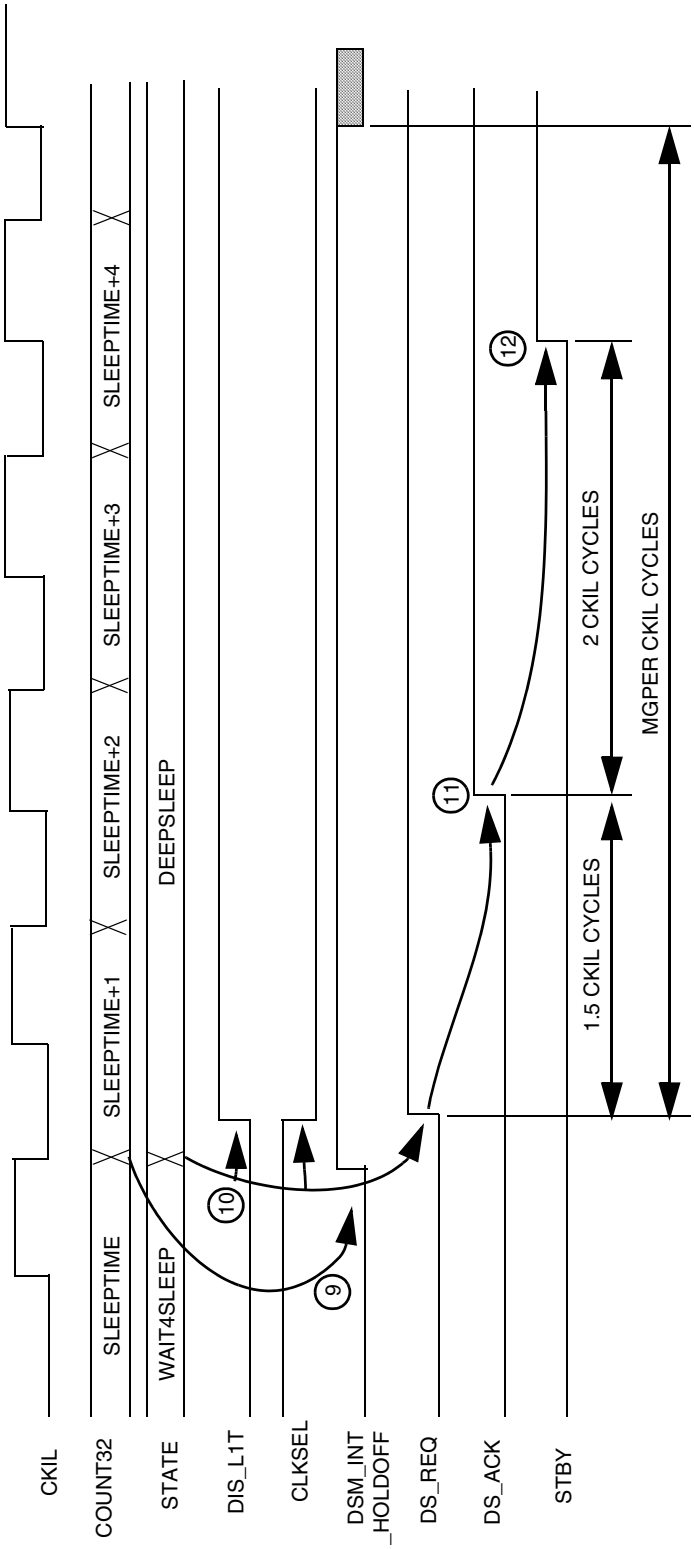


Figure 52-11. Modification to sleep sequence when MAGIC bugfix is enabled

52.9 Appendix

NOTE:

These timing parameters are for GSM. Parameters for the other MA's are TBD.

52.9.1 Timing Relationships

1 quarter-bit = 12 CLKIN cycles
= 0.92308 microseconds

1 frame = 5000 quarter-bits
= 60000 CLKIN cycles
= 147.68 CKIL cycles
= 4.6150 milliseconds

1 CKIL cycle = 33.854 quarter-bits
= 406.25 CLKIN cycles
= 31.250 microseconds

4 CKIL cycles = 1625.0 CLKIN cycles
= 125.00 microseconds

52.9.2 Crystal Requirements

- 26 MHz AT crystal, 40 ppm
- 0.3-0.4 ppm drift over 30 seconds
- < 0.1 ppm drift over 2 seconds
- Warmup time of 10-20 milliseconds (allows processing in 5-10 milliseconds)
- Maximum Reference DPLL lock time of 100 microseconds
- 32 kHz AT crystal, < 0.1 ppm drift over 2 seconds, 40 ppm

52.9.3 Measurement Time Tables

Table 52-24. Measurement Time (msec) vs. DRX mode, Reference Oscillator Accuracy, and Required Timing Slack

DRX MODE	Oscillator Accuracy (ppm)	+/- Timing Slack in Quarter-Bit Counts											
		1	2	3	4	5	6	7	8	9	10	11	12
DRX2 470 msec sleep time	0.05	22.6	11.2	7.5	5.6	4.5	3.7	3.2	2.8	2.5	2.2	2.0	1.9
	0.10	22.7	11.3	7.5	5.6	4.5	3.7	3.2	2.8	2.5	2.2	2.0	1.9
	0.30	24.0	11.6	7.6	5.7	4.5	3.8	3.2	2.8	2.5	2.2	2.0	1.9
	0.50	25.3	11.9	7.7	5.7	4.6	3.8	3.2	2.8	2.5	2.3	2.1	1.9
DRX9 2.12 sec sleep time	0.05	597	257	164	120	95	78.5	66.8	58.2	51.6	46.3	42.0	38.4
	0.10	879	298	180	129	100	81.9	69.3	60.1	53.1	47.5	43.0	39.2
	0.30	-	834	293	178	128	99.5	81.5	69.0	59.9	52.9	47.3	42.8
	0.50	-	-	793	288	176	127	98.9	81.1	68.8	59.7	52.7	47.2

52.9.4 MATLAB Code for Calculating Measurement Time Tables

```
function req_accuracy = gsm_rtc(qbc_slack, ref_accuracy, drx)
%************************************************************************
% GSM_RTC.M
% GSM WCP Real-Time Clock timing accuracy measurements
%
% OPTIONAL
% INPUTS:   QBC_SLACK - maximum uncertainty in Quarter-Bit Periods  VECTOR
%         REF_ACC- reference accuracy    (ppm)          SCALAR
%           DRX-mode     - 2-9                        SCALAR
% OUTPUTS: Display
%         REQ_ACCURACY - Required accuracy
%************************************************************************
%************************************************************************

qbc_period =  1/(13e6/12)  / (1e-6);
frametime = qbc_period*5000*1e-6;

if nargin == 0
                % maximum allowable time uncertainty in usec
                % (1/4 bit to 3 bits)
    qbc_slack = 1:12;
    ref_accuracy = 0.3;
    drx = 9;
end
max_uncertainty = qbc_period*qbc_slack;
total_sleep_time = drx*51*frametime;     % sleep time in sec (1 paging channel)

sleep_clk = 32000;            % Hz
ref_osc = 13e6;         % Hz
clock_jitter = 8e-9;

%-------------------------------------------------------------------------
sleep_period = 1/sleep_clk;   %sec
ref_period = 1/ref_osc;      % sec
sleep_frames = total_sleep_time/frametime;

%-------------------------------------------------------------------------
ref_accuracy = ref_accuracy*1e-6;
max_uncertainty = max_uncertainty*1e-6;  % sec


%************************************************************************
%************************************************************************

n = 1:5:5000;                          % measurement period index
                                         %  (# 32 kHz cycles)
meas_period   = n*sleep_period;    % sec

%-------------------------------------------------------------------------

slip_per = (1 + ref_accuracy);  % percentage slip
slip = 1/(ref_osc*slip_per);  % slip in sec
ref_error = ref_period - slip;  % error per cycle in ref. osc.
```

```
%-------------------------------------------------------------------------

A =  ref_period + 2*clock_jitter;
B =  ref_error/ref_period;

meas_accuracy = A  + B*meas_period;


%-------------------------------------------------------------------------
C = total_sleep_time;
total_error = (meas_accuracy./meas_period)'*C + 3*ref_period;



%*************************************************************************
%          SOLVE FOR REQUIRED MEASURMENT TIME
%*************************************************************************
req_accuracy = (max_uncertainty/total_sleep_time);    %Required Accuracy
D = req_accuracy;
req_meas_time = (A*C)./(D-B*C);      %Required Measurement time


%*************************************************************************
%                     DISPLAY SOLUTION
%*************************************************************************


fprintf('\n\nOscillator Accuracy: %3.2f ppm\n', ref_accuracy*1e6);
fprintf('Sleep Time:          %3.2f sec.\t DRX mode: %1d\n\n', ...
           total_sleep_time,drx);
fprintf('\t QBC \t\t   Required \t\tRequired\tDuty\n');
fprintf('        Slack    \tMeasurement Time\tAccuracy\tCycle\n');
fprintf('\t(QBCs)\t\t     (msec)\t\t (ppm)\n');
for j = 1:length(req_accuracy)
     fprintf('\t  %1d\t\t    % -4.1f\t\t %6.3f\t\t%-5.2f\n', qbc_slack(j), ...
                       req_meas_time(j)*1e3', req_accuracy(j)*1e6, ...
                       (req_meas_time(j)/total_sleep_time)*100);
end


%*************************************************************************
%                       PLOTS
%*************************************************************************
plot(meas_period*1e3,1e6*meas_accuracy./meas_period);
xlabel('Measurement Period (msec)');
ylabel('Measurement Accuracy (ppm)');

%keyboard
```

Figure 52-12.

Chapter 53
Watchdog Timer Module (WDOG)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	03/19/02	Vinay Kumar P P	1. Module made IP Bus compliant 2. Updated as per DDTS TLSbo19137 3. Updated as per DDTS DSPhl12266
0.2	05/03/02	Vinay Kumar P P	1. Updated as per the DDTS TLSbo16551
0.3	05/07/03		Updated for LTE specification release.

53.1 General Overview

The WDOG Timer module protects against system failures by providing a method of escaping from unexpected events or programming errors. Once activated, the timer must be serviced by software on a periodic basis. If servicing does not take place, the timer times out. Upon a time-out, the WDOG Timer module either asserts the wdog_b signal or a system reset signal wdog_rst_b depending on software configuration. The WDOG Timer module also generates a system reset via a software write to the Watchdog Control Register (WCR), a detection of a clock monitor event, an external JTAG reset signal, or if a power-on-reset has occurred. The wdog_b signal is asserted via a software write to the WCR, a detection of a clock monitor event, or upon a watchdog time-out. A table of the Watchdog Timer module's inputs and outputs is shown in Table 53-1. A state machine of the counter operation is shown in Figure 53-3. It only demonstrates the time-out operation

53.1.1 WDOG Module Pin List

Table 53-1 is a list of the WDOG module pins.

Table 53-1. WDOG Module Pin List

Pin Name	Direction	Description	signal_name	envelope_pin
por_b	I	Power-on-reset from DSP		
ccm_reset_b	I	System reset from CCM		
ipg_clk	I	Neptune main clock		
ckil_clk	I	Neptune slow reference clock (CKIL)		
cmon_noclk	I	Clock Monitor event		
ccm_ext_rst_b	I	External reset signal from CCM		
ccm_wclk_mcs	I	mcu_clk (High Speed Test Clk Select)		
ipt_clk_se	I	Scan enable for clock gating cell		
ips_module_en	I	Module select		
ipg_debug	I	DBUG mode enable		
ips_rwb	I	Module read/write signal		
ips_addr[11:1]	I	Module address Bus		
ips_byte_7_0	I	Byte enable for 7-0		
ips_byte_15_8	I	Byte enable for 15-8		
ccm_lpmd[1:0]	I	Low Power Mode		
nmode2	I	JTAG reset signal		
ipt_scan_mode	I	Indicates scan mode		
wdog_rst_b	O	Watchdog Timer underflow reset		
wdog_b	O	Asserted by software, timeout, or clock monitor event		
wdog_oe	O	Wdog_b output enable at pin.		
wdog_por	O	Active high power-on reset.		
ips_rdata[15:0]	O	Read data bus to IP Bus		
ips_wdata[15:0]	I	Write data from IP Bus		
ips_xfr_wait	O	Transfer wait		
ips_xfr_err	O	Transfer error acknowledge		

53.2 Watchdog Timer Operation

53.2.1 Timing Specifications

The Watchdog Timer provides time-out periods from 0.5 seconds up to 64 seconds with a time resolution of 0.5 seconds. It uses the ckil clock as an input to prescalers. The prescalers divide the clock by a fixed value of 16384 (div4 and div4K) to achieve the resolution of 0.5 seconds and a frequency of 2 Hz. The output of the prescaler circuitry is connected to the input of a 7-bit counter to obtain a range of 0.5 to 64 seconds. The user can determine the time-out period by writing to the Watchdog Time-out field (WT[6:0]) in the Watchdog Control Register (WCR).

53.2.2 Watchdog During Reset

All registers are reset to their reset values and the counter is placed in the idle state until the watchdog is enabled. The Watchdog Reset Status Register (WRSR) will contain the source of the reset event.

53.2.3 Watchdog After Reset

The following sub-sections define the Watchdog Timer state after reset.

53.2.3.1 Initial Load

The Watchdog Control Register (WCR) bits WT[6:0], shall be written to prior to enabling the Watchdog. The Watchdog is then enabled by setting the one-time writable Watchdog Enable (WDE) bit in the WCR. The time-out value will be loaded into the counter after the service sequence is written to the Watchdog Service Register (WSR) or after the Watchdog has been enabled. The service sequence is described in Section 53.2.3.3. The counter state machine is shown in Figure 53-3.

53.2.3.2 Countdown

The counter is activated after the Watchdog is enabled and begins to count down from its initial programmed value. If any system errors have occurred which prevents the software from servicing the Watchdog Service Register (WSR), the timer will time-out when the counter reaches zero. If the WSR is serviced prior to the counter reaching zero, the Watchdog will reload its counter to the time-out value indicated by bits WT[6:0] of the WCR and re-start the countdown. A reset will reset the counter and place it in the idle state at any time during the countdown. The counter state machine is shown in Figure 53-3.

53.2.3.3 Reload

The proper service sequence is to write a $5555 followed by a $AAAA to the WSR. In order to reload the counter, the writes must take place within the time-out value indicated by bits WT[6:0] of the WCR. Any number of instructions can be executed between the two writes. This service sequence is also used to activate the counter during the initial load. See Section 53.2.3.1, “Initial Load,” on page 53-3.

If the WSR is not loaded with a $5555 prior to a write of $AAAA to the WSR, the counter will not be reloaded. If any value other than $AAAA is written to the WSR after $5555, the counter will not be reloaded.

53.2.3.4 Time-out

If the counter reaches zero, the Watchdog will output either a system reset or the wdog_b signal depending on the state of the WRE bit in the WCR. A “0” written to the WRE bit will configure the Watchdog to generate a system reset. A “1” will configure the Watchdog to generate the wdog_b signal. The counter state machine is shown in Figure 53-3.

53.2.4 Low Power and DBUG Modes

The Watchdog is affected by the low power STOP, WAIT, DOZE, and DBUG modes. A diagram of low power control is shown in Figure 53-1.

53.2.4.1 WAIT and DOZE and STOP Mode

The Watchdog Timer can be configured for continual operation or be suspended. if the Watchdog low power Enable (WDZST) bit in the WCR is set to ‘0”, Watchdog continue to operation use CKIL clock. If low power Enable (WDZST) bit is set to “1”, then the Watchdog operation will be suspended. Upon exiting low power mode, the Watchdog operation returns to what it was prior to entering the mode.

53.2.4.2 DBUG Mode

The Watchdog Timer can be configured for continual operation or be suspended. If the Watchdog DBUG Enable (WDBG) bit is set to “1” in the Watchdog Control Register (WCR), then the Watchdog Timer module operation will be suspended. At this point, the counter is stopped, but register read and write accesses continue to function normally. Also, while in DBUG mode, the WDE bit one-time-write lock is disabled and the bit can be cleared.

NOTE:

If the WDE bit is cleared while in DBUG mode, it will remain cleared upon exiting DBUG mode. If the WDE bit is not cleared while in DBUG mode, the Watchdog count will continue from its value before DBUG mode was entered.

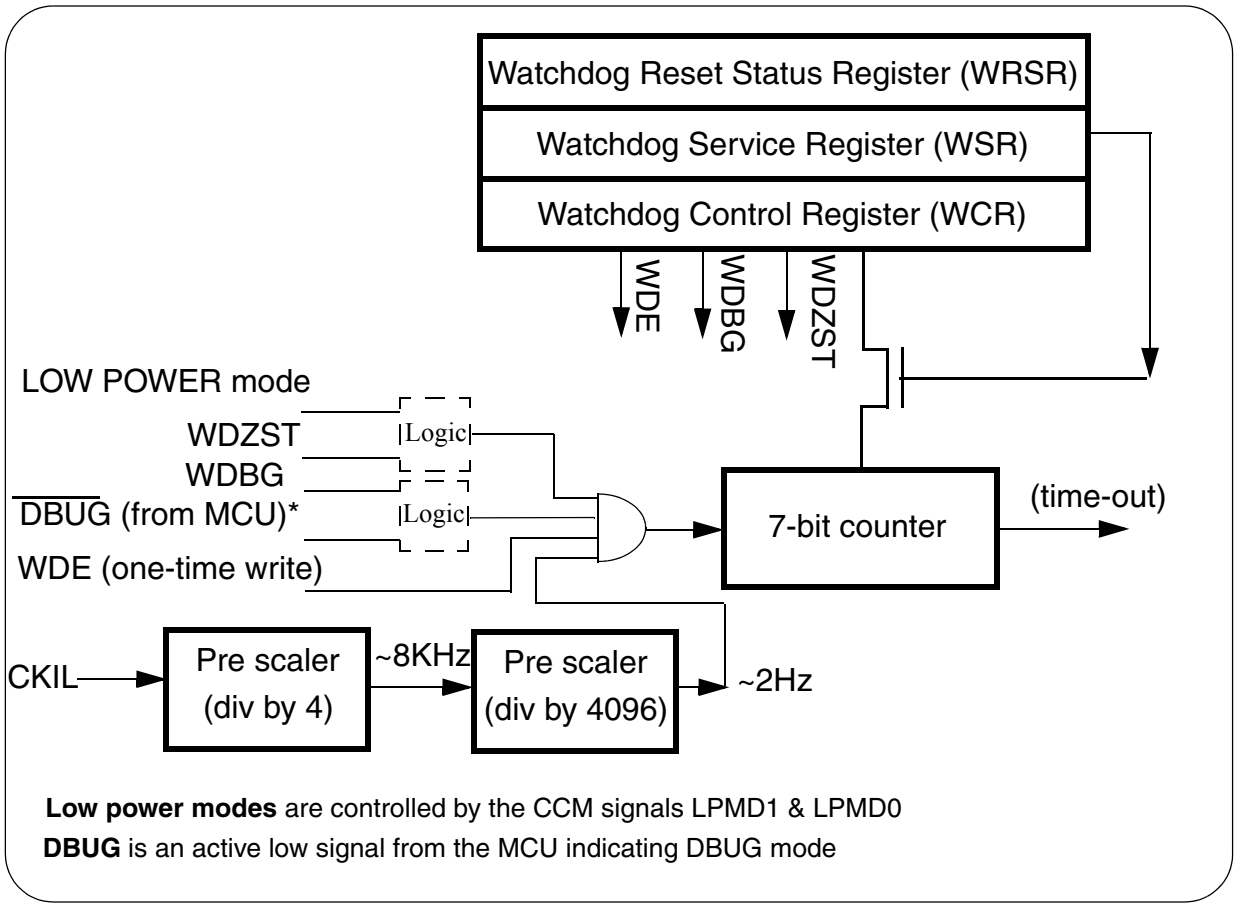


Figure 53-1. Low Power Control

53.2.5 State Machine

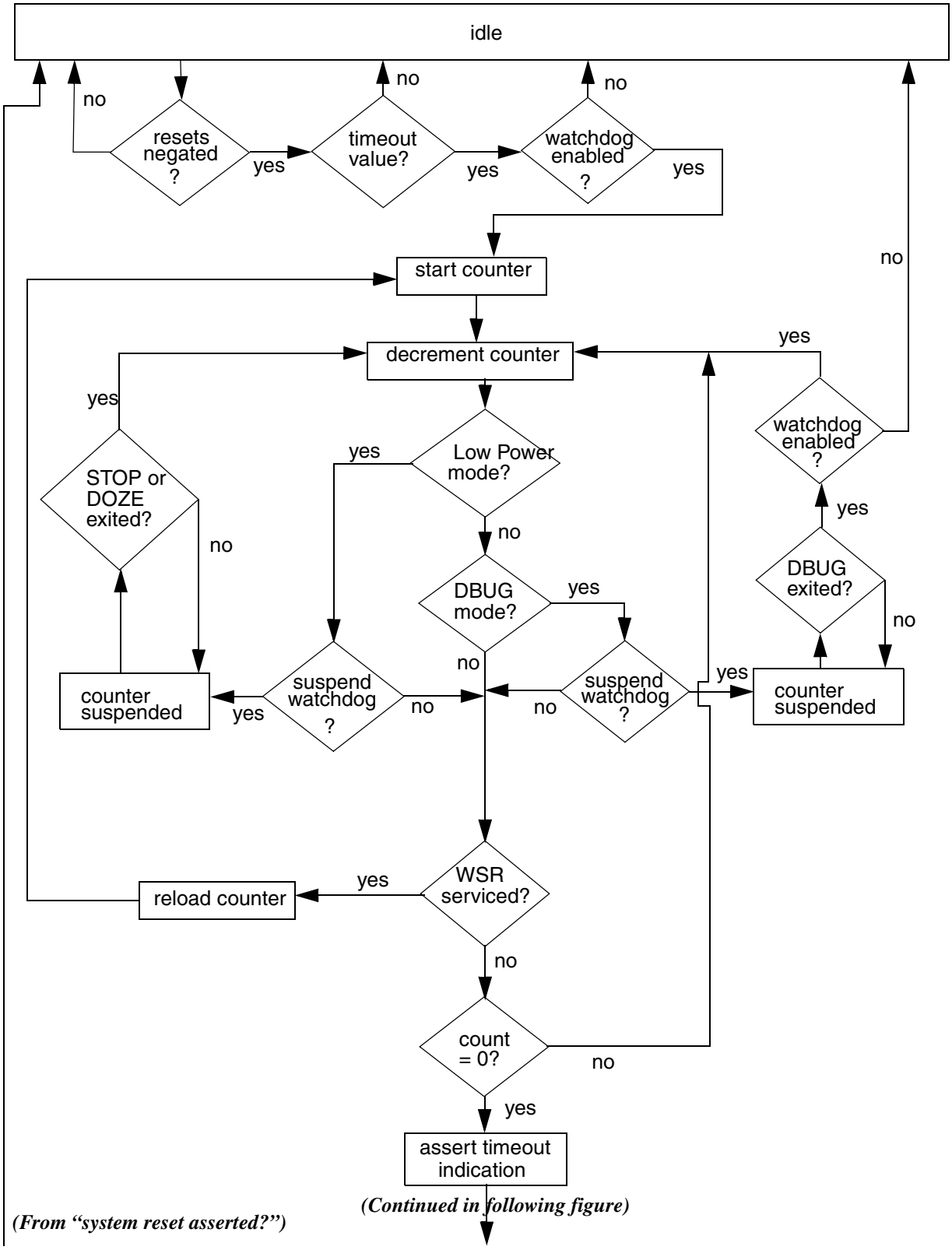
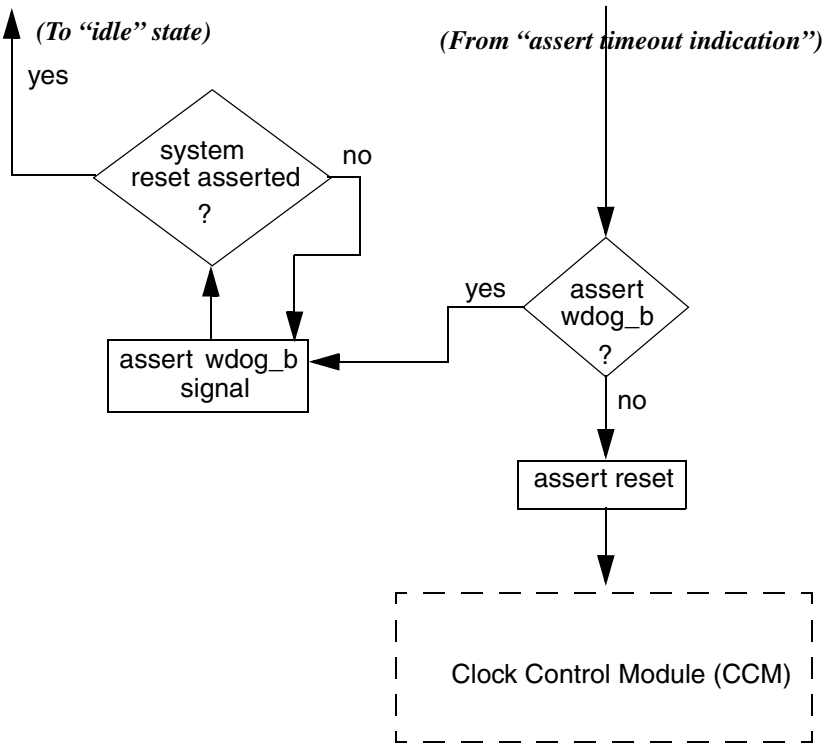


Figure 53-2.



NOTE: A system reset will force the state machine to “idle” at any time during countdown.

Figure 53-3. Counter State Machine

53.3 Watchdog Reset Control

53.3.1 Reset Sources

The watchdog-generated reset signal wdog_rst_b can be asserted through software writes to the Software Reset Signal (SRS) bit of the WCR. It can also be generated as a result of a WDOG time-out, high frequency clock monitor event, an external JTAG reset signal, or a power-on-reset.

The watchdog-generated reset signal wdog_rst_b is an output to the Clock Control Module for system reset generation.

An active high power-on reset “por” is generated by inverting the active low power-on reset input “por_b”.

53.3.2 WDOG_B Operation

Wdog_b can be asserted through software writes to the WDOG_B Assertion (WDA) bit of the WCR. It can also be generated as a result of a WDOG time-out or a high frequency clock monitor event.

The WDOG pin is an I/O pad. After reset, Can config the bit WOE in WCR to control the direction of this pad.

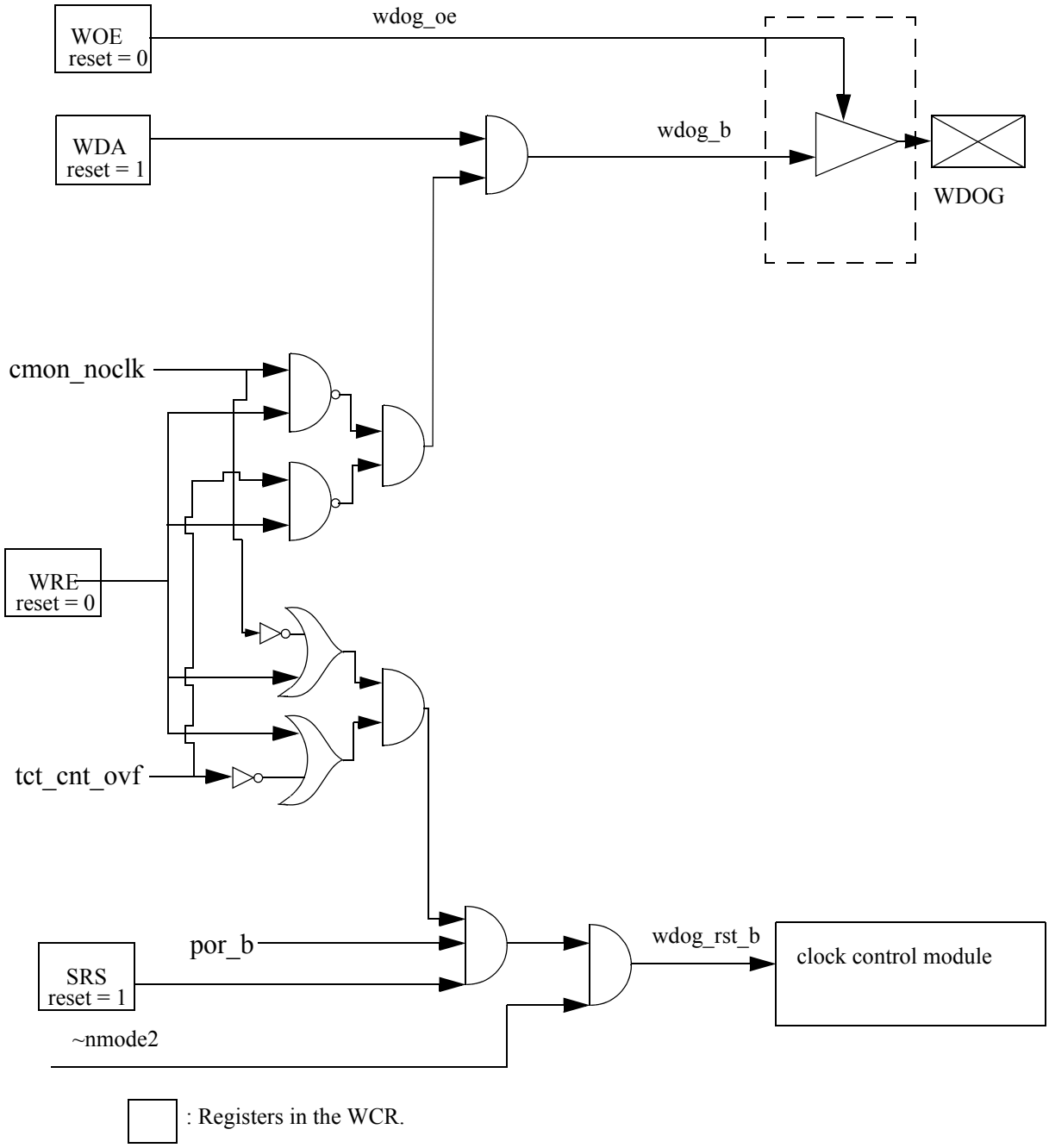


Figure 53-4. WDOG and wdog_rst_b Operation

53.4 Clock Monitor

A clock monitor event results in the wdog-generated signals wdog_rst_b or wdog_b being asserted depending on the WRE bit in the WCR. See the following Figure 53-5.

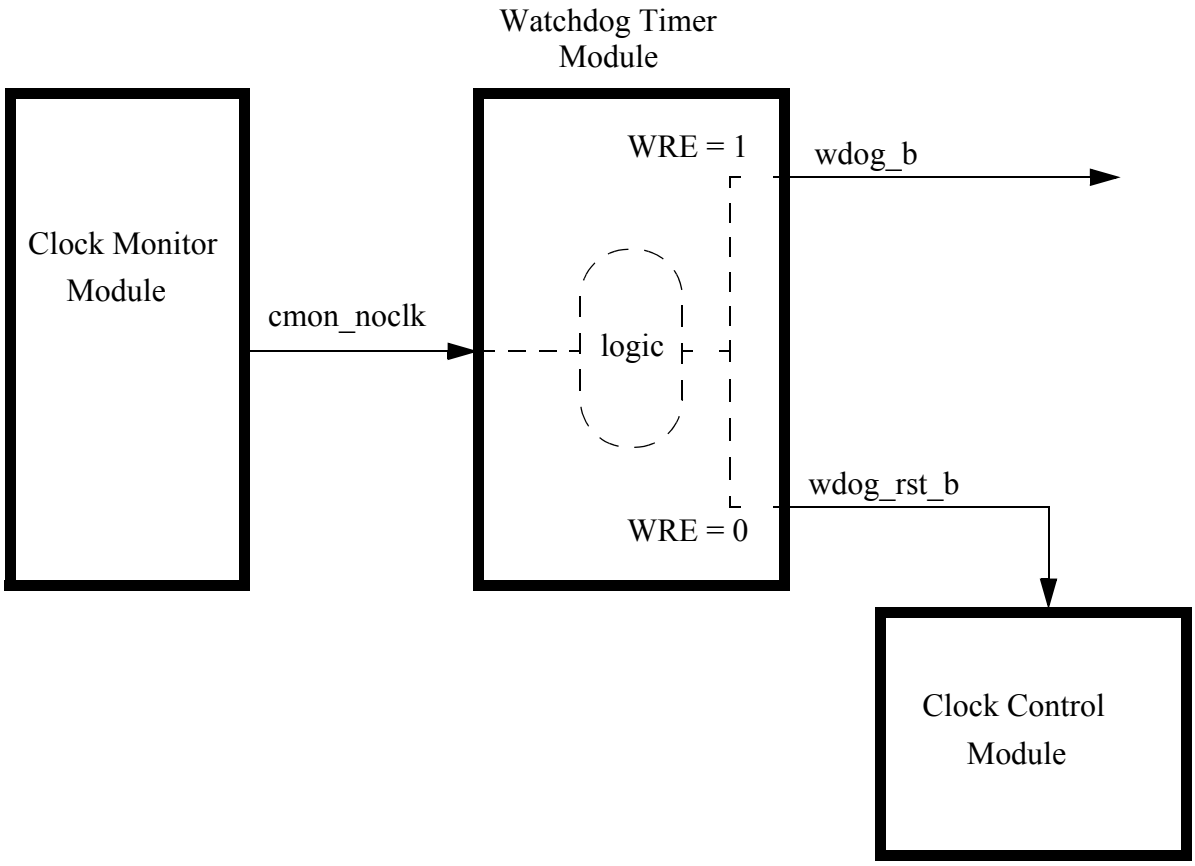


Figure 53-5. Cmon_noclk

53.5 Watchdog Programming Model

The Watchdog Timer has three registers in its programming model: Watchdog Control Register (WCR), Watchdog Service Register (WSR), and Watchdog Reset Status Register (WRSR).

Figure 0-13.

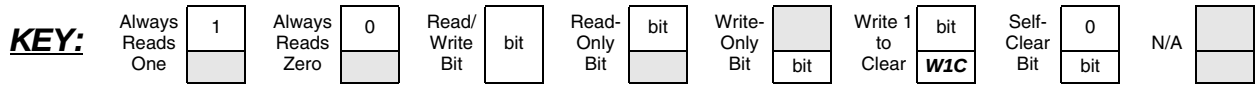


Table 53-2. Watchdog Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WCR ($2484_9000)	R	WT[6:0]							0	0	WOE	WDA	SRS	WRE	WDE	WDBG	WDZST
	W																
WSR ($2484_9002)	R	WSR[15:0]															
	W																
WRSR ($2484_9004)	R	0	0	0	0	0	0	0	0	0	0	JRST	PWR	EXT	CMON	TOUT	SFTW
	W																

53.5.1 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various WDOG registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

The WCR is a 16-bit read/write (byte writable) register. It controls the Watchdog operation. All bits except for bits[5:4] are cleared during reset. Bits[5:4] are set to “1” during reset.

WCR Watchdog Control Register **Addr $2484_9000**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	WT[6:0]									WOE	WDA	SRS	WRE	WDE	WDBG	WDZST
TYPE	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

Table 53-3. WCR Description

Name	Description	Settings
WT[6:0] Bits 15-9	**Watchdog Timeout Field —** This 7-bit field contains the time-out value and is loaded into the Watchdog counter after the service routine has been performed. After reset, WT[6:0] must be written before enabling the Watchdog.	Set to desired timeout value.
Bits 8-7	**Reserved —** These bits are reserved for future expansion.	Always read as zero and must be written with zeros for future compatibility.
WOE Bit 6	**Wdog_b Output Enable—** Determines if the WDOG pin is configured as an output or is input when GPIO gives WDOG ability to control this pin’s direction after reset.	0 = Pin is tri-stated 1 = Pin configured as output
WDA Bit 5	**Wdog_b Assertion —** Controls the software assertion of the wdog_b signal.	0 = Assert wdog_b output 1 = No effect on system
SRS Bit 4	**Software Reset Signal —** Controls the software assertion of the WDOG-generated reset signal. Note: ***This bit automatically resets to “1” after it has been asserted to “0”.***	0 = Assert system reset signal 1 = No effect on system
WRE Bit 3	**Wdog_b or wdog_rst_b Enable —** Determines if the watchdog will generate the reset signal or wdog_b upon a watchdog timeout or clock monitor event. ***Write once-only.***	0 = Generate a reset signal 1 = Generate wdog_b
WDE Bit 2	**Watchdog Enable —** Enables or disables the WDOG module. ***enable once-only.***	0 = Disable Watchdog 1 = Enable Watchdog
WDBG Bit 1	**Watchdog DBUG Enable —** Determines the operation of the WDOG module during DBUG mode. ***Write once-only.***	0 = Continue timer operation 1 = Suspend watchdog timer
WDZST Bit 0	**Watchdog Low Power —** Determines the operation of the WDOG module during Low power modes. ***Write once-only***.	0 = Continue timer operation 1 = Suspend watchdog timer

Note : Functionality of WOE bit with software reset

When the source for ccm_reset_b was due to software reset , then the WOE bit will retains its value prior to assertion of software reset.For all other resets the WOE bit is cleared, i.e output will be tri-stated.

When enabled, the Watchdog requires that a service sequence be written to the Watchdog Service Register (WSR) as described in Table 53-4

WSR Watchdog Service Register **Addr $2484_9002**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	WSR[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 53-4. WSR Description

Name	Description	Settings
WSR[15:0] Bits 15-0	**Watchdog Service Register —** This 15-bit field contains the watchdog service sequence. **Note:** Both writes must occur in the order listed prior to the time-out, but any number of instructions can be executed between the two writes.	The service sequence must be performed as follows: a. Write $5555 to the Watchdog Service Register (WSR). b. Write $AAAA to the Watchdog Service Register (WSR)

The WRSR is a read-only register which records the source of the RESET_OUT event. It is not cleared by reset. It records the source of the RESET_OUT event. Therefore, one and only one bit in the WRSR will always be asserted high.

RESET_OUT can be generated by the following sources as listed in priority from highest to lowest: Power-on reset, External reset, Clock Monitor event, Watchdog Timeout, and Software reset.

WRSR Watchdog Reset Status Register **Addr $2484_9004**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
											JRST	PWR	EXT	CMON	TOUT	SFTW
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?

Table 53-5. WRSR Description

Name	Description	Settings
Bits 15-6	**Reserved bits —** These bits are read as 0 and should be written as 0 for future compatibility.	N/A
JRST **Bit 5**	Jtag Reset - Indicates whether the reset was a result of JTAG reset.	0 = Reset is not a result of Jtag reset. 1 = Reset is a result of JTAG reset.
PWR Bit 4	**Power-On Reset —** Indicates whether the reset was a result of a power-on reset.	0 = Reset is not a result of a power-on reset 1 = Reset is a result of a power-on reset
EXT Bit 3	**External Reset —** Indicates whether the reset was a result of an external reset.	0 = Reset is not a result of an external reset 1 = Reset is a result of an external reset
CMON Bit 2	**Clock Monitor—** Indicates whether the reset was a result of a clock monitor event.	0 = Reset is not a result of clock monitor event 1 = Reset is a result of clock monitor event.
TOUT Bit 1	**Time-out —** Indicates whether the reset was a result of WDOG time-out.	0 = Reset is not the result of WDOG time-out 1 = Reset is the result of a WDOG time-out
SFTW Bit 0	**Software Reset —** Indicates whether the reset was a result of a software reset.	0 = Reset is not a result of a software reset 1 = Reset is a result of a software reset

Note : A write to the WRSR register will generate an ips_xfr_err.

Chapter 54
GPRS Encryption Module (GEM)

Revision History

Version	Date	Author	Notes
Revision 0.1	October 21, 1998	Bill Getka	Preliminary Draft
Revision 0.2	2/14/99	Bill Getka	Reworked flow diagram. Moved INPUT to registers. Added restriction that frames must start on 32 bit boundaries and that they must reserve a block that ends on a 32 bit boundary. Also added restriction that all of header and data must be present before processing can begin.
Revision 0.3	3/5/99	Anna Yan	Pasted in PATRIOT GEM Specification Revision 1.0 (author: Bill Getka). Added the requirements and modifications paragraph for RAINBOW.
Revision 0.4	9/14/99	Bill Getka	Corrected document for Rainbow system
Revision 1.0	9/30/99	Bill Getka	Removed MIN_ACCESS_SIZE bit and cleaned up visual look of document.
Revision 1.1	11/5/99	Bill Getka	Removed support for byte and half-word register access since it is not supported in Rainbow at a system level. Fixed error in description of DIR bit.
Revision 1.2	1/31/00	Bill Getka	Add debug control bit.
Revision 1.3	2/18/00	Bill Getka	Add GEA2 bit.
Revision 1.4	8/14/00	Bill Getka	Fix address offsets in figures Must clear FRAME ERROR prior to GEMDONE bit Add note about setting STOP bit prior to hitting ABORT or SWRST Add section about known system issue
Revision 1.5	11/02/00	Bill Getka	Updates for Rainbow production part. See Section 54.4, “Summary of Changes between the Neptune and Neptune LTS Parts,” on page 54-15.
Revision 1.6	12/07/00	Bill Getka	Replaced proposed break at byte functionality with automatic linked list processing option.

Revision History

Version	Date	Author	Notes
Revision 1.7	12/13/00	Bill Getka	Again reworked linked list processing based on inputs from software community. Also reworked register section to put new bits in a separate register. Also redefined START_BIT field as BIT_OFFSET and added LSB_FIRST bit. Clarified meaning of COUNT and FRESH bit definitions relative to f8/f9 specifications.
Revision 1.8	12/18/00	Bill Getka	Redefined register map to have CTL register on Neptune match Rainbow.
Revision 1.9	1/02/01	Bill Getka	Defined maximum number of potential segments and maximum size of each segment when doing linked list processing. Added information on time it takes to do f8, f9, and DMA processing.
Revision 2.0	2/01/01	Bill Getka	Expand BIT_LENGTH to 26 bits and IMAC to 64 bits to support hash of large memories.
Revision 2.1	7/26/01	Tom Hueske	Modify to reflect GEM for Patriot Indy by removing f8/f9 and burst related information.
Revision 2.2	10/16/01	Bill Getka	Bring in INDY variables and paragraph formats Fix address in WRITE_FRAME_START bit description
Revision 2.2	10/16/01	Bill Getka	Bring in Hip7 variables and paragraph formats Fix address in WRITE_FRAME_START bit description
Revision 2.3	2/26/02	Bill Getka	Fix FRAME_LENGTH and CTL2 typos
Revision 3.0	2/25/02	Chris Daniels	Updated for Neptune LTS
Revision3.1	7/19/02	Chris Daniels	Fix a typo in register block
3.2	05/07/03		Updated for LTE specification release.

54.1 Module Overview

The GPRS Encryption Module (GEM) is a hardware accelerator designed to assist in the processing of General Packet Radio Service (GPRS) data packets. The GEM supports both GEA-1 and GEA-2 algorithms for GPRS ciphering as well as the ability to generate or verify the 24 bit Frame Check Sequence (FCS) that goes with each frame.

The GEM is configured to be a secondary bus master to the ARM7TDMIS (ARM7) core. The GPRS data to be manipulated can be placed in any RAM in the system. Control and status information is passed back and forth from the GEM to the micro controller via registers accessible over the micro controller's peripheral bus. In addition, there is a single interrupt from the GEM to the micro controller to indicate the completion of selected tasks. The register interface with the micro controller is a 32 bit connection via the IP bus, as described in the reference document *R-AHB IP Interface (AIPI)*. The data interface is a 32 bit connection via the secondary bus master interface on the ARM7.

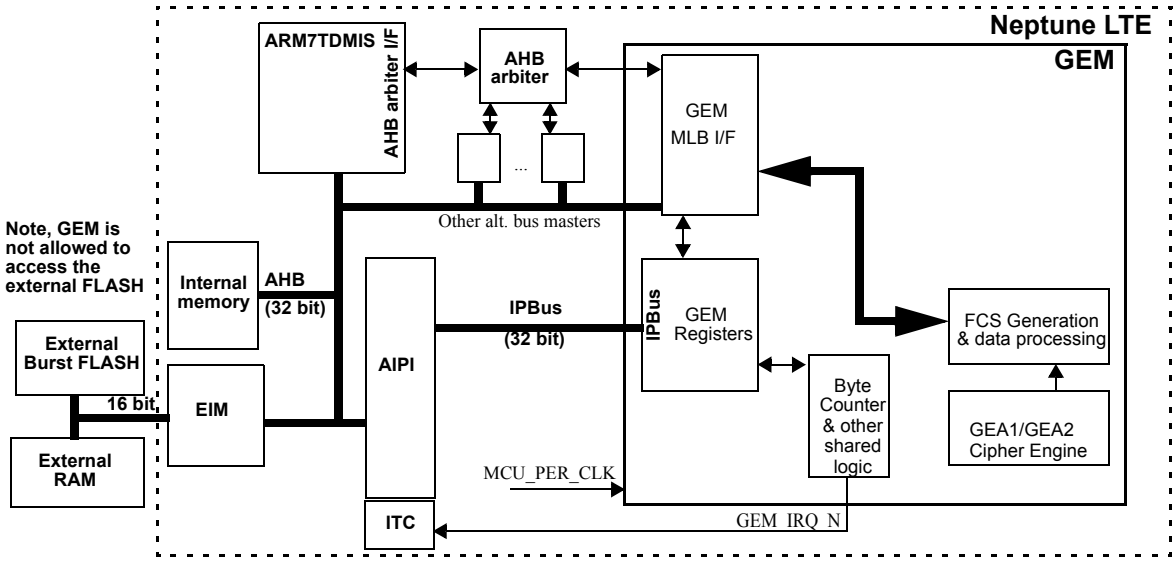


Figure 54-1. Top Level GEM Diagram

54.2 GEM Functional Description

The GEM can be configured via its control registers to do one of two kinds of processing: GPRS Frame processing (where it will do both the FCS and ciphering at the same time) or simple DMA data movement with no processing of the data. The PROCESSING_MODE bits in the GEM's CTL2 register select which type of processing will be done. The GEM can be configured for only one such operation at a time.

54.2.1 GPRS Processing Functional Description

The GPRS Encryption Module accelerates two computationally intensive tasks associated with GPRS LLC data frames: Frame Check Sequence (FCS) generation and data ciphering. The FCS generation is based on a 24-bit cyclic redundancy check (CRC) that is calculated over a selected portion of the GPRS LLC frame (either header or header & data). The generation function supplied by the GEM can be used to either add an FCS to an outgoing frame on uplink or verify an incoming frame on downlink. The ciphering of data is done as specified by the GPRS Encryption Algorithm (GEA-1/GEA-2) specifications and only on the data portion of an GPRS frame if at all. A single control bit changes whether the engine is configured for the encryption of uplink data or decryption of downlink data. A separate control bit indicates whether it should be GEA-1 or GEA-2. When the FCS and ciphering functions are being done to the same section of the GPRS frame, the two functions are done in parallel to speed processing.

54.2.1.1 Frame Check Sequence (FCS) Generation/Verification Process

In order to detect bit errors in the transmission of GPRS data, a 24 bit Frame Check Sequence (FCS) is added to each frame. It is based on a 24 bit CRC code specified in section 5.5 of GSM 4.64 (see references in Section 54.6.2, "Reference Documents."). The FCS is applied before data is ciphered on the transmission side and after deciphering at the receiving side. The FCS can be applied to either the entire frame (protected mode) or just to the header (unprotected mode). The PM (Protected Mode) bit in the CTL register controls whether the GEM is in protected or unprotected mode. When in unprotected mode, it is sometimes necessary to add the first few bytes of the data section to the FCS. In GSM 4.64, this number of bytes is referred to as the N202 length. This length can be specified using the N202_LENGTH field in the GEM's control register.

The GEM can be used to generate an FCS for a frame to be transmitted or verify a frame that has been received. The DIR (Direction) bit in the CTL register controls whether the FCS is generated or verified. When generating an FCS, the GEM will place the resulting FCS in the three octets directly following the data frame itself (see Section 54.3.2, “Parameters for GPRS LLC Frame Location in RAM.” on how the data frame is organized). As described in the GSM 4.64 specification, the FCS will be stored according to the format shown in Table 54-1. It will be ciphered prior to being stored if the E and CIPHER_FCS control bits are both set in the GEM’s CTL register.

Table 54-1. Format of the FCS Field within the RAM

Octet Number	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Octet 1	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}
Octet 2	2^{8}	2^{9}	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}
Octet 3	2^{0}	2^{1}	2^{2}	2^{3}	2^{4}	2^{5}	2^{6}	2^{7}

For FCS verification, the GEM generates an FCS on the selected section of the GPRS frame and then includes the three octets following the data frame (the received FCS) into the FCS polynomial. Note, the received FCS octet will be deciphered prior to use if the E and CIPHER_FCS bits in the GEM’s CTL register are set. If the resulting polynomial does not match the expected constant (see GSM 4.64), the FCS_ERROR flag in the STAT register will be set. The FCS_ERROR flag does not generate an interrupt since it will be set at the same time the GEMDONE flag, which can cause an interrupt, is set. Software should always check the FCS_ERROR flag when the GEMDONE flag in the STAT register becomes set. When verifying a data frame, the FCS calculated by the GEM will not be stored into the RAM.

See Section 54.3, “GEM Data Interface.” for more information on how the GEM is given the RAM location of the frame to be processed.

54.2.1.2 GPRS Encryption/Decryption Process

The ciphering of GPRS data frames is over the data section of the GPRS frame as well as the FCS if the CIPHER_FCS bit is set. The E (Encryption) bit in the CTL register controls if ciphering is enabled. The GEA2 bit in the CTL register selects whether the GEA1 or GEA2 algorithm is used. In either case, there are three inputs into the ciphering algorithm in addition to the data itself: the 64 bit secret key (Kc), the 32 bit message key (INPUT), and a single direction bit (DIR). The Kc parameter is stored in the Kc register set (two 32 bit registers). The INPUT parameter is stored in the INPUT register. The DIR bit is a single bit in the CTL register. The method for determining the correct value to use for the Kc, INPUT, and DIR parameters is given in GSM 4.64 (see references in Section 54.6.2, “Reference Documents.”).

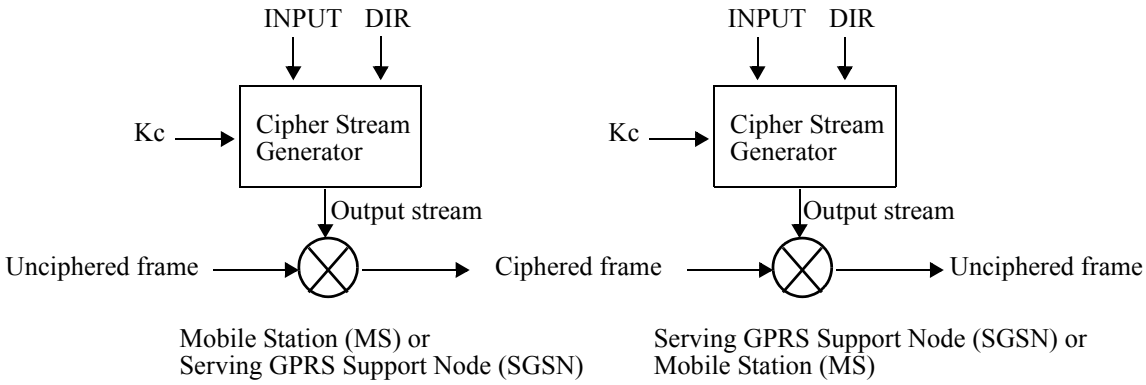


Figure 54-2. Pictorial View of GPRS Ciphering

As Figure 54-2 shows, the GPRS Encryption Algorithms (GEA-1/GEA-2) generate a bit stream that is then used to modify the incoming frame via an XOR function. Therefore, ciphering / deciphering of the frames does not change the number of octets in the frame, only the values in those octets.

See Section 54.3, "GEM Data Interface." for more information on how the GEM is given the RAM location of the frame to be processed.

54.2.1.3 Flow of GRPS Processing

The type of GPRS processing the GEM module will do depends on the parameters Direction (DIR), Encryption Enable (E), and Protected Mode (PM). These three parameters are taken from the GPRS specifications themselves. They have the same meaning and function to the GEM module as they do in the GPRS specifications. Section 54.6.1, "Detailed Register Descriptions." gives detailed register descriptions of these various bits.

Note that since the GEM is designed to work only in a Mobile Station (MS) and not in a Serving GPRS Support Node (SGSN), the DIR parameter is interpreted from the MS point of view. That is, in the GPRS specification, a DIR of 0 is to imply GPRS data going from MS to SGSN whereas a DIR of 1 is to imply GPRS data going from SGSN to MS. For the MS point of view, this means that a DIR of 0 implies encrypting data if needed and generating an FCS. A DIR of 1 implies decrypting data if needed and verifying an FCS.

Table 54-2 gives a summary of the actions taken by the GEM based on the DIR, E, and PM bits.

Table 54-2. Summary of the GEM Function Controls

DIR	E	PM	Description of GEM function to be undertaken when START bit is set
0	0	0	FCS will be generated over the header & N202 bytes of data. No encryption of data is done.
0	0	1	FCS will be generated over the header and all data. No encryption of the data is done.
0	1	0	FCS will be generated over the header & N202 bytes of data. Encryption of data will be done.
0	1	1	FCS will be generated over the header and all data. Encryption of the data will be done.
1	0	0	FCS will be verified over the header & N202 bytes of data. No decryption of data is done.
1	0	1	FCS will be verified over the header and all data. No decryption of the data is done.
1	1	0	FCS will be verified over the header & N202 bytes of data. Decryption of data will be done.

Table 54-2. Summary of the GEM Function Controls

DIR	E	PM	Description of GEM function to be undertaken when START bit is set
1	1	1	FCS will be verified over the header and all data. Decryption of the data will also be done.

The flow diagram shown in Figure 54-3 shows the flow that the GEM will follow to process a GPRS LLC frame based on the settings of the various parameters. Before this flow begins, it is assumed that the software has properly initialized the Kc and INPUT keys as well as the FRAME_START, HEADER_LENGTH, and FRAME_LENGTH parameters. In addition, any control settings should also be set (AUTOSTOP, MAX_ACCESS_SIZE, ACCESS_DELAY, GEMDONE_MASK, N202_LENGTH, CIPHER_FCS, etc.).

As the flow of Figure 54-3 shows, there are a number of places where the GEM module will do two operations at the same time to speed processing and allow it to minimize the number of data accesses done. At the start of the process, the GEM module will add the header information to the FCS and if the E bit is set, process the INPUT and Kc key information. Later if both the PM and E bits are set, the GEM will both add the data to the FCS and cipher the data at the same time. As called for in the GPRS specifications, the GEM will always use decrypted data for addition to the FCS. That is, if the GEM is encrypting data and generating an FCS (DIR = 0, E =1, PM = 1), it will add the data octet to the FCS before it is encrypted whereas if it is decrypting the data and verifying an FCS (DIR = 1, E =1, PM = 1), it will add the data to the FCS once the decryption of the data octet is complete.

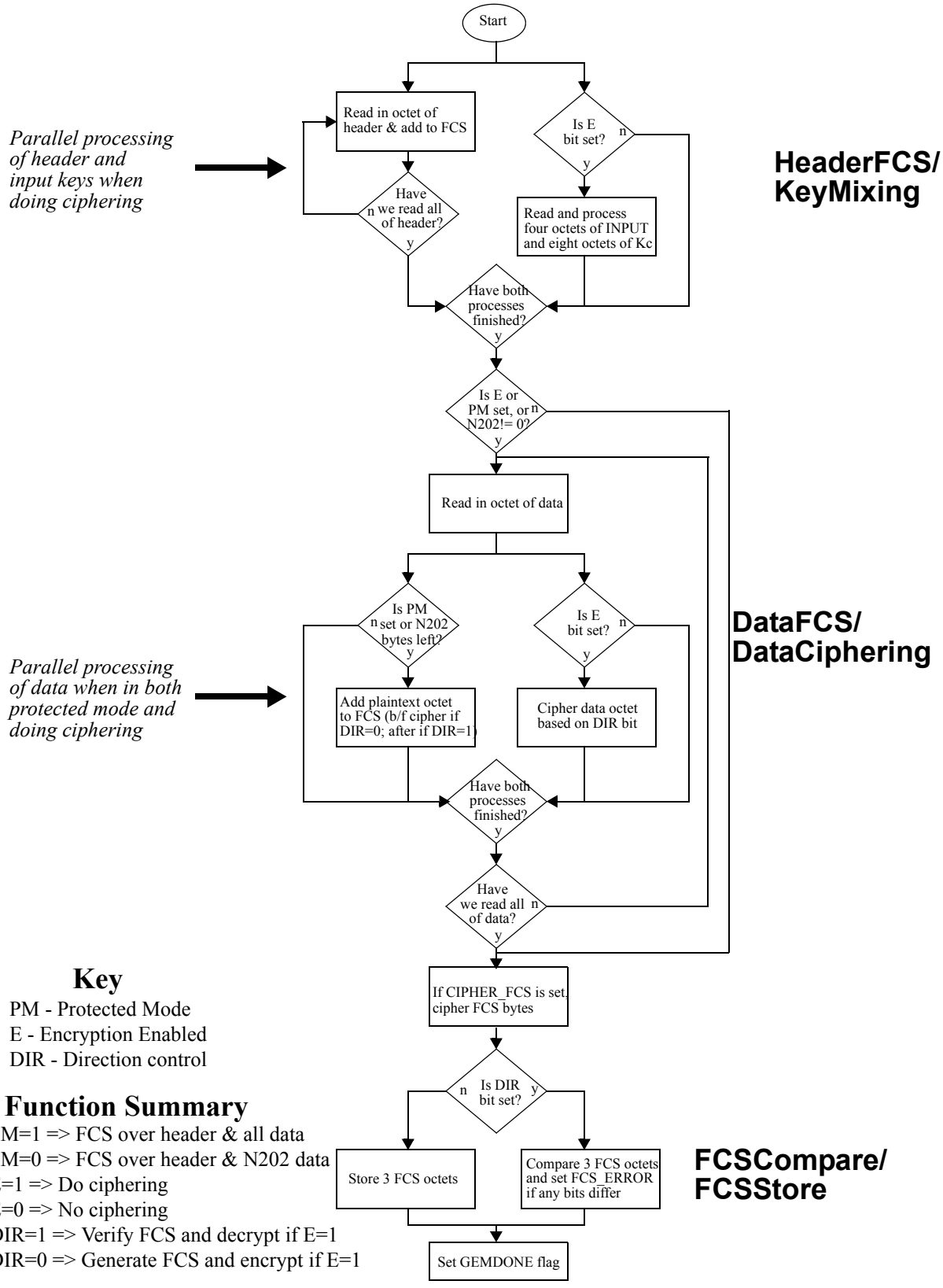


Figure 54-3. GPRS Execution Flow

54.2.1.4 Calculating GPRS Processing Time\Percentage Bus Cycles Stolen

The amount of time it takes the GEM to do the requested actions will depend on the frequency of the GEM clock, the type of processing requested, the amount of information to be processed, and any access delays incurred when reaching this data. Table 54-3 gives a summary of the amount of processing cycles it takes to do the various tasks the GEM is capable of doing.

Table 54-3. Processing Cycles Required by GEM to do Various Tasks

Task	When done	Cycles Required to Process
Add header to FCS	Always	1 cycle/bit * 8 bits/octet * # of octets in header (1-37) = 8 to 296 cycles
Mix key information	if E=1	289 cycles (reasons described in GEA specification)
Add data to FCS	if PM=1	1 cycle/bit * 8 bits/octet * # of octets in data (140-1520) = 1120 to 12160 cycles
Cipher data	if E=1	1 cycle/bit * 8 bits/octet * # of octets in data (140-1520) = 1120 to 12160 cycles
Store/check FCS	Always	2-3 cycles
Overhead	Always	5-8 cycles

The table is grouped in a way to show which actions will be done in parallel if possible. For instance, the inclusion of the header to the FCS and mixing of key information will be done in parallel, so the actual time for this task is the maximum of the two, not their sum. Likewise, the inclusion of data to the FCS and ciphering of the data each take the same amount of time, and will be done in parallel if both tasks are to be done.

The part of the equation that this table does not take into account is any processing delays that the GEM incurs in attempting to access data. These delays could be due to wait states applied by the memories themselves, arbitration delays caused by the ARM7, or halted processing requests set either by hardware or software via the STOP bit. The GEM has some amount of built-in pipelining of its data path that will help to prevent access delays from slowing the actual processing. The GEM can prefetch data up to 8 cycles in advance and continue processing with write data pending up to 8 cycles as well, so this flexibility can serve to isolate the GEM from small delays incurred at the data interface. If delays longer than this are incurred, the GEM will automatically stall its processing while waiting for the pending data access to complete.

Because the number of variables that affect whether access delays will affect GEM processing (wait states, access size limits, only reading data versus having to write data if USE_WRITE_PTR is used, etc.), it is difficult to describe a method to predict what these delays will be for any possible configuration. Instead, it is probably easiest for the user to simply run a number of test vectors using the configuration they are planning to use, and from that determining what the typical processing time is and what the number of stolen cycles were.

If the percentage of cycles stolen during this time is considered too high, the percentage can be reduced by increasing the value programmed into the alternate master arbitrator's (AMARB's) ACCESS_DELAY register which will effectively increase the cycles to do processing number. By increasing this number, the latency of the GEM processing is also increased and so these two conflicting goals must be weighed when setting the ACCESS_DELAY value. Note, this ACCESS_DELAY function used to be supplied by the GEM module itself, but has since been centralized in the AMARB. While this functionality still exists in the GEM design, the control in the AMARB should be used instead.

54.2.2 Using the GEM for Simple DMA Operations

Since the GEM has the ability to access any location in the MCU memory, and it has separate read and write pointers, it was a simple matter to provide a method for it to just copy data instead of processing it. To use this mode, just set the PROCESSING_MODE bits to DMA mode, set the USE_WRITE_PTR, initialize the READ_FRAME_START and WRITE_FRAME_START, and place the number of bytes to be copied in the FRAME_LENGTH register. Once these registers are ready, the user can set the START bit, and the GEM will set the GEMDONE flag when the copy is complete. Note, the data does not have to be in one continuous block; it can be broken into segments if desired (see Section 54.3.3.4, “Automatic Linked List Processing,” on page 54-14).

54.2.2.1 Calculating DMA Processing Time

The DMA design leverages the data processing flow utilized by the other GEM cores to access data to minimize its size impact and also allow it to perform the same types of functions (linked list processing, etc.). For these reasons, the DMA processing is not as fast as it could be. If access delays don’t affect operation, the DMA will move an average of 1 byte per every 5clock cycles. Since it will normally read and write 32 bits at a time, the normal bus utilization assuming 0 wait states accesses is 2 out of each 20 cycles, or 10%. A higher number of wait states will increase this as expected. That is, a 2 wait state memory would cause the GEM to take 6 out of each 20 cycles. The user can probably best determine the expected throughput and bus utilization by trying his exact configuration. If the bus utilization percentage is too high, software can force the GEM to wait longer between accesses with the AMARB’s ACCESS_DELAY register.

54.2.3 Aborting, Resetting, and Stopping GEM Processing

Once the GEM begins its processing, there are various methods to interrupt it: software abort, software reset, assertion of software stop bit by software.

If the frame currently being processed is no longer needed (or not the highest priority), its processing can be aborted by setting the ABORT bit in the CTL register. Setting this bit will immediately reset the GEM state machine, halt any pending GEM data access, and return the GEM back to its idle state. The GEMDONE flag will not be set. The register interface will not be affected--any previously programmed values will remain unchanged. It is recommended that the user assert the STOP bit prior to writing to the ABORT bit to ensure the GEM is not in the middle of a bus request when the abort occurs.

A similar effect can be achieved by setting the SWRST bit in the CTL register. The only difference between the ABORT bit and SWRST bit is that the registers also get reset back to their default states. Again, it is recommended that the user assert the STOP bit prior to writing to the SWRST bit to ensure the GEM is not in the middle of a bus request when the reset occurs.

If the software simply wants to delay the GEM’s processing of a given frame, it can set the STOP bit in the CTL register. The GEM will finish any currently pending access and then freeze its processing at this point. It will not drop down to its low power state. Once software clears the STOP bit, processing will resume from that point. If the STOP bit is written when the START bit is written, processing will begin once the STOP bit is cleared.

The GEM’s processing will automatically be aborted if a bus error occurs during a GEM access over the AHB. As an indication of this, the GEM will set the FRAME_ERROR and GEMDONE bits in the STAT register. If ciphering was being done to this frame (i.e. E = 1), the frame’s contents may be corrupted and should not be used.

54.2.4 GEM Clock Usage and Power Management

The GEM module runs at the same clock frequency that the ARM7 does. If the clock source to the ARM7 is disabled, the clock source to the GEM will be disabled as well. The GEM will automatically gate as many internal clocks as possible when it is not processing. There is no software bit that needs to be set to reduce the power consumption of the module when it is not in use. It is possible to read and write the GEM registers with the GEM in this low power state.

The GEM will continue processing in low power modes (WAIT/DOZE). Since STOP mode will gate the clock to the ARM7 and GEM, this will effectively stop the GEM's processing.

54.3 GEM Data Interface

The GEM has a flexible data interface. In order to give it full access to all RAM locations, the GEM actually steals cycles away from the ARM7 on its main bus, the AHB. To allow software to manage this interface, there are controls on the bus interface for things like how much data is fetched on each access, and the minimum number of clock cycles that must occur between each access. There is even a control to prevent the GEM from accessing any data at all to ensure that a given software routine won't be interrupted. Careful use of these controls should allow a system to find an acceptable trade-off between GEM processing latency and maximum percentage of ARM7 bus stalls.

It is assumed for all types of processing that the data exists before the GEM is started and will not be touched by any other master until the GEM is complete. Having another master (ARM7, DMAC, etc.) access the data that the GEM is also accessing could produce unpredictable results.

The GEM design on Neptune LTE supports both big endian and little endian memory configurations. The mode used is done based on the ARM's configuration, and should be transparent to the user.

54.3.1 Controlling the GEM Data Interface

In order for the GEM to have access to any required data, it has been made a secondary master on the ARM7 Local Bus (AHB) as shown in Figure 54-1. The ARM7 specification has more details concerning the arbitration interface between the GEM and the ARM7 (see references in Section 54.6.2, "Reference Documents."). Being a secondary master allows the GEM module to access the entire MCU memory map. It is assumed that the GEM will only access RAM space. The GEM can physically address all of the ARM7's memory map, but software shouldn't configure it to access any memory that would cause system problems.

WARNING:

The GEM should never be configured to access any memory in the burst FLASH area so as to prevent corruption of the ARM7's burst FLASH state. It is up to software to ensure this.

54.3.2 Parameters for GPRS LLC Frame Location in RAM

There are four (or five) parameters which dictate how the GPRS frames are stored in memory, and they are kept in the following registers: READ_FRAME_START register/WRITE_FRAME_START register if used, HEADER_LENGTH register, N202_LENGTH field in CTL register and FRAME_LENGTH register. The relationship between these parameters is illustrated in Figure 54-4.

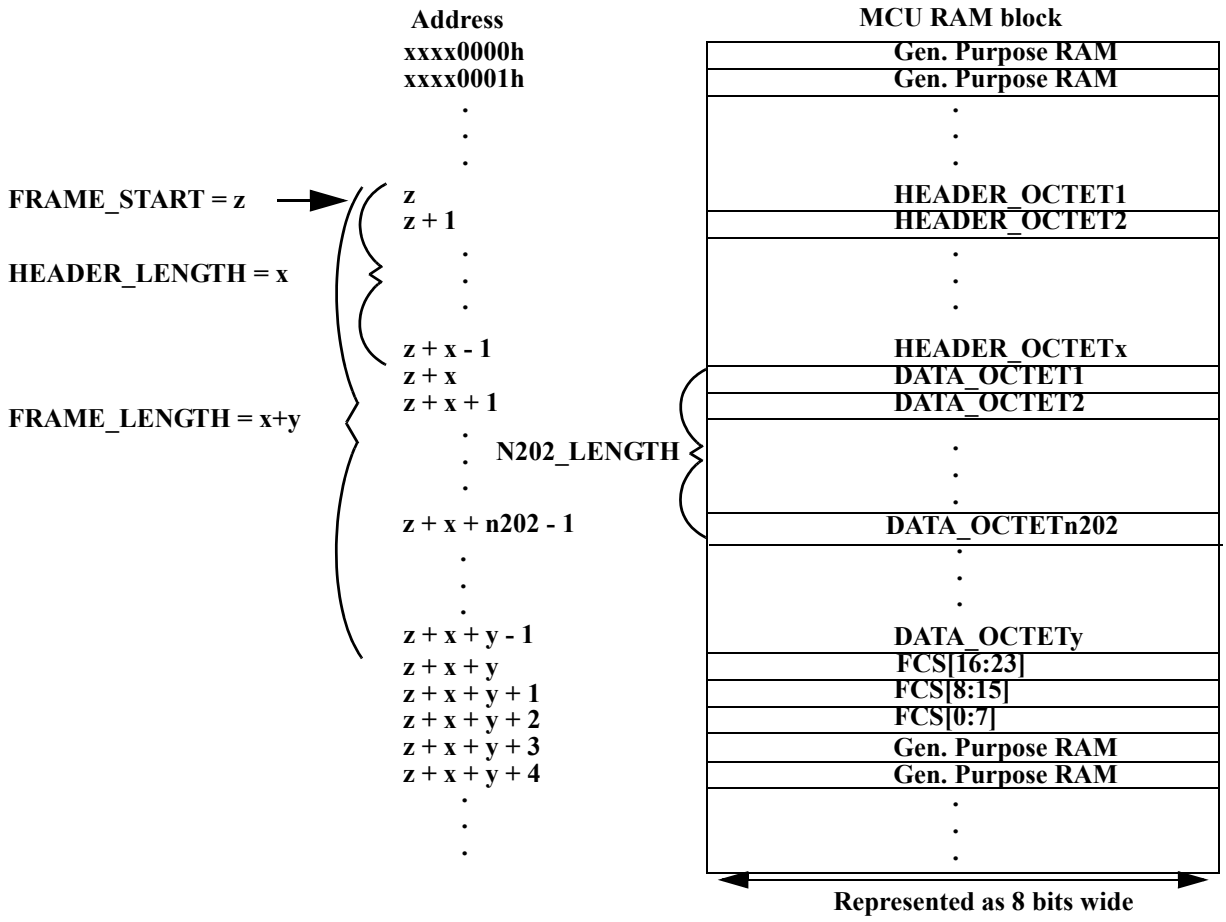


Figure 54-4. Method of Addressing a GPRS LLC Frame Stored in RAM

Figure 54-4 shows how the various pieces of information that make up the GPRS LLC frame (header section, data section--complete with N202 length segment, and FCS) are delimited by the FRAME_START, HEADER_LENGTH, N202_LENGTH, and FRAME_LENGTH parameters. The 32 bit FRAME_START parameter points to the first octet of the header. The READ_FRAME_START can be used to control read and writing, or if the USE_WRITE_PTR control bit is set, the WRITE_FRAME_START can be used to control where writes should occur (see Section 54.3.3.3, “Having Separate Read and Write Pointers,” on page 54-13 for more information). The 16 bit HEADER_LENGTH parameter tells the GEM how many octets are in the header section of the GPRS LLC frame (max of 64Kbytes). The four bit N202_LENGTH indicate how many bytes of the data section need to be added to the FCS when in unprotected mode (PM = 0). The 24 bit FRAME_LENGTH field tells the GEM how many total octets (header and data) are in the GPRS LLC frame. Note that it does not include the three octets of FCS information that will be appended to/read from the end of the frame.

While the diagram here seems to imply the data must be kept in a contiguous block, it is actually possible to break the data into segments, and process them that way (see Section 54.3.3.4, “Automatic Linked List Processing,” on page 54-14 for more information).

54.3.3 Miscellaneous Data Interface Controls

There are numerous controls offered by the GEM to allow the user to customize the nature of the data accesses that are done, regardless of the type of processing being done. The user can limit the size of data accesses done if the memory being accessed doesn't support either full 32 bit or 8 bit accesses (e.g. MDI); the user can put a minimum space between accesses to ensure the GEM doesn't steal too high a percentage of bus cycles; the user can setup writes to occur at a different location than the reads; and the user can break the data to be processed into segments or linked lists instead of having to have it all in one contiguous block. The following sections describes each of these in more detail.

54.3.3.1 Controlling the Size of Data Accesses

The GEM can access data in either 32 bit, 16 bit, or 8 bit chunks. The user can program both the maximum width used (either 32 or 16 bits) and the minimum width used (either 16 or 8 bits). There is a different maximum access size for read and writes since they may have different requirements. There is no need to limit the minimum size on reads since the GEM will never do a minimum sized read--it just does a maximum sized read all of the time and ignores any extra data.

A minimum access size of 16 bits on writes implies that if the GEM needs to write a byte to a given location, it must first read that 16 bit location and then only modify the byte that should be altered when writing those 16 bits back. These access sizes have to be configured by software prior to the start of a frame using the READ_MAX_ACCESS_SIZE, WRITE_MAX_ACCESS_SIZE and WRITE_MIN_ACCESS_SIZE bits in the CTL register, and cannot be switched during processing. When USE_WRITE_PTR is low, the READ_MAX_ACCESS_SIZE value is used for both reads and writes just as the READ_FRAME_START is. The WRITE_MIN_ACCESS_SIZE is used regardless of the value of USE_WRITE_PTR.

In the Neptune LTE system, an example when to use these access size controls is when doing accesses to MDI space. Since the MDI only supports 16 bit accesses, a user must limit reads to this space to be a maximum of 16 bits, and writes to be a maximum of 16 bits and a minimum of 16 bits.

The GEM will always try to access data using the maximum data width possible to minimize the total number of accesses required to do its processing. There are times however when it will not be able to do that maximum-sized access. Figure 54-5 can be used to see some examples of when this might be true. In all three cases, assume that the GEM is generating an FCS at the end of a frame where data was neither ciphered nor protected. Therefore, the FCS information must be written, but the data information is neither written nor read. In this case, both the data section before the FCS field and the general purpose RAM after the FCS field needs to be preserved by the GEM module. In case #1, the frame ends on a clean 32 bit boundary, but since the FCS is only the last three bytes of this value, the full 32 bit value cannot be written since it would corrupt the last data octet. Instead, the first FCS octet will be written as a single byte, and the last two will be written as a 16 bit value. In this case, if the MIN_WRITE_ACCESS_SIZE is configured for 16 bits, the GEM would automatically read the full 16 bit half-word, and then only modify the second byte before writing the 16 bit word back out. In case #2, the FCS doesn't end on either a 32 bit or 16 bit boundary, so it must be written as a half-word and a byte to prevent the RAM location following the frame from being corrupted. Again a 16 bit read and write will be done for this last byte if MIN_WRITE_ACCESS_SIZE is set to 16 bits. Finally, in case #3, the FCS itself straddles a 32 bit boundary and doesn't end on a clean boundary, so again two writes will be done--a halfword followed by a single byte (or 16 bit read modify write). In addition to these examples, there are other scenarios where the GEM will drop down to smaller than maximum accesses when writing the last few bytes of a frame, but in all cases it is just a few accesses out of the total for the frame.

CASE #1

A[1:0]	RAM Contents
00	last data octet
01	FCS[16:23]
10	FCS[8:15]
11	FCS[0:7]
00	RAM
01	RAM
10	RAM
11	RAM
00	RAM
...	...

CASE #2

A[1:0]	RAM Contents
00	data octet
01	data octet
10	data octet
11	last data octet
00	FCS[16:23]
01	FCS[8:15]
10	FCS[0:7]
11	RAM
00	RAM
...	...

CASE #3

A[1:0]	RAM Contents
00	data octet
01	last data octet
10	FCS[16:23]
11	FCS[8:15]
00	FCS[0:7]
01	RAM
10	RAM
11	RAM
00	RAM
...	...

Figure 54-5. Illustration of Time when Non-Maximum Sized Access will be Done

54.3.3.2 Controlling the Spacing of Data Accesses

When doing its processing, the GEM will be reading and writing to the selected memory as needed. Every clock cycle that the GEM is doing an access is a clock cycle that the ARM7 cannot. The percentage of bus cycles that the GEM will steal from the ARM7 depends on the type of processing being done, the width of the RAM being accessed, and the number of wait states associated with each access. It can vary from >50% down to as little as 3%. Software should always be able to predict what the percentage of usage will be for any given situation (see Section 54.2.1.4, “Calculating GPRS Processing Time\Percentage Bus Cycles Stolen.”). If this percentage is considered too high, software can limit how often the GEM can access RAM by appropriately setting the parameters in the alternate master arbiter (AMARB).

To stop the GEM from using the data interface completely, software can either set the STOP bit in the CTL register to temporarily halt the processing of a given frame, or the ABORT bit in the CTL register to permanently halt the processing of a given frame. More on these functions can be found in Section 54.2.3, “Aborting, Resetting, and Stopping GEM Processing.” .

54.3.3.3 Having Separate Read and Write Pointers

When processing a frame, the GEM can either write data to the same place it was read from, or use a separate write pointer, the WRITE_FRAME_START register, to place the data elsewhere. To tell the GEM to use the WRITE_FRAME_START for writes, set the USE_WRITE_PTR bit in the GEM’s CTL2 register. If the USE_WRITE_PTR is clear, then all writes occur to the same location the data was read from.

With GPRS frame processing, the GEM does not always write all of the bytes that it reads (e.g. header bytes), and in some cases, writes bytes it never read at all (e.g. when generating the FCS). The GEM will still only write the bytes that need to be written when USE_WRITE_PTR is cleared. However, if a separate read and write pointer are used (e.g. USE_WRITE_PTR is 1), the GEM will write all bytes of the frame so that the resulting frame at the destination location will contain all of the bytes that were in the original frame.

When processing frames using the linked list buffer descriptor, the same behavior as described here applies. The only difference is that the initial write address does not come from the WRITE_FRAME_START register, but from the buffer descriptor instead.

54.3.3.4 Automatic Linked List Processing

In order to allow more efficient use of MCU memory space, the GEM has the capability to optionally process data from a linked list rather than requiring the data to appear in one contiguous block. The amount of total processing is still determined by the appropriate length controls for each algorithm (HEADER_LENGTH/FRAME_LENGTH for GPRS; FRAME_LENGTH for DMA;). However, the GEM can optionally be configured where the data to operate on is pointed to by entries in a buffer descriptor, the location and format of which is described below. Using this feature, software is able to put the location and size of each segment of the frame to be processed into this buffer descriptor location, and when the GEM is started, it will process from one segment to the next until the proper length field is finally reached.

Processing of data via linked list can be enabled separately for reads or writes or both. The method of processing depends on whether linked lists are enabled (LINK_EN/WRITE_LINK_EN), and if a separate writes are enabled (USE_WRITE_PTR). Table defines the various configurations supported. Note, the WRITE_LINK_EN bit is only used if USE_WRITE_PTR is set.

Table 54-4. Potential data access configurations

USE WRITE PTR	LINK EN	WRITE LINK EN	Description of method GEM uses to access the data
0	0	x	All reads and writes done based on address in READ_FRAME_START register
0	1	x	All reads and writes done based on addresses contained in buffer descriptor pointed at by READ_FRAME_START address
1	0	0	Reads done based on addresses in READ_FRAME_START; Writes done based on addresses in WRITE_FRAME_START
1	0	1	Reads done based on addresses in READ_FRAME_START; Writes done based on addresses contained in buffer descriptor pointed at by WRITE_FRAME_START
1	1	0	All reads done based on addresses contained in buffer descriptor pointed at by READ_FRAME_START address Writes done based on addresses in WRITE_FRAME_START
1	1	1	All reads done based on addresses contained in buffer descriptor pointed at by READ_FRAME_START address Writes done based on addresses contained in buffer descriptor pointed at by WRITE_FRAME_START

What this table shows is that the GEM can access data from either contiguous memory or from a series of memory segments described by a buffer descriptor. When USE_WRITE_PTR is set, reads and writes are handled separately, and therefore can be enabled to do either configuration independent of the other using the LINK_EN and WRITE_LINK_EN bits. When USE_WRITE_PTR is cleared, the write accesses are done using whatever configuration the reads are set up for.

Figure 54-6 shows the format of the buffer descriptor used for either reads or writes. As described in Table , the start of the buffer descriptor for reads is always pointed at by READ_FRAME_START. If USE_WRITE_PTR and WRITE_LINK_EN are both high, the writes will have their own buffer descriptor which will be pointed at by the WRITE_FRAME_START register. If both are used, there will be a separate buffer descriptor for each.

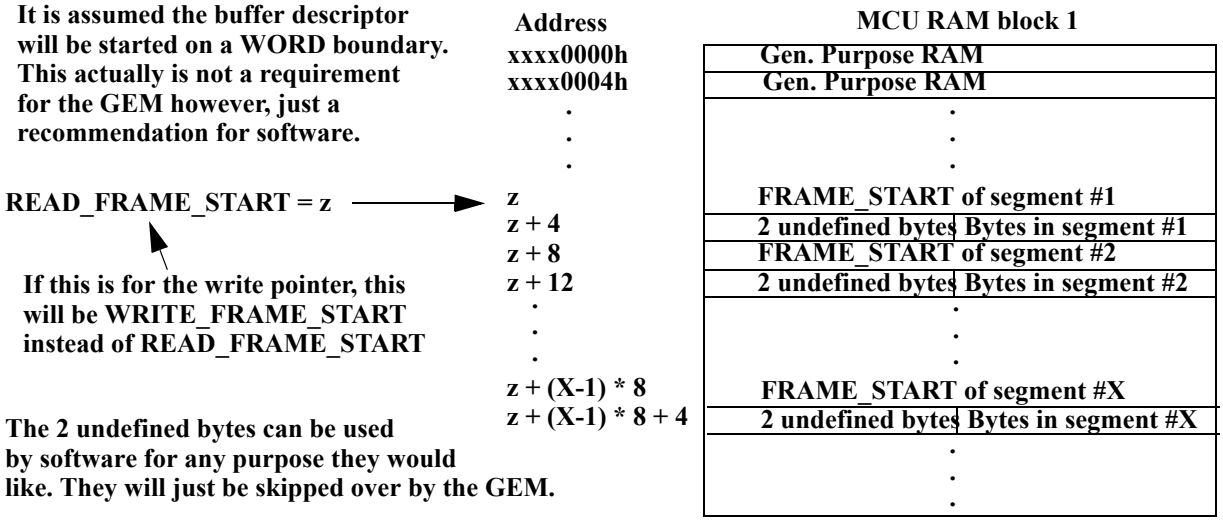


Figure 54-6. Buffer descriptor format

As depicted in Figure 54-6, the buffer descriptor format takes up 8 bytes per segment. Only 6 of these bytes are defined. 8 bytes are reserved to keep the 32 bit frame pointers on word boundaries. The byte length field only needs to be 16 bits in length, so this leaves two bytes per buffer entry for use by software for any purpose they would like, if any.

The maximum number of either read or write segments the GEM can process for a given frame is 1024. That is, for any given frame, there can be as many as 1024 read segments and if desired, as many as 1024 write segments, but no more than this for either. The maximum number of bytes in any given segment is 65536.

54.4 Summary of Changes between the Neptune and Neptune LTS Parts

The Neptune LTS GEM is a port from the Patriot Indy project, where the Neptue GEM was ported from Patriot Hip6w. There were numerous changes to the GEM module design between Neptune and the Neptune LTS part. This section is meant to document that list of changes, as well as the modifications that were needed to port Indy to Neptune LTS. Note, as long as the software does not write to any register that wasn't previously defined, nor set any control bit that was not defined before in the GEM's CTL register, all software that ran on the Patriot part should run exactly the same way on the Patriot Indy part.

54.4.1 Additional processing modes added

- Simple DMA mode (copy a number of bytes from source to destination)

54.4.2 Additional data accessing features added

- Support automatic linked list processing
- Support minimum 16 bit writes so GEM can write to MDI
- Support separate read/write pointers so output can go to different location than input

54.4.3 Modifications Made Going from Patriot to Neptune

When transitioning the GEM design from the Patriot IC to the Neptune IC, a number of changes were made. These are listed below.

- IGNORE_DEBUG bit (bit #14) in control register no longer functions. This was due to this functionality now being taken care of by Alternate Master Arbiter (AMARB) module. That is, the AMARB will prevent bus requests from occurring when in debug mode on a port by port basis. Note, this bit still physically exists in the GEM design, it just has no effect other than it can be set and cleared by software.
- SLOW_LPM bit (bit #13) in control register no longer functions. This was due to this functionality now being taken care of by AMARB module. That is, the AMARB has the functionality that allows for a certain number of cycles to exist between requested accesses. Note, this bit still physically exists in the GEM design, it just has no effect other than it can be set and cleared by software.
- AUTO_STOP bit (bit #8) in control register no longer functions. This was due to this functionality now being taken care of by Core Arbiter (CARB) module. That is, the CARB will prevent bus requests from stalling the MCU while an interrupt is being processed. Note, this bit still physically exists in the GEM design, it just has no effect other than it can be set and cleared by software.
- TC[2:1] bits (bits 6,5) in control register no longer function for AHB applications. The AHB interface has similar control bits, but only one bit is required for Neptune requirements and therefore the TC[0] bit has been renamed to HPROT1. Note, these bits still physically exist in the GEM design, they just have no effect other than they can be set and cleared by software.
- ACCESS_DELAY register should no longer be used. Note, the ACCESS_DELAY register (the 32 bit register immediately following the FRAME_LENGTH register), still exists, but should no longer be used since this functionality is now in the AMARB. If the bits in this unspecified register are set, they will delay the spacing between the start of GEM accesses by that value. Note, since the GEM no longer has an indication of low power mode, it will continue to operate at this delay even in low power mode. Again, these bits should not be set by the user; the corresponding function in the AMARB should be used instead.

54.5 Programmer's Model

This section will describe the normal method of using the module for the various types of processing supported. Note, only one type of processing can be programmed at a time. That is, one cannot change the registers for the next processing to be run while a current process is in progress.

54.5.1 Programming for GPRS frame processing

Once the portion of the frame to be processed is in place, the following steps can be taken to process it properly.

- Setup any controls over the nature of the GEM GPRS process (can be done in any order).
  - — Set PROCESSING_MODE for GPRS frame processing.
  - — Set parameters for nature of GEM accesses.
    - – HPROT1 for what value is used on HPROT1 of AHB
    - – MAX_WRITE_ACCESS_SIZE, MIN_WRITE_ACCESS_SIZE, MAX_READ_ACCESS_SIZE as needed for memory type
    - – GEMDONE_MASK as desired.

  - — Set CIPHER_FCS, GEA2, DIR, E, and PM to get required processing.
  - — Set up USE_WRITE_PTR, LINK_EN, WRITE_LINK_EN, READ_FRAME_START, and WRITE_FRAME_START to properly point to data to be processed and location for resultant storage
  - — Set up HEADER_LENGTH, N202_LENGTH and FRAME_LENGTH parameters to access proper memory.
  - — If doing ciphering (E=1), set INPUT and KC values.
- Set the START bit.
- Wait for GEM interrupt or poll the GEMDONE flag as appropriate.
- Check the FRAME_ERROR and FCS_ERROR bits once GEMDONE has been set.
- Clear any set status bits by writing a 1 to them.
- Use any generated information (FCS or ciphered data) as needed.

54.5.2 Programming for Simple DMA Mode Processing

Once the portion of the frame to be processed is in place, the following steps can be taken to process it properly.

- Setup any controls over the nature of the data copy process (can be done in any order).
  - — Set PROCESSING_MODE for DMA mode processing.
  - — Set parameters for nature of GEM accesses.
    - – HPROT1 for what value is used on HPROT1 of AHB
    - – MAX_WRITE_ACCESS_SIZE, MIN_WRITE_ACCESS_SIZE, MAX_READ_ACCESS_SIZE as needed for memory type
    - – GEMDONE_MASK as desired.
  - — Set up USE_WRITE_PTR, LINK_EN, WRITE_LINK_EN, READ_FRAME_START, and WRITE_FRAME_START to properly point to data to be processed and location for resultant storage
  - — Set up FRAME_LENGTH value to access proper total number of bytes.
- Set the START bit.
- Wait for GEM interrupt or poll the GEMDONE flag as appropriate.
- Check the FRAME_ERROR bit once GEMDONE has been set.
- Clear any set status bits by writing a 1 to them.
- Use the copied information as needed.

54.6 Register Map

The GEM's registers can be accessed only as 32 bit values. If any register in the GEM's address space that is outside of the memory map defined here is accessed, a bus exception will be generated. No bus exception is generated for writing to unimplemented bits.

Figure 0-14.

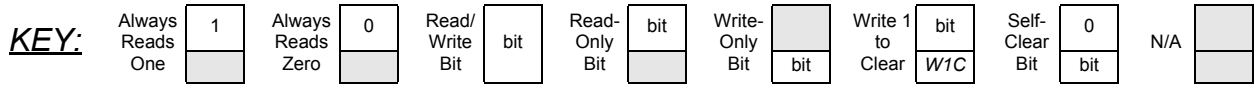


Table 54-5. GEM Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTL ($2485_9000)	R	0	0	0	0	0	0	0	0	0	0	0	CIPHER FCS	N202_LENGTH[3:0]			
	W																
	R		0	0	0	0	0		0	MAX ACCESS SIZE	0	0	HPROT	GEM DONE MASK	DIR	E	PM
	W	GEA2			START	SWRST	ABORT	STOP									
STAT ($2485_9004)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	GEM BUSY	FRAME ERROR	FCS ERROR	GEM DONE
	W														W1C	W1C	W1C
Kc ($2485_9008)	R	Kc[63:48]															
	W																
	R	Kc[47:32]															
	W																
Kc0 ($2485_900C)	R	Kc[31:16]															
	W																
	R	Kc[15:0]															
	W																
INPUT ($2485_9010)	R	INPUT[31:16]															
	W																
	R	INPUT[15:0]															
	W																
READ_FRAME START ($2485_9014)	R	READ_FRAME_START[31:16]															
	W																
	R	READ_FRAME_START[15:0]															
	W																
HEADER LENGTH ($2485_9018)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	HEADER_LENGTH [15:0]															
	W																
FRAME LENGTH ($2485_901C)	R	0	0	0	0	0	0	0	0	FRAME_LENGTH[**23**:16]							
	W																
	R	FRAME_LENGTH[15:0]															
	W																
ACCESS DELAY ($2485_9020)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	ACCESS_DELAY[6:0]						
	W																
CTL2 ($2485_9024)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table 54-5. GEM Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	R	0	0	0	0	0	0	0	0	PROCESSING MODE[1:0]		MAX WRITE ACCESS SIZE	MIN WRITE ACCESS SIZE	0	WRITE LINK EN	LINK EN	USE WRITE PTR
	W																
WRITE_FRAME START ($2485_9028)	R	WRITE_FRAME_START[31:16]															
	W																
	R	WRITE_FRAME_START[15:0]															
	W																

54.6.1 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various GEM registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

CTL GEM Control Register **Addr $2485_9000**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
												CIPHER FCS	N202_LENGTH[3:0]			
TYPE	r	r	r	r	r	r	r	r	r	r	r	rw	rw			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0			

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	GEA2			START	SWRST	ABORT	STOP		MAX ACCESS SIZE			HPROT	GEM DONE MASK	DIR	E	PM
TYPE	rw	rw	rw	slfclr	slfclr	slfclr	rwm	rw	rw		rw		rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	1		0		1	0	0	0

Table 54-6. CTL Description

Name	Description	Settings
CIPHER FCS Bit 20	**Cipher FCS**—This read/write bit controls whether the GEM will cipher the FCS or not when doing GPRS frame ciphering. CIPHER_FCS resets to a 0.	0 = GEM does not cipher FCS. 1 = GEM will cipher FCS when ciphering is done.
N202 LENGTH [3:0] Bits 19:16	**N202 Length—**These read/write bits indicate the number of data bytes at the start of the data portion of the GPRS frame that should be added to the FCS when PM = 0. See Figure 54-4 on page 54-11 for a diagram showing what this looks like. N202_LENGTH resets to a 0.	Number of bytes.
GEA2 Bit 15	**GEA2 Select—** This read/write bit controls whether the GEM uses the GEA-2 or GEA-1 algorithm when doing GPRS ciphering. GEA2 resets to a 0.	0 = GEM uses GEA-1 algorithm when doing ciphering. 1 = GEM uses GEA-2 algorithm when doing ciphering.
Bits 14-13	**Non functional bits— Non Functional Bit** — Can be set, but has no effect. Function available in Patriot has been disabled for Neptune (See section Section 54.4.3, “Modifications Made Going from Patriot to Neptune,”).	N/A
START Bit 12	**Start GEM Processing —**This self-clearing bit starts the processing of the GEM. The bit always reads as 0. The type of processing that will be done is controlled by the PROCESSING_MODE bits. If either the SWRST or ABORT bits are set on the same cycle the START bit is set, the software reset will take precedence and no processing will take place. Writing a one to the START bit has no function if the GEMDONE flag in the STAT register is set. **Note:** The GEMDONE flag must be cleared for the setting of the START bit to have any effect. **Note:** If the STOP bit is high when the START bit is set, the GEM will not start the processing until the STOP bit is cleared. Once the STOP bit is cleared, processing will immediately begin.	0 = Do not start the GEM processing 1 = Start the GEM processing

Table 54-6. CTL Description (Continued)

Name	Description	Settings
SWRST Bit 11	**Software Reset of GEM —** This self-clearing bit initiates a software reset of the entire GEM module. The bit always reads as 0. A software reset will reset all state machines back to their default states thereby halting any processing in progress, as well as all register values back to their reset states. It will clear all status bits and any pending interrupts. If both the SWRST and START bits are set in the same cycle, the software reset will take precedence and no processing will take place. The SWRST is automatically released internally after two clocks of the GEMCLK, so the GEMCLK must run at least two cycles before the user can write new values to any register bits. **Note:** It is recommended that if the GEM may be running when the software decides to reset it, that the user first write to the STOP bit to ensure the GEM is not in the middle of a bus request when the reset occurs. The write to the STOP bit should be done in a separate write than the setting of the SWRST bit.	0 = Do not perform a software reset of the GEM 1 = Perform a software reset of the module
ABORT Bit 10	**Abort GEM Processing —** This self-clearing bit initiates a software reset of the module. The bit always reads as 0. A software abort will reset all state machines back to their default states thereby halting any processing in progress, but it will not change the state of any of the control register bits. It will clear all status bits and any pending interrupts. If both the ABORT and START bits are set in the same cycle, the process abort will take precedence and no processing will take place. The ABORT is automatically released internally after two clocks of the GEMCLK. **Note:** It is recommended that if the GEM may be running when the software decides to abort it, that the user first write to the STOP bit to ensure the GEM is not in the middle of a bus request when the abort occurs. The write to the STOP bit should be done in a separate write than the setting of the ABORT bit.	0 = Do not perform a process abort of the module 1 = Perform a process abort of the module leaving register bits unmodified
STOP Bit 9	**Stop GEM Processing —** This read/write bit controls whether the GEM can continue an active process. Once the STOP bit is set, the GEM will halt its processing as soon as any pending data transfers are complete. The STOP bit can be set automatically by hardware any time an interrupt is issued if the AUTOSTOP bit is set. Software must clear the STOP bit manually to resume processing. STOP must be cleared for a START to take effect. If a START is issued with STOP set, and the STOP is later taken away, the processing will immediately begin. The ABORT bit can be used to prevent the processing from proceeding if desired. This bit is cleared by a SWRST but not by an ABORT. STOP resets to a 0.	0 = GEM allowed to process data 1 = GEM will stop processing following the completion of any pending access
Bit 8	**Non Functional Bit** — Can be set, but has no effect. Function available in Patriot has been disabled for Neptune (See section Section 54.4.3, “Modifications Made Going from Patriot to Neptune,”).	N/A

Table 54-6. CTL Description (Continued)

Name	Description	Settings
MAX READ ACCESS SIZE Bit 7	**Maximum Read Access Size —** This read/write bit controls the maximum size of GEM RAM accesses. If USE_WRITE_PTR is a 0, it controls the maximum size of reads and writes; if USE_WRITE_PTR is 1, it only controls reads. See Section 54.3.3.1, “Controlling the Size of Data Accesses,” on page 54-12 for more information. MAX_READ_ACCESS_SIZE resets to a 1.	0 = Maximum size of GEM reads (& writes if USE_WRITE_PTR = 0) will be 16 bits wide 1 = Maximum size of GEM reads (& writes if USE_WRITE_PTR = 0) will be 32 bits wide
Bits 6-5	**Non Functional Bits—**.Can be set, but has no effect. Function available in Patriot has been disabled for Neptune (See section 54.2.7 Modifications Made Going from Patriot to Neptune).	N/A
HPROT1 Bit 4	**AHB Protection Control—** Controls what value is driven onto the internal AHB signal for HPROT[1] each time a GEM access is done. The HPROT1 bit is reset to 0.	HPROT1: 1 => Privileged mode 0 => User mode
GEM DONE MASK Bit 3	**GEM Done Mask—** Controls whether the GEMDONE flag is masked from creating the GEM interrupt. If the GEMDONE bit is set, and the GEMDONE_MASK is cleared, the GEM interrupt will be asserted. The GEMDONE_MASK resets to 1.	0 = GEM DONE flag can cause GEM interrupt 1 = GEM DONE flag cannot cause GEM interrupt

Table 54-6. CTL Description (Continued)

Name	Description	Settings
DIR Bit 2	**Direction Control —** For GPRS processing, controls whether the ciphering function being undertaken (if E=1) is going to be encryption or decryption. Also controls whether the FCS function being undertaken is going to be generation or verification. The direction bit should be cleared for uplink and set for downlink. This bit has the same function and polarity as the DIR bit described in the GSM 4.64 specification when one considers this from a MS point of view. For 3GPP processing, he value of the DIRECTION is 0 for messages from UE (User Equipment--cell phone) to RNC (network) and 1 for messages from RNC to UE.	0 = For GPRS, if a ciphering function is going to take place (E=1), it will be encryption. FCS will be generated. For 3GPP, message is from UE to RNC. 1 = For GPRS, if a ciphering function is going to take place (E=1), it will be decryption. FCS will be verified. For 3GPP, message is from RNC to UE.
E Bit 1	**Encryption Enable —** For GPRS processing, E enables the GPRS encryption/decryption on a data block. When the start bit is set, either both an encryption/decryption and an FCS will be generated/verified if E=1, or just an FCS will be generated/verified if E=0. This bit has the same function and polarity as the E bit described in the GSM 4.64 specification.	0 = Disable encryption / decryption of the data block. 1 = Enable encryption / decryption of the data block.
PM Bit 0	**Protected Mode —** For GPRS processing, PM controls whether the FCS generated protects both the header and the data information or just the header and first N202 bytes of the data. This bit has the same function and polarity as the PM bit described in the GSM 4.64 specification.	0 = Include only the header and first N202 bytes of data information in the generation / verification of the FCS. 1 = Include the header and all data information in the generation / verification of the FCS

WARNING::

The bits PM, E, and DIR should not be written while the GEM is processing.

Table 54-7. Summary of the GEM GPRS Function Controls

DIR	E	PM	Description of GEM function to be undertaken when START bit is set
0	0	0	FCS will be generated over the header & N202 bytes of data. No encryption of data is done.
0	0	1	FCS will be generated over the header and all data. No encryption of the data is done.
0	1	0	FCS will be generated over the header & N202 bytes of data. Encryption of data will be done.
0	1	1	FCS will be generated over the header and all data. Encryption of the data will be done.
1	0	0	FCS will be verified over the header & N202 bytes of data. No decryption of data is done.
1	0	1	FCS will be verified over the header and all data. No decryption of the data is done.
1	1	0	FCS will be verified over the header & N202 bytes of data. Decryption of data will be done.
1	1	1	FCS will be verified over the header and all data. Decryption of the data will also be done.

STAT GEM Status Register **Addr $2485_9004**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
													GEM BUSY	FRAME ERROR	FCS ERROR	GEM DONE
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	w1c	w1c	w1c
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 54-8. STAT Description

Name	Description	Settings
Bits 31-4	**Unimplemented Bits—** Always reads as zero; writing has no effect.	N/A
GEM BUSY Bit 3	**GEM is busy—**When this bit is set, it indicates that the GEM is currently processing. It does not create an interrupt. Since the start bit is self-clearing, this bit is the only way for software to know that a GPRS frame is currently being processed or not. This is a read only bit that reflects the state of the GEM. This bit will remain high when a process has been frozen due to the setting of the STOP bit.This bit is cleared by a software reset (SWRST) or process abort (ABORT).	0 = The GEM is not currently processing. 1 = The GEM is currently processing (but may be stopped).
FRAME ERROR Bit 2	**Error occurred when processing frame —** When this write 1 to clear bit is set, it indicates that the FRAME processing was aborted due to a bus error during a GEM access over the MLB. If a bus error occurs, the GEM will set the FRAME_ERROR bit and then end the processing by setting the GEMDONE bit and resetting the GEM processing engines. This bit is cleared by a software reset (SWRST) or process abort (ABORT). If ciphering was being done, data may be corrupted.	0 = The frame processing finished without any MLB access errors. 1 = The frame processing was halted abnormally due to a MLB access error.
FCS ERROR Bit 1	**FCS verification error —** When this write 1 to clear bit is set, it indicates that the FCS check done on the frame failed. This bit can only be set following an FCS verification process (DIR = 1). It has no meaning and therefore will never be set following an FCS generation process (DIR = 0).It does not create an interrupt. It is cleared by writing a 1 to this bit. This bit is cleared by a software reset (SWRST) or process abort (ABORT).	0 = The GPRS FCS verification process passed for this frame. 1 = The GPRS FCS verification process failed for this frame.
GEM DONE Bit 0	**GEM done processing —** This write 1 to clear bit is set when the selected processing initiated by the START bit has completed. It will create an interrupt if the GEMDONE_MASK bit is clear. It is cleared by writing a 1 to this bit. This bit is cleared by a software reset (SWRST) or process abort (ABORT). **Note:** The GEMDONE bit must be cleared for the setting of the START bit to have any effect.	0 = GEM interrupt is not pending. 1 = GEM has finished most recent selected processing. Interrupt is pending

Kc1 — GEM GPRS Kc1 Register — Addr $2485_9008

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	KC[63:48]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	KC[47:32]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Kc0 — GEM GPRS Kc0 — Addr $2485_900C

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	KC[31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	KC[15:0]]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 54-7.

Table 54-9. Kc Description

Name	Description	Settings
KC [63:31] Kc1 Bits 31-0 Kc0 Bits 31-0	**Secret Key Constant—** This set of registers hold the upper and lower 32 bits of the 64 bit secret key constant (Kc) described in the GEA specification (see reference in Section 37.5).	N/A

WARNING:

This register must not be written to while the GEM is processing

INPUT — GEM GPRS Input — Addr $2485_9010

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	INPUT[31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	INPUT[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 54-10. INPUT Description

Name	Description	Settings
INPUT[31:0] Bits 31-0	**Input key —** This set of registers hold the 32 bit input key (a.k.a message key) described in the GEA specification (see reference in Section 37.5)	N/A

WARNING:

This register must not be written to while the GEM is processing

READ_FRAME_START GEM Read Frame Start Register **Addr $2485_9014**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	READ_FRAME_START[31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	READ_FRAME_START[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 54-11. READ_FRAME_START Description

Name	Description	Settings
READ_FRAME START [31:0] Bits 31-0	**Starting address of frame to process** — If LINK_EN is 0, this register contains a pointer in the M*CORE address space to the start of the data frame to be processed by the GEM module. If LINK_EN is 1, this register contains a pointer to the buffer descriptor used to determine the address of the various segments to be processed by the GEM. If USE_WRITE_PTR is a 0, this register controls the address for both reads and writes. If USE_WRITE_PTR is a 1, it only controls the start address for reads. See Section 54.3.3.4, “Automatic Linked List Processing,” on page 54-14for more information.	N/A

WARNING:

This register must not be written to while the GEM is processing

HEADER_LENGTH BEARER

GEM GPRS Header Length Register

Addr $2485_9018

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	HEADER_LENGTH[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 54-12. HEADER_LENGTH Description

Name	Description	Settings
Bits 31-16	**Unimplemented Bits—** Always reads as zero; writing has no effect.	N/A
HEADER_LENGTH [15:0] Bits 15-0	**Length of header of GPRS data frame to be processed —** This register contains the length (in bytes/octets) of the header of frame to be processed by the GEM module as depicted in Figure 54-4 on page 54-11.	N/A

FRAME_LENGTH

GEM GPRS Frame Length Register

Addr $2485_901C

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
									FRAME_LENGTH[**23**:16]							
TYPE	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	FRAME_LENGTH[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 54-13. FRAME_LENGTH Description

Name	Description	Settings
Bits 31-24	**Unimplemented Bits—** Always reads as zero; writing has no effect.	N/A
FRAME_LENGTH [23:0] Bits 23-0	**Length of GPRS data frame to be processed —** This register contains the length in bytes (octets) of the frame to be processed by the GEM. The value in this register is the number of bytes of header and data octets (not including 3 bytes of FCS at end of frame) as depicted in Figure 54-4 on page 54-11. This is also the register that holds the byte count for the number of bytes to be copied when the GEM is configured for simple DMA operation. **Note:** The maximum FRAME_LENGTH value is 24 bits in length. The GEM cannot support a full 26 bit value. The 26 bits is for the BIT_LENGTH field only.	N/A

ACCESS_DELAY GEM Access Delay Register **Addr $2485_9020**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
										ACCESS_DELAY[6:0]						
TYPE	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 54-14. ACCESS_DELAY Description

Name	Description	Settings
Bits 31-7	**Unimplemented Bits—** Always reads as zero; writing has no effect.	N/A
ACCESS DELAY [6:0] Bits 6-0	**GEM Access Delay Count —** This register contains the count value for the minimum number of clock cycles that must exist between the beginning of GEM accesses to memory. This can be used to limit the peak bus usage that the GEM can achieve. It allows the software to trade-off this peak bus usage with the latency of the GEM processing itself. This value can be optionally ignored by the GEM any time the ARM7 is in a low power mode (WAIT/DOZE) based on the state of the SLOW_LPM bit in the CTL register.	N/A

NOTE:

Increasing the size of the ACCESS_DELAY parameter can increase the time it takes the GEM to process a frame.

NOTE:

This value can be changed during GEM processing.

WARNING:

While this register still exists in the GEM design, the intention is for this functionality to be control centrally for all alternate masters in the alternate master arbitrator (AMARB). Therefore, this register should never be change by software from its reset state.

CTL2 GEM Control Register 2 — Addr $2485_9024

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									PROCESSING MODE[1:0]		MAX WRITE ACCESS SIZE	MIN WRITE ACCESS SIZE		WRITE LINK EN	LINK EN	USE WRITE PTR
TYPE	r	r	r	r	r	r	r	r	rw		rw	rw	r	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Table 54-15. CTL2 Description

Name	Description	Settings
PROCESSING MODE [1:0] Bits 7-6	**Processing Mode—**This read/write bit controls what kind of processing the GEM will be doing when started--either GPRS frame processing or simple DMA mode. When configured for GPRS frame processing, the bits CIPHER_FCS, GEA2, DIR, E, & PM are used to control the type of GPRS processing that is done. PROCESSING_MODE resets to a 0.	00= GPRS frame processing. 01= Simple DMA mode.
MAX WRITE ACCESS SIZE Bit 5	**Maximum write access size —**This read/write bit controls what the maximum size of the GEM's write accesses should be when USE_WRITE_PTR is 1. Note, when USE_WRITE_PTR is a 0, the maximum write access size is controlled with the READ_MAX_ACCESS_SIZE bit since reads and writes occur to the same address space. MAX_WRITE_ACCESS_SIZE resets to a 1. **Note:** When USE_WRITE_PTR=0, this bit has no effect.	0 = When USE_WRITE_PTR=1, maximum size of writes is 16 bits. 1 = When USE_WRITE_PTR = 1, maximum size of writes is 32 bits.
MIN WRITE ACCESS SIZE Bit 4	**Minimum write access size—**This read/write bit controls what the minimum size of the GEM's write accesses should be. Unlike MAX_WRITE_ACCESS_SIZE, the GEM always uses this value to determine the minimum size of writes regardless of the value of USE_WRITE_PTR since there is never a minimum limit on read accesses. When GEM has 1 byte to write, but minimum size is 16, it will automatically read the 16 bits first, and modify only the byte to be written. MIN_WRITE_ACCESS_SIZE resets to a 0.	0 = Minimum size of writes is 8 bits. 1 = Minimum size of writes is 16 bits. GEM will read 16 bits and modify byte when a byte needs to be written.
WRITE LINK EN Bit 2	**Write Link Enable—** If USE_WRITE_PTR is set, this read/write bit controls if the GEM writes data via a linked list or a contiguous block. See Section 54.3.3.4, "Automatic Linked List Processing," on page 54-14for more information. WRITE_LINK_EN resets to 0.	0 = Write data as a contiguous block. 1 = If USE_WRITE_PTR is set, write data as a linked list.
LINK EN Bit 1	**Link Enable—** This read/write bit controls if the GEM processing frames via a linked list or a contiguous block. See Section 54.3.3.4, "Automatic Linked List Processing," on page 54-14for more information. LINK_EN resets to 0.	0 = Process frame as a contiguous block. 1 = Process frame as a linked list.

Table 54-15. CTL2 Description (Continued)

Name	Description	Settings
USE WRITE PTR Bit 0	**Use Write Pointer—**This read/write bit controls whether the GEM will use the same data pointers for writes as reads, or if it will use a different data pointer for writes than what was used for reads (See Section 54.3.3.4, “Automatic Linked List Processing,” on page 54-14 for more information). USE_WRITE_PTR resets to a 0.	0 = Writes of bytes should go to same address as they were read from. 1 = Writes should be done to different address than where they were read from.

WRITE_FRAME START — GEM Write Frame Start Register — **Addr $2485_9028**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	WRITE_FRAME_START[31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	WRITE_FRAME_START[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 54-16. WRITE_FRAME_START Description

Name	Description	Settings
WRITE_FRAME START [31:0] Bits 31-0	**Starting address of where to write output data** — If USE_WRITE_PTR is a 0, this register is not used; instead the configuration of the GEM used for reading is used for both reading and writing. If USE_WRITE_PTR is a 1, this register controls where data is written--if WRITE_LINK_EN is a 0, the register contains a pointer in the ARM7 address space to where the GEM should start the writing of output data when the frame is being processed; if WRITE_LINK_EN is a 1, the register contains the start address of the buffer descriptor which contains the segment information used to determine where writes should occur. See Section 54.3.3.4, “Automatic Linked List Processing,” on page 54-14for more information.	N/A

54.6.2 Reference Documents

Table 54-17.

Document	Source	Acronym	Version
IP Bus specification	SPS	IPBus	?
Advanced Microcontroller Bus Architecture Specification	ARM	AMBA	2.0
Neptune LTE IC Summary Specification	SPS	Neptune LTE	?
Clock Control Module Specification	SPS	CCM	?
GPRS LLC Layer Specification (GSM 04.64, TS 101 351)	ETSI	GSM 4.64	8.0
Specification of the GPRS Encryption Algorithm	ETSI	GEA	N/A

Chapter 55
Enhanced General Purpose Timers (EGPT)

Revision History Table

Revision	Date	Author	Changes
0.0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	03/20/02	Saurabh Chutani	Interface changes from MIG/PIG to IPBUS.
0.2	09/16/02	Gaurav Davra	Update for DSPhl15045 Added status bit SYNC in EGPTSR
0.3	11/13/02	Mark Babcock	Updated per DDTS# DSPhl15682.
0.4	05/07/03		Updated for LTE specification release.

Note: While making changes from MIG/PIG to IPBUS, MCU_CLK has been renamed as ipg_clk in Port List (Section 55.6) and Block Diagram (Section 55.7). . However, references to MCU_CLK in the text have not been changed for better understanding.

55.1 OVERVIEW

This section describes the MCU General Purpose Timers, the Pulse Width Modulator Timer and the Enhanced Periodic Interrupt Timer, as well as their register sets. All of them comprise one module called Enhanced General Purpose Timers (EGPT). It is composed of two timer prescalers, two 16-bit free-running counters for GPT & PWM and a 24-bit free running counter for the EPIT. It also includes three 16-bit output-compare blocks, 24-bit output compare and two input-capture blocks.

The clock sources for this block are MCU-PER-CLK and LOW-FREQ-REF (32.768KHz). These sources are the inputs for two programmable prescalers & one fixed prescaler. These prescalers's outputs use as inputs for the timers.

The General Purpose Timers Counter (TCNT) is a 16-bit free-running up counter which starts counting after it is enabled by software (TE=1). TCNT's clock source is the prescaler's output labeled "Timers Prescaler Clock Output". If the timer is disabled (TE=0), the counter freezes at its current value. On re-enabling the timers, TCNT resume counting up from its last value. TCNT can be read at any time by the processor

The PWM Counter (PWCNT) is a 16-bit free-running loadable down counter which starts counting after the PWM is enabled by software (PWE=1). PWCNT's clock source is the prescaler's output labeled "PWM Prescaler Clock Output". If the PWM is disabled (PWE=0), the counter freezes at its current value. If the PWM is re-enabled, the PWCR value is reloaded back to the counter which starts down-counting. The PWCNT can be read at any time by the processor.

The Enhanced Periodic Interrupt Timer (EPIT) is a 24-bit “set and forget” timer which starts counting after the EPIT is enabled by software (ITE=1). It is capable of providing precise interrupts at regular intervals with minimal processor intervention. EPIT’s clock is the fixed prescaler’s output which can be derived from either MCU-PER-CLK divided by 512, or by LOW-FREQ-REF under software control using the EPIT Clock Mode (ECM) bit. The timer can either count down from the value written in the modulus register, or it can be a free-running counter. Interrupt to the MCU will be generated when the 24-bit’s counter reaches the value of the compare register.

For the GPT,PWM and EPIT to work correctly in any of the low power modes (STOP, DOZE, WAIT), the required clock must be available to these submodules in that mode.

The EGPT register interface to the ARM High Performance Bus (AHB) is controlled by the Peripheral Interface Gasket (PIG) in conjunction with the EGPT’s Module Interface Gasket (MIG). Refer to the PIG/MIG chapter for a description of its operation and related signals.

The EGPT scan interface will be in accordance to the “DSP-MCU Peripherals Test Architecture.” Please refer to that chapter for a description of the scan signals’ operation.

55.2 EGPT ARCHITECTURE

The block diagram of the prescaler is given in Figure 55-1. The block diagrams of the GPT, PWM and EPIT are given in Figure 55-2, Figure 55-3 and Figure 55-4 respectively.

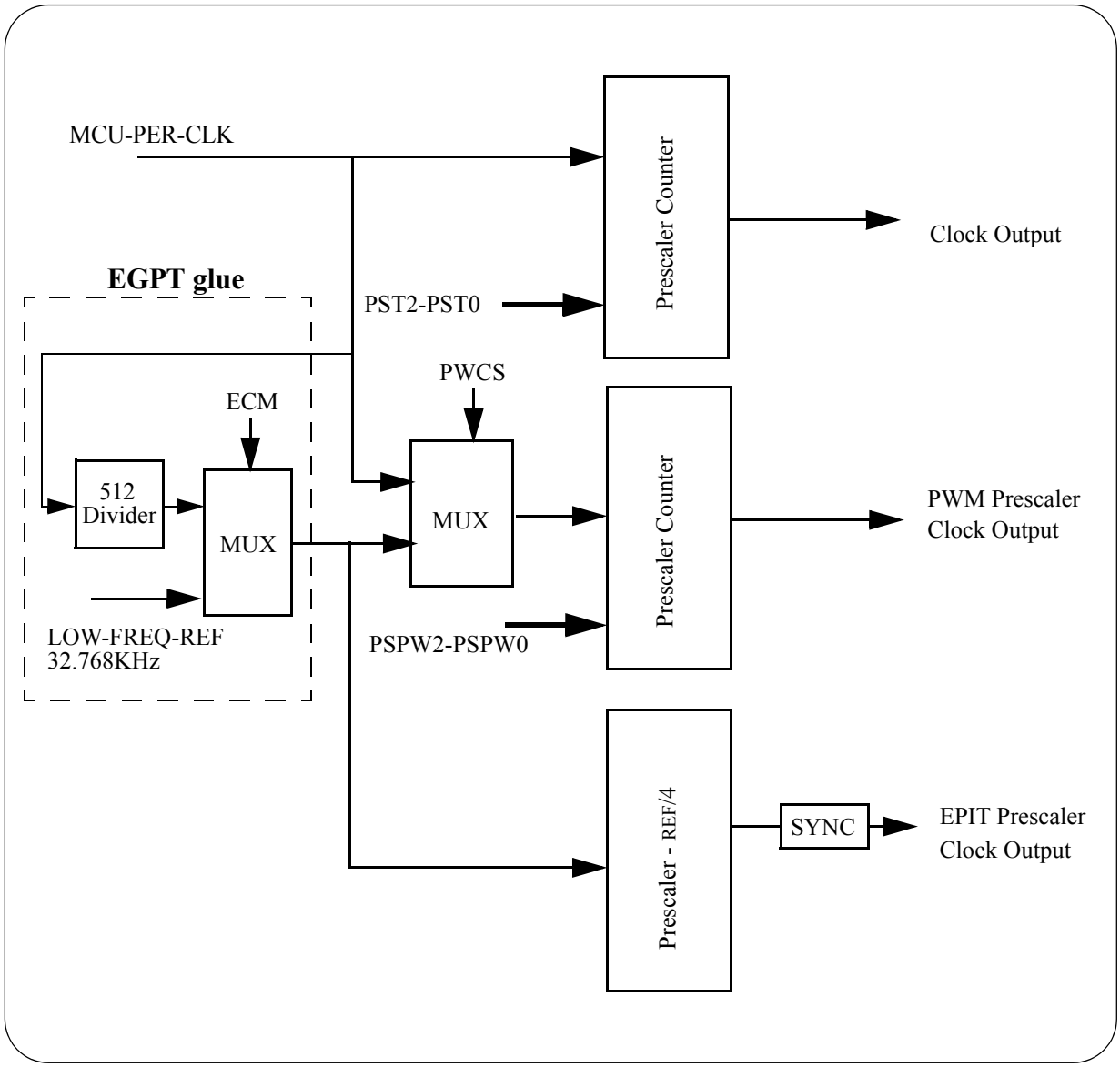


Figure 55-1. EGPT Clock Scheme

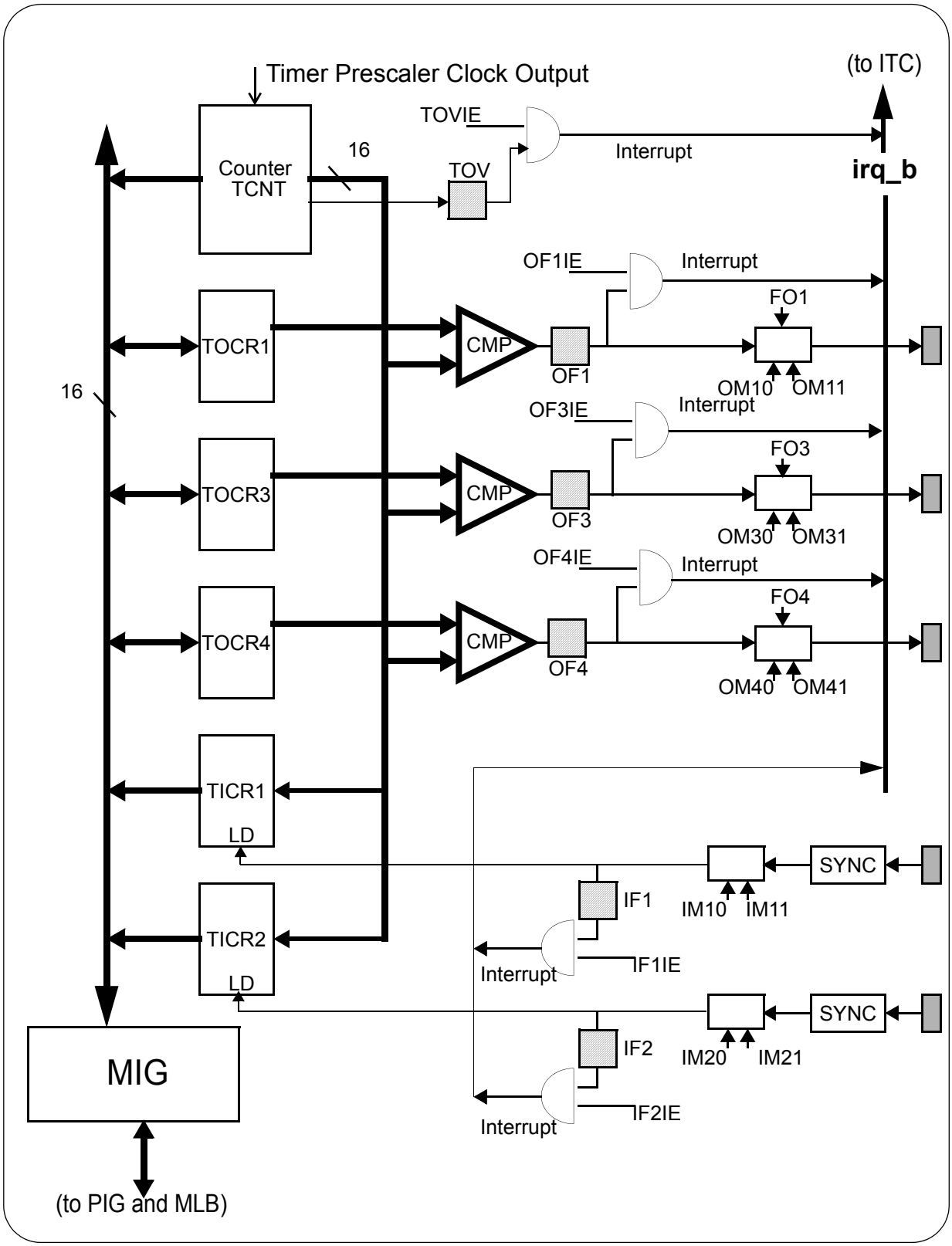


Figure 55-2. Timers Block Diagram

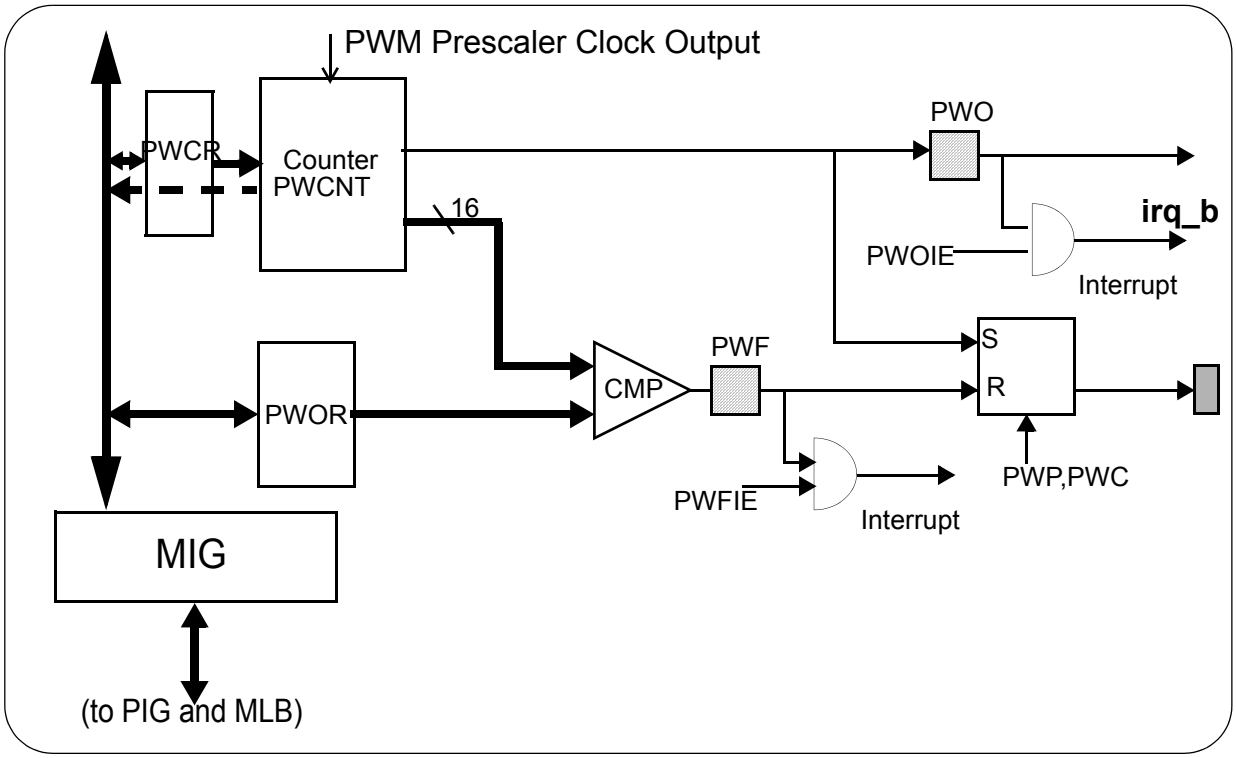


Figure 55-3. PWM Block Diagram

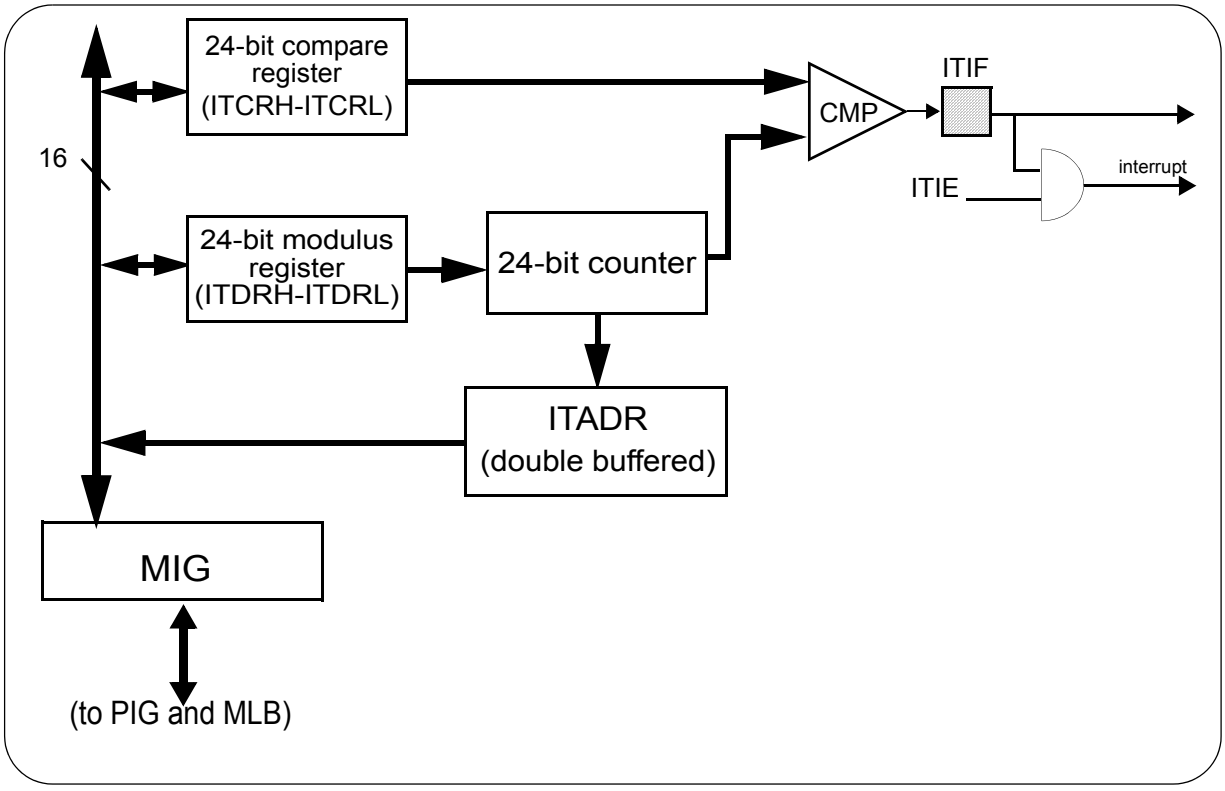


Figure 55-4. EPIT Block Diagram

55.3 GPT AND PWM OPERATION

The Capture/Compare and PWM units have independent free-running counters as a main timing component. The GPT timer derives its clock from a division of the MCU-CLK and the PWM can derive its clock from a division of the MCU-CLK or the LOW-FREQ-REF according to bits PWCS in EGPTCR and ECM in EGPTIR. Figure 55-1 is the GPT/PWM clock scheme.

These input clocks are divided by 8-stage divider chains. Multiplexers can select one of eight prescalers taps. Multiplexer outputs for the GPT Counter (TCNT) is selected by the bits PST2-PST0. Multiplexer outputs for the PWM Counter (PWCNT) is selected by the bits PSPW2-PSPW0.

After reset the MCU_CLK prescale rate is 2 for both GPT and PWM Counters (TCNT and PWCNT), i.e. the “GPT Prescaler Clock Output” and the “PWM Prescaler Clock Output” will be both MCU_CLK/2 (see Figure 55-1). The software can change the prescale rates for the GPT counter (TCNT) or the PWM counter (PWCNT) at any time (by changing the bits PST2-0 or PSPW2-0 respectively) but this change will take effect max. after “256 cycles” of the prescaler counter clock. The software can also change PWCS bit but this must be performed when the PWM is disabled (PWE=0).

The GPT Counter (TCNT) is a 16-bit free-running counter which starts counting after the GPT is enabled by software (TE=1). TCNT clock is the output of the prescaler labeled “GPT Prescaler Clock Output”. If the GPT is disabled (TE=0), the counter freezes at its current value and, if they are re-enabled, it resumes counting from this value. The TCNT can be read at any time by the processor.

Both Input Capture channels use the same 16-bit counter (TCNT). Each input capture pin has a dedicated 16-bit latch and input edge-detection/selection logic. Each input capture function has an associated status flag, and can cause the MCU to make an interrupt service request. When a selected edge transition occurs on an input capture pin, the associated 16-bit latch captures the content of TCNT and sets the appropriate interrupt status flag. An interrupt request can be generated when the transition is detected. The capture can be programmed to occur on the input pin rising edge, falling edge or on both edges or can be disabled. Since input capture events are asynchronous to the GPT counter, they are conditioned by a synchronizer and a digital filter. The events are synchronized with the MCU_CLK so that latching of the TCNT value and the TCNT increment occur on opposite half-cycles of the MCU_CLK. Any input transition shorter that one MCU_CLK period has no effect. Capture of any transition longer than two MCU_CLK periods is guaranteed. There can be up to one MCU_CLK cycle of uncertainty in latching of the input transition. The input capture registers can be read at any time without affecting their values.

The three Output Compare channels use the same counter as the Input Capture channels (TCNT). When the programmed content of an output compare register matches the value in TCNT, an output compare status flag is set and an interrupt is generated (if enabled). Consequently, the output compare timer pin will be set, cleared, toggled or not affected at all, according to the “mode” bits programmed. There exists also a “forced-compare” feature allowing the software to make force compares. The action taken as a result of a forced compare is the same as when an output compare match occurs, except that status flags are not set. Forced channels take programmed action immediately after the write to the force-compare bits if GPT is enabled (TE=1). These bits are self-negating and always read as zeros.

The PWM Counter (PWCNT) is a 16-bit loadable counter which starts counting after the PWM is enabled by software (PWE=1). PWCNT clock is the output of the prescaler labeled “PWM Prescaler Clock Output”. If the PWM is disabled (PWE=0), the counter freezes at its current value. If the PWM is re-enabled, the PWCR value is reloaded in the counter which starts down-counting. The PWCNT can be read at any time by the processor.

When the PWM is enabled first time after reset (PWE=1), the PWCR value is loaded in the PWCNT counter which starts down-counting. The pin is in its inactive state. When the PWCNT value equals the value preprogrammed in the PWOR, a compare match signal is generated, the PWM output compare flag is set, an interrupt is generated (if enabled) and a latch is set. This latch output is driven to the pin and optionally inverted (according to the PWP bit value). The PWCNT counter continues to down-count, sets an rollover flag at roll-over, generates an interrupt (if enabled), clears the latch whose output is driven to

the pin, reloads the PWCR value and the whole cycle repeats again. The PWCR and the PWOR values do not have to be reloaded by the MCU; the last programmed value of PWCR will be re-loaded in the PWCNT at each roll-over and the last programmed value of the PWOR will be used for the PWM output compare function. The interrupt generation and the flag updates are useful to indicate the MCU when the register value was used and can be optionally re-programmed. The duty cycle achievable at the PWM output is from 0 (when the PWOR contents equals 0) to 99.9985%=65535/65536*100 (when the PWOR contents and PWCR contents both equal $FFFF). A 100% duty cycle may be achieved by changing the output polarity and programming of the PWOR with a zero. The PWM period can vary between a minimum of 2 MCU_CLK cycles and a maximum of 65536*256 LOW-FREQ-REF cycles.

If the processor executes the DOZE instruction and the TD bit is set, then the GPT counter (TCNT) freezes at its current value. When coming out of the DOZE mode, the TCNT will resume counting from its previous value.

If the processor disables the GPT (TE bit is cleared), then the GPT counter (TCNT) freezes at its current value. When re-enabling the GPT (set the TE bit), the TCNT will resume counting from its previous value.

If the processor executes the DOZE instruction and the PWD bit is set, then the PWM counter (PWCNT) freezes at its current value. When coming out of the DOZE mode, the PWCNT will resume counting from its previous value.

If the processor disables the PWM (PWE bit is cleared), then the PWM counter (PWCNT) freezes at its current value. When re-enabling the PWM (set the PWE bit), the PWCNT will be re-loaded with the PWCR value and start down-counting.

If the processor executes the STOP instruction, then the GPT counter (TCNT) freezes at its current state. If the clock source of the PWM is selected to be LOW-FREQ-REF, its counter continues running also in stop mode. However, if PWD bit is set, also the PWM counter (PWCNT) freezes at its current value. When exiting the STOP state, the TCNT and PWCNT will resume counting from their previous value.

All the input capture register keep their values during STOP, DOZE-with TD=1 or disable.

55.4 EPIT Operation

The EPIT data register can be written to at any time by the MCU. The value written to the EPIT Data Register (ITDRL & ITDRH) determines the modulus of the timer. Data read is the present value of the modulus register. Reading the present counter value is accomplished by reading the EPIT Alternate Data Register (ITADRL & ITADRH). The data written to EPIT Compare Register (ITCRL & ITCRH) determines the timing of the compare interrupt.

The counter is clocked at one of two fixed rates depending on the state of the ECM bit. If the ECM bit is set, the clock selected is REF_CLK/512/4. With a MCU-PER-CLK of 16.8 MHz, this yields a clock period of 121.9µs. This mode is intended for use only when the phone has initially powered on and prior to the LOW-FREQ-REF setting. Software should switch to the LOW-FREQ-REF setting as soon as possible to allow the EPIT to continue operating during periods when the REF-CLK may be turned off in slotted mode. When ECM is cleared, the clock source reverts to a period of 1/8192 seconds (122.5µs) which is derived from the 32.768 kHz crystal clock divided by 4. The internal count registers are clocked using the above divide by 4 of LOW-FREQ-REF clock. In order to accurately read and write these registers via the MCU core (which runs from the MCU-CLK), clock synchronization logic is used internal to the EPIT. This logic requires that the MCU peripheral clock frequency (driven from the MCU Clock Divider), always be greater than or equal to the LOW_FREQ_REF clock that the counter clocks are derived from. Care should be taken for a case in which the mcu clock is lower than LOW-FREQ /4. In such cases, the EPIT functionality may be wrong.

55.4.1 EPIT as a “Set-and-Forget” Timer

This mode of operation is selected when the RLD bit in the EGPT Control Register (EGPTCR) is set to a value of one. The counter is not directly writable from the module data bus. instead, it gets its data from the modulus register. Whenever the counter value equals the value in the EPIT Compare Register (ITCRH & ITCRL), the interrupt flag (ITIF) is set in the EGPTSR. If the ITIE bit is set in the EGPTIR, the interrupt flag can issue an interrupt to the MCU. Whenever the counter reaches a count of zero, the value in the modulus register is loaded into the counter to be decremented toward zero. If we want the interrupt to be in case of reload, the value written in the modulus register (ITDRH & ITDRL) should be the same value written to the compare register (ITCRH & ITCRL). The counter may by directly initialized, without having to wait for the count to reach zero, when the ITDR is written with the OVW bit in the EGPTMR set.

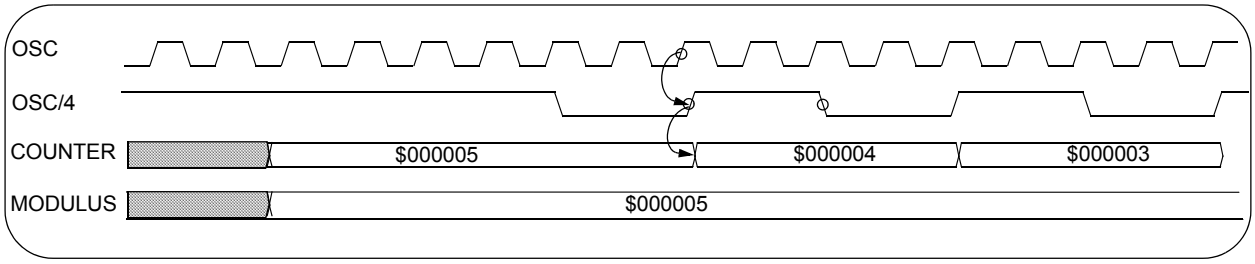


Figure 55-5. Starting a Count from an Off State

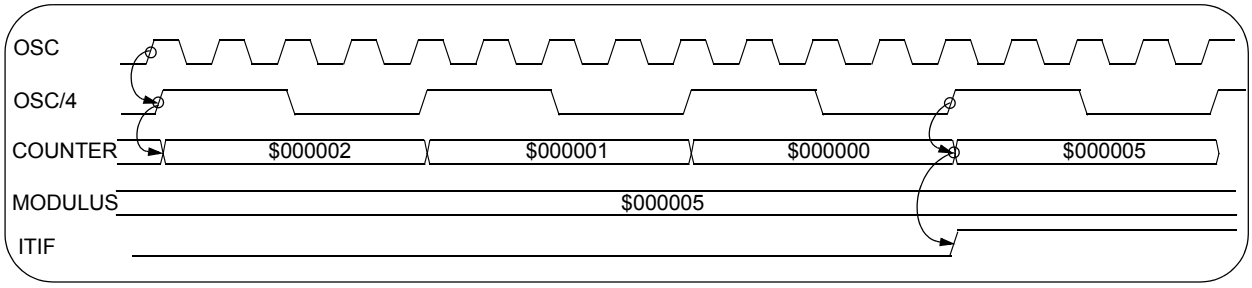


Figure 55-6. Counter Reloading from the Modulus Latch

55.4.2 EPIT as a “Free-Running” Timer

This mode of operation is selected when the RLD bit in the EGPTCR is cleared to a value of zero. In this mode, the counter rolls over from $000000 to $FFFFFF without reloading from the modulus register and continues to count.

Whenever the counter value equals the value in the EPIT Compare Register (ITCRH & ITCRL), the compare interrupt flag (ITIF) is set in the EGPTSR. If the ITIE bit is set in the EGPTIR, the interrupt flag can issue an interrupt to the MCU. In case we want the interrupt to be generated in the case of roll-over, the compare register (ITCRH & ITCRL) should be written with $FFFFFF.

The counter may by directly initialized, without having to wait for the count to reach zero, when the ITDR is written while the OVW bit is set.

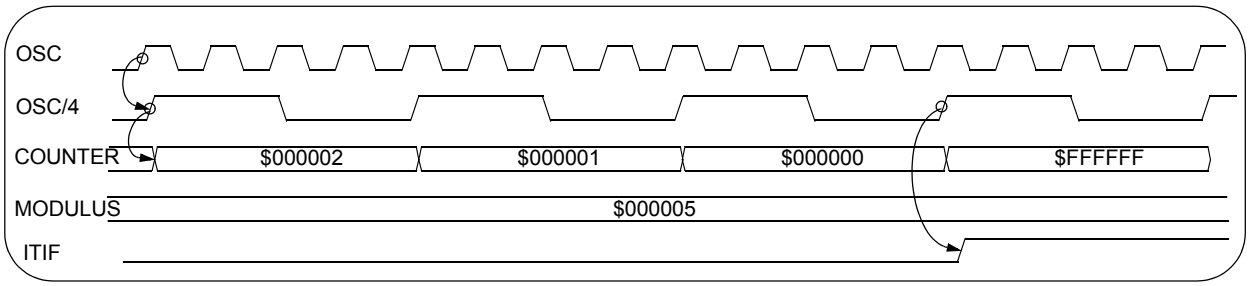


Figure 55-7. Counter in Free-Running Mode

55.4.3 EPIT in MCU LOW POWER and DEBUG MODES

55.4.3.1 EPIT in Low Power Modes

The EPIT continues to operate in STOP,DOZE and WAIT as long as the selected clock is available.

55.4.3.2 EPIT in DEBUG Mode

In DEBUG mode, the module can have the option of continuing to run or be halted. If the IDBG bit is set in the EGPT Control Register, the timer is halted. When DEBUG mode is exited, the timer operation reverts to what it was prior to entering DEBUG mode.

55.5 EGPT Programming Model

55.5.1 EGPT Register Summary

The EGPT programming model is described in the following sections. All the registers can be accessed either as a half word or as a byte. The registers in the programming model are given in Table 55-1.

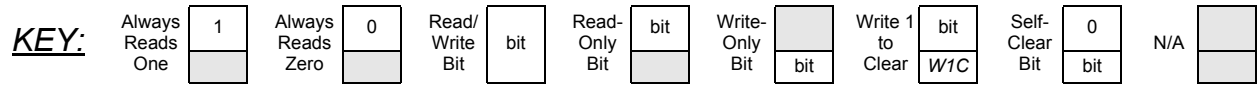


Table 55-1. EGPT Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EGPTCR ($2484_8000)	R	RLD	IDBG	ITE	PWCS	PWDBG	TDBG	PWD	PWE	TD	TE	PSPW[2:0]			PST[3:0]		
	W																
EGPTMR ($2484_8002)	R	IOVW	PWC	PWP	0	0	0	IM2[1:0]		IM1[1:0]		OM4[1:0]		OM3[1:0]		OM1[1:0]	
	W				FO4	FO3	FO1										
EGPTSR ($2484_8004)	R	0	0	0	0	0	0	0	ITIF	PWO	TOV	PWF	IF2	IF1	OF4	OF3	OF1
	W								*W1C*	*W1C*	*W1C*	*W1C*	*W1C*	*W1C*	*W1C*	*W1C*	*W1C*
EGPTIR ($2484_8006)	R	ECM	0	0	0	0	0	0	ITIE	PWOIE	TOVIE	PWFIE	IF2IE	IF1IE	OF4IE	OF3IE	OF1IE
	W																
TOCR1 ($2484_8008)	R	TOCR1[15:0]															
	W																
TOCR3 ($2484_800A)	R	TOCR3[15:0]															
	W																
TOCR4 ($2484_800C)	R	TOCR4[15:0]															
	W																
TICR1 ($2484_800E)	R	TICR1[15:0]															
	W																
TICR2 ($2484_8010)	R	TICR2[15:0]															
	W																
PWOR ($2484_8012)	R	PWOR[15:0]															
	W																
TCR ($2484_8014)	R	TCR[15:0]															
	W																
PWCR ($2484_8016)	R	PWCR[15:0]															
	W																
PWCNR ($2484_8018)	R	PWCNR[15:0]															
	W																
ITADRH ($2484_801C)	R	0	0	0	0	0	0	0	0	ITA[23:16]							
	W																
ITADRL ($2484_801E)	R	ITA[15:0]															
	W																
ITDRH ($2484_8020)	R	0	0	0	0	0	0	0	0	ITD[23:16]							
	W																
ITDRL ($2484_8022)	R	ITD[15:0]															
	W																
ITCRH ($2484_8024)	R	0	0	0	0	0	0	0	0	ITC[23:16]							
	W																
ITCRL ($2484_8026)		ITC[15:0]															

55.5.2 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various EGPT registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - — **r**: read only. Writing this bit has no effect.
  - — **w**: write only.
  - — **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - — **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - — **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - — **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - — **0**: Will reset to a logic 0
  - — **1**: Will reset to a logic 1
  - — **?**: The reset state is unknown.
  - — **u**: Unaffected by reset

The read/write 16-bit EGPT Control Register (EGPTCR) enables the GPT, PWM or EPIT, as well as their behavior in DOZE mode and Debug mode. It also controls the prescale factor for the GPT and the PWM. The EGPTCR bits are described in the following paragraphs.

EGPTCR EGPT Control Register **Addr $2484_8000**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	RLD	IDBG	ITE	PWCS	PWDBG	TDBG	PWD	PWE	TD	TE	PSPW[2:0]			PST[2:0]		
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 55-2. EGPTCR Description

Name	Description	Settings
RLD Bits 15	**EPIT Counter Reload control —**This bit is cleared by hardware reset.	RLD = 1 → When the counter reaches zero it reloads from the modulus register. RLD = 0 → When the counter reaches zero it rolls over to $FFFFFF.

Table 55-2. EGPTCR Description (Continued)

Name	Description	Settings
IDBG Bit 14	**EPIT DeBuG mode control bit—**This bit controls the function of the EPIT in Debug mode. This bit is cleared by hardware reset.	IDBG = 1 → EPIT function is frozen while in Debug mode. IDBG = 0 → EPIT function is not affected while in Debug mode.
ITE Bit 13	**EPIT Enable —**This bit is cleared by hardware reset.	ITE = 1 → EPIT is enabled. ITE = 0 → EPIT is disabled.
PWCS Bit 12	**PWM Clock Source —**The read/write control bit PWCS selects the clock source for the PWM prescaler. This bit is cleared by hardware reset.	PWCS = 0 → PWM prescaler clock output is derived from MCU-CLK. PWCS = 1→ PWM prescaler clock output is derived from LOW-FREQ-REF or MCU-CLK/512 based on ECM bit in EGPTIR.
PWDBG Bit 11	**PWM DeBuG mode enable —T**he read/write control bit PWDBG enables the operation of the PWM during Debug mode. This bit is cleared by hardware reset.	PWDBG = 0 → PWM Counter is freezed during Debug mode. PWDBG = 1→ PWM Counter continues to run during Debug mode.
TDBG Bit 10	**GPT DeBuG mode enable —**The read/write control bit TDBG enables the operation of the GPT during Debug mode. This bit is cleared by hardware reset.	TDBG = 0 → GPT Counter is freezed during Debug mode. TDBG = 1 → GPT Counter continues to run during Debug mode.
PWD Bit 9	**PWM Doze mode enable —**The read/write control bit PWD enables the operation of the PWM during DOZE mode. This bit is cleared by hardware reset.	PWD = 0 → PWM is enabled during DOZE mode. PWD = 1 → PWM is disabled during DOZE mode.
PWE Bit 8	**PWM Enable —**The read/write control bit PWE enables the operation of the PWM. If both TE and PWE bits are cleared, then the prescaler counter is stopped. This bit is cleared by hardware reset.	PWE = 0 → PWM is disabled. PWCNT is stopped. PWE = 1→ PWM is enabled. PWCNT is counting.
TD Bit 7	**GPT Doze mode enable—**The read/write control bit TD enables the operation of the GPT during DOZE mode. This bit is cleared by hardware reset.	TD = 0→ GPT is enabled during DOZE mode. TD = 1→ GPT is disabled during DOZE mode.
TE Bit 6	**GPT Enable —**The read/write control bit TE enables the operation of the GPT. This bit is cleared by hardware reset.	TE = 0 → GPT is disabled. TCNT is stopped. TE = 1→ GPT is enabled. TCNT is counting.

Table 55-2. EGPTCR Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>PSPW[2:0]
Bits 5-3</td><td>PreScaler of PWM multiplexer control —The read/write PSPW2-PSPW0 bits specify the divide ratio of the PWM prescale divider. These bits specify a power of two prescale factor in the range from 2^1 to 2^8. The clock derived from MUX between MCU-CLK and LOW-FREQ-REF is divided according to this prescale factor. The table on the right shows the programming of the PSPW2-PSPW0 bits. These bits are cleared (prescale by two) after hardware reset</td><td>
<table>
<tr><th>PSPW[2:0]</th><th>Prescale Factor PSPW</th></tr>
<tr><td>$0</td><td>$2^1$</td></tr>
<tr><td>$1</td><td>$2^2$</td></tr>
<tr><td>$2</td><td>$2^3$</td></tr>
<tr><td>$3</td><td>$2^4$</td></tr>
<tr><td>$4</td><td>$2^5$</td></tr>
<tr><td>$5</td><td>$2^6$</td></tr>
<tr><td>$6</td><td>$2^7$</td></tr>
<tr><td>$7</td><td>$2^8$</td></tr>
</table>
</td></tr>
<tr><td>PST[2:0]
Bits 2-0</td><td>PreScaler of Timer multiplexer control—The read/write PST2-PST0 bits specify the divide ratio of the Timer prescale divider. These bits specify any power of two prescale factor in the range from 2^1 to 2^8. The clock derived from the MCU-CLK is divided according to this prescale factor. The table on the right shows the programming of the PST2-PST0 bits. These bits are cleared (prescale by two) after hardware reset.</td><td>
<table>
<tr><th>PST[2:0]</th><th>Prescale Factor PST</th></tr>
<tr><td>$0</td><td>$2^1$</td></tr>
<tr><td>$1</td><td>$2^2$</td></tr>
<tr><td>$2</td><td>$2^3$</td></tr>
<tr><td>$3</td><td>$2^4$</td></tr>
<tr><td>$4</td><td>$2^5$</td></tr>
<tr><td>$5</td><td>$2^6$</td></tr>
<tr><td>$6</td><td>$2^7$</td></tr>
<tr><td>$7</td><td>$2^8$</td></tr>
</table>
</td></tr>
</table>

The read/write 16-bit EGPT Mode Register (EGPTMR) controls the operating mode of each one of the GPT/PWM input/output register and the output signal polarity of the PWM. It also controls the OVW bit of EPIT. The EGPTMR bits are described in the following paragraphs.

EGPTMR EGPT Mode Register **Addr $2484_8002**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	IOVW	PWC	PWP	FO4	FO3	FO1	IM2[1:0]		IM1[1:0]		OM4[1:0]		OM3[1:0]		OM1[1:0]	
TYPE	rw	rw	rw	slfclr	slfclr	slfclr	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 55-3. EGPTMR Description

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>IOVW
Bits 15</td><td>EPIT Counter Overwrite Enable—This bit controls what happens to the counter data when the modulus register is written.</td><td>IOVW = 1 → Modulus latch is transparent. All writes to the latch will also overwrite the counter contents.
IOVW = 0 → Modulus latch is a holding register for values to be loaded into the counter when the count expires to zero</td></tr>
<tr><td>PWC
Bit 14</td><td>PWm Control—The read/write control bit PWC controls the connection of the PWM to the output pin.This bit is cleared by hardware reset.</td><td>PWC = 0 → PWM disconnected from the pin.
PWC = 1→ PWM connected to the pin.</td></tr>
<tr><td>PWP
Bit 13</td><td>PWm pin Polarity (PWP)—The read/write control bit PWP controls the polarity of the PWM output pin. The pin “active” time is the one during the time period between the compare-occurred and the PWCNT roll-over. This bit is cleared by hardware reset.</td><td>PWP = 0 → Active-high polarity.
PWP = 1 → Active-low polarity.</td></tr>
<tr><td>FO4
Bit 12</td><td>Force Output compare timer 4 —The read/write control bit FO4 causes the pin action programmed for the timer output compare 4 pin (according to the bits OM41-OM40). The OF4 flag is not affected. This bit is self negating.</td><td>FO4 = 0 → Writing a 0 to the FO3 bit has no meaning.
FO4 = 1 → Causes pin action on timer output compare 4 pin; OF4 flag is not set.</td></tr>
<tr><td>FO3
Bit 11</td><td>Force Output compare timer 3—The read/write control bit FO3 causes the pin action programmed for the timer output compare 3 pin (according to the bits OM31-OM30). The OF3 flag is not affected. This bit is self negating.</td><td>FO3 = 0 → Writing a 0 to the FO2 bit has no meaning.
FO3 = 1 → Causes pin action on timer output compare 3 pin; OF3 flag is not set.</td></tr>
<tr><td>FO1
Bit 10</td><td>Force Output compare timer 1—The read/write control bit FO1 causes the pin action programmed for the timer output compare 1 pin (according to the bits OM11-OM10). The OF1 flag is not affected.This bit is self negating.</td><td>FO1 = 0 → Writing a 0 to the FO1 bit has no meaning.
FO1 = 1 → Causes pin action on timer output compare 1 pin; OF1 flag is not set.</td></tr>
<tr><td>IM2[1:0]
Bits 9-8</td><td>Input capture timer 2 operating Mode —The read/write IM21-IM20 bits specify the operating mode of the Timer 2 (Input Capture). Table on right shows the programming of the IM21-IM20 bits. These bits are cleared after hardware reset.</td><td>
<table>
<tr><th>IM2[1:0]</th><th>Operating Mode</th></tr>
<tr><td>00</td><td>Capture Disabled</td></tr>
<tr><td>01</td><td>Capture On Rising Edge Only</td></tr>
<tr><td>10</td><td>Capture On Falling Edge Only</td></tr>
<tr><td>11</td><td>Capture On Any Edge</td></tr>
</table>
</td></tr>
</table>

Table 55-3. EGPTMR Description (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>IM1[1:0]
Bit 7-6</td><td>Input capture timer 1 operating Mode —The read/write IM11-IM10 bits specify the operating mode of the Timer 1 (Input Capture). Table on the right shows the programming of the IM11-IM10 bits. These bits are cleared after hardware reset</td><td>
<table>
<tr><th>IM1[1:0]</th><th>Operating Mode</th></tr>
<tr><td>00</td><td>Capture Disabled</td></tr>
<tr><td>01</td><td>Capture On Rising Edge Only</td></tr>
<tr><td>10</td><td>Capture On Falling Edge Only</td></tr>
<tr><td>11</td><td>Capture On Any Edge</td></tr>
</table>
</td></tr>
<tr><td>OM4[1:0]
Bit 5-4</td><td>Output compare timer 4 operating Mode —The read/write OM41-OM40 bits specify the operating mode of the Timer 4 (Output Compare). Table on the right shows the programming of the OM41-OM40 bits. These bits are cleared after hardware reset.</td><td>
<table>
<tr><th>OM4[1:0]</th><th>Operating Mode</th></tr>
<tr><td>00</td><td>Timer Disconnected From Pin</td></tr>
<tr><td>01</td><td>Toggle Output Pin</td></tr>
<tr><td>10</td><td>Clear Output Pin</td></tr>
<tr><td>11</td><td>Set Output Pin</td></tr>
</table>
</td></tr>
<tr><td>OM3[1:0]
Bits 3-2</td><td>Output compare timer 3 operating Mode—The read/write OM31-OM30 bits specify the operating mode of the Timer 3 (Output Compare). The table to the right shows the programming of the OM31-OM30 bits. These bits are cleared after hardware reset.</td><td>
<table>
<tr><th>OM3[1:0]</th><th>Operating Mode</th></tr>
<tr><td>00</td><td>Timer Disconnected From Pin</td></tr>
<tr><td>01</td><td>Toggle Output Pin</td></tr>
<tr><td>10</td><td>Clear Output Pin</td></tr>
<tr><td>11</td><td>Set Output Pin</td></tr>
</table>
</td></tr>
<tr><td>OM1[1:0]
Bits 1-0</td><td>Output compare timer 1 operating Mode—The read/write OM11-OM10 bits specify the operating mode of the Timer 1 (Output Compare). The table on the right shows the programming of the OM11-OM10 bits. These bits are cleared after hardware reset.t</td><td>
<table>
<tr><th>OM1[1:0]</th><th>Operating Mode</th></tr>
<tr><td>00</td><td>Timer Disconnected From Pin</td></tr>
<tr><td>01</td><td>Toggle Output Pin</td></tr>
<tr><td>10</td><td>Clear Output Pin</td></tr>
<tr><td>11</td><td>Set Output Pin</td></tr>
</table>
</td></tr>
</table>

The read/write 16-bit EGPT Status Register (EGPTSR) indicates the occurrence of the Timer Input Capture, Timer or PWM Output Compare, Timer or PWM counter overflow and EPIT compare occurrence. The EGPTSR bits are described in the following paragraphs.

EGPTSR EGPT Status Register **Addr $2484_8004**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
								ITIF	PWO	TOV	PWF	IF2	IF1	OF4	OF3	OF1
TYPE	r	r	r	r	r	r	r	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 55-4. EGPTSR Description

Name	Description	Settings
Bits 15-9	**Reserved —**These bits are read as zero's and should be written with zero's for future compatibility.	N/A
ITIF Bits 8	**EPIT Interrupt Flag—**This bit is the interrupt flag that is set when the content of counter equals the content of the compare register (ITCRH concatenated with ITCRL). It is cleared by writing a one to this bit or by writing to the EPIT High Compare Register (ITCRH) or by writing to the EPIT Low Compare Register (ITCRL) or by writing to the EPIT Low Data Register (ITDRL).	ITIF = 1 → EPIT compare interrupt is pending. ITIF = 0 → EPIT compare interrupt is not pending.
PWO Bit 7	**PWm counter rollOver—**The read/write PWO bit indicates if the PWM counter (PWCNT) rollover has occurred.This bit can be cleared when writing it with a one. Writing a zero has no effect. This bit is cleared after hardware reset.	PWO = 0 → PWCNT rollover not occurred. PWO = 1 → PWCNT rollover occurred.
TOV Bit 6	**Timer counter OVerflow —**The read/write TOV bit indicates if the Timer counter (TCNT) overflow has occurred. This bit can be cleared when writing it with a one. Writing a zero has no effect. This bit is cleared after hardware reset.	TOV = 0 → TCNT Overflow not occurred. TOV = 1 → TCNT Overflow occurred.
PWF Bit 5	**PWm output compare Flag—**The read/write WF bit indicates if the PWM compare occurred.This bit can be cleared when writing it with a one. Writing a zero has no effect. This bit is cleared after hardware reset.	PWF = 0 → Compare not occurred. PWF = 1 → Compare occurred.
IF2 Bit 4	**Input capture timer 2 Flag—**The read/write IF2 bit indicates if the capture occurred.This bit can be cleared when writing it with a one. Writing a zero has no effect. This bit is cleared after hardware reset.	OF2 = 0 → Capture not occurred. OF2 = 1 → Capture occurred.
IF1 Bit 3	**Input capture timer 1 Flag —**The read/write IF1 bit indicates if the capture occurred.This bit can be cleared when writing it with a one. Writing a zero has no effect. This bit is cleared after hardware reset.	IF1 = 0 → Capture not occurred. IF1 = 1 → Capture occurred.

Table 55-4. EGPTSR Description (Continued)

Name	Description	Settings
OF4 Bit 2	**Output compare timer 4 Flag—**The read/write OF4 bit indicates if the compare occurred. This bit can be cleared when writing it with a one. Writing a zero has no effect. This bit is cleared after hardware reset.	OF4 = 0 → Compare not occurred. OF4 = 1 → Compare occurred.
OF3 Bit 1	**Output compare timer 3 Flag—**The read/write OF3 bit indicates if the compare occurred. This bit can be cleared when writing it with a one. Writing a zero has no effect. This bit is cleared after hardware reset.	OF3 = 0 → Compare not occurred. OF3 = 1 → Compare occurred.
OF1 Bit 0	**Output compare timer 1 Flag —**The read/write OF1 bit indicates if the compare occurred. This bit can be cleared when writing it with a one. Writing a zero has no effect. This bit is cleared after hardware reset.	OF1 = 0 → Compare not occurred. OF1 = 1 → Compare occurred.

The read/write 16-bit EGPT Interrupts enable Register (EGPTIR) enables individually each one of the EGPT interrupts. The EGPTIR bits are described in the following paragraphs.

EGPTIR EGPT Interrupts enable Register **Addr $2484_8006**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	ECM							ITIE	PWOIE	TOVIE	PWFIE	IF2IE	IF1IE	OF4IE	OF3IE	OF1IE
TYPE	rw	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 55-5. EGPTIR Description

Name	Description	Settings
ECM Bit15	**EPIT Clock Mode (ECM)—**Selects the CKIL clock used by EPIT and PWM.	ECM = 1 → The CKIL clock feeding EPIT and optionally also the PWM is the REF-CLK divided by 512. ECM = 0 → The CKIL clock feeding EPIT and optionally also the PWM is the chip CKIL coming externally (32 KHz input).
Bits 14-9	**Reserved —**These bits are read as zeros and should be written with zeros for future compatibility.	N/A
ITIE Bit 8	**EPIT Interrupt Enable (ITIE)—**Generates interrupt requests.	ITIE = 1 → EPIT generates an interrupt request to the MCU interrupt controller. ITIE = 0 → EPIT interrupt is disabled.
PWOIE Bit 7	**PWm counter rollOver Interrupt Enable—** The read/write PWOIE bit enables the PWM counter (PWCNT) rollover interrupt. This bit is cleared after hardware reset.	PWOIE = 0 → Interrupt disabled. PWOIE = 1 → Interrupt enabled.
TOVIE Bit 6	**Timer counter OVerflow—** The read/write TOVIE bit enables the Timer counter (TCNT) overflow interrupt. This bit is cleared after hardware reset.	TOVIE = 0 → Interrupt disabled. TOVIE = 1 → Interrupt enabled.

Table 55-5. EGPTIR Description (Continued)

Name	Description	Settings
PWFIE Bit 5	**PWm output compare Interrupt Enable —**The read/write PWFIE bit enables the PWM Output Compare interrupt. This bit is cleared after hardware reset.	PWFIE = 0 → Interrupt disabled. PWFIE = 1 → Interrupt enabled.
IF2IE Bit 4	**Input capture timer 2 Interrupt Enable —**The read/write IF2IE bit enables the Timer 2 Input Capture interrupt. This bit is cleared after hardware reset.	OF2IE = 0 → Interrupt disabled. OF2IE = 1 → Interrupt enabled.
IF1IE Bit 3	**Input capture timer 1 Interrupt Enable—** The read/write IF1IE bit enables the Timer 1 Input Capture interrupt. This bit is cleared after hardware reset.	IF1IE = 0 → Interrupt disabled. IF1IE = 1 → Interrupt enabled.
OF4IE Bit 2	**Output compare timer 4 Interrupt Enable—**The read/write OF4IE bit enables the Timer 4 Output Compare interrupt. This bit is cleared after hardware reset.	OF4IE = 0 → Interrupt disabled. OF4IE = 1 → Interrupt enabled.
OF3IE Bit 1	**Output compare timer 3 Interrupt Enable—** The read/write OF3IE bit enables the Timer 3 Output Compare interrupt. This bit is cleared after hardware reset.	OF3IE = 0 → Interrupt disabled. OF3IE = 1 → Interrupt enabled.
OF1IE Bit 0	**Output compare timer 1 Interrupt Enable—**The read/write OF1IE bit enables the Timer 1 Output Compare interrupt. This bit is cleared after hardware reset.	OF1IE = 0 → Interrupt disabled. OF1IE = 1 → Interrupt enabled.

TOCR1 Timer 1 Output Compare Register **Addr $2484_8008**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	TOCR1[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 55-6. TOCR1 Description

Name	Description	Settings
TOCR1[15:0] Bits 15-0	**Timer 1 Output Compare Register—**The read/write 16-bit Timer 1 Output Compare Register (TOCR1) contains the 16-bit Output Compare value of the Timer 1. The TOCR1 bits are cleared after hardware reset.	N/A

TOCR3 Timer 3 Output Compare Register **Addr $2484_800A**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	TOCR3[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 55-7. TOCR3 Description

Name	Description	Settings
TOCR3[15:0] Bits 15-0	**Timer 3 Output Compare Register—**The read/write 16-bit Timer 3 Output Compare Register (TOCR3) contains the 16-bit Output Compare value of the Timer 3. The TOCR3 bits are cleared after hardware reset.	N/A

TOCR4 Timer 4 Output Compare Register **Addr $2484_800C**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	TOCR4[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 55-8. TOCR4 Description

Name	Description	Settings
TOCR4[15:0] Bits 15-0	**Timer 4 Output Compare Register—**The read/write 16-bit Timer 4 Output Compare Register (TOCR4) contains the 16-bit Output Compare value of the Timer 4. The TOCR4 bits are cleared after hardware reset.	N/A

TICR1 Timer 1 Input Capture Register — Addr $2484_800E

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	TICR1[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 55-9. TICR1 Description

Name	Description	Settings
TICR1[15:0] Bits 15-0	**Timer 1 Input Capture Register—**The read-only 16-bit Timer 1 Input Capture Register (TICR1) contains the 16-bit value captured by the Timer 1. The TICR1 bits are cleared after hardware reset.	N/A

TICR2 Timer 2 Input Capture Register — Addr $2484_8010

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	TICR2[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 55-10. TICR2 Description

Name	Description	Settings
TICR2[15:0] Bits 15-0	**Timer 2 Input Capture Register—**The read-only 16-bit Timer 2 Input Capture Register (TICR2) contains the 16-bit value captured by the Timer 2. The TICR2 bits are cleared after hardware reset.	N/A

PWOR PWM Output compare Register **Addr $2484_8012**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PWOR[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 55-11. PWOR Description

Name	Description	Settings
PWOR[15:0] Bits 15-0	**PWm Output compare Register—**The read/write 16-bit PWM Output compare Register (PWOR) contains the 16-bit Output Compare value of the PWM. The PWOR bits are cleared after hardware reset.	N/A

TCR Timer Counter Register **Addr $2484_8014**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	TCR[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 55-12. TCR Description

Name	Description	Settings
TCR[15:0] Bits 15-0	**Timer Counter Register**—The read-only 16-bit Timer Counter Register (TCR) contains the 16-bit value of the Timer Counter (TCNT). The TCR bits are cleared after hardware reset.	N/A

PWCR PWm Count Register **Addr $2484_8016**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PWCR[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 55-13. PWCR Description

Name	Description	Settings
PWCR[15:0] Bits 15-0	**PWm Count Register** —The read/write 16-bit PWm Count Register (PWCR) can be written with the PWM clock cycle count value which is loaded into the PWM Counter (PWCNT) and determines its roll-over period. The PWCNT roll-over period equals the value loaded + 1.	N/A

PWCNR PWm CouNter Register **Addr $2484_8018**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PWCNR[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 55-14. PWCNR Description

Name	Description	Settings
PWCNR[15:0] Bits 15-0	**PWm CouNter Register** —The read-only 16-bit PWm CouNter Register (PWCNR) can be used to read the 16-bit value of the PWM Counter (PWCNT) at a specific moment. The PWCNR bits are cleared after hardware reset.	N/A

ITADRH EPIT High Alternate Data Register **Addr $2484_801C**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									ITA{23:16]							
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	u	u	u	u	u	u	u	u

Table 55-15. ITADRH Description

Name	Description	Settings
Bits 15-8	**Reserved bits** — Read as 0.	N/A
ITA[23:16] Bits 7-0	**EPIT High Alternate Data Register** —The EPIT Alternate Data Register High byte is a 16-bit read-only register which provides access to the 24-bit counter 8 MSB bits. It's value is not valid unless ITADRL register was read earlier. In order to read the counter value, these registers should be read twice. On the first read, operation bits 15-0 of the counter will be placed in ITADRL and in the second read, operation bits 23-16 of the counter will be placed in ITADRH[7:0]. Internally these registers are implemented as a two stage buffer, and when the user reads the register for the first time, the upper part of the counter value is already latched in the buffer. thus, the two read operations will be correlated. This register is not byte accessible.	N/A

ITADRL EPIT Low Alternate Data Register **Addr $2484_801E**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	ITA[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Table 55-16. ITADRL Description

Name	Description	Settings
ITA[15:0] Bits 15-0	**EPIT Low Alternate Data Registers** —The EPIT Alternate Data Register Low word is a 16-bit read only register which provides access to the 24- bit counter 16 LSB bits. In order to read the counter present value, this register should be read first & the 16 LSB bits will be shown. At the same time the 8 MSB bits of the counter value will be latched to ITADRH. This register is not byte accessible.	N/A

ITDRH EPIT High Data Register **Addr $2484_8020**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									ITD23	ITD22	ITD21	ITD20	ITD19	ITD18	ITD17	ITD16
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Table 55-17. ITDRH Description

Name	Description	Settings
Bits 15-8	**Reserved Bits**—Read as zero. Must be written with zero for future compatibility.	N/A
ITD[23:16] Bits 7-0	**EPIT High Data Register**—This register holds the 8 MSBs of the value the counter will be reloaded with when it reaches a value of $000000. The write operation to ITDRH does not affect the counter operation. Only a write to EPIT Low Data Register (see below) will affect the counter. On a read operation, the ITDRH returns the value written in the 8 MSBs of the modulus register. This register should only be written when the ITE bit is set in the EGPTCR register.	N/A

ITDRL EPIT Low Data Register **Addr $2484_8022**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	IDT[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 55-18. ITDRL Description

Name	Description	Settings
ITD[15:0] Bits 15-0	**EPIT Low Data Register** — On a write operation to this register, the content of both ITDRH and ITDRL become the new timer modulus. This value is retained and is used at the next and all subsequent reloads until changed by another write to ITDRL. This value is initialized to the maximum count of $FFFFFF on Reset. On a read operation, the ITDRL returns the value written in the 16 LSBs of the modulus register. The only way to directly change the value of the count is to preload a modulus with the OVW bit set to one. The counter value can be read from the EPIT Alternate Data Register. This register should only be written when the ITE bit is set in the EGPTCR register.	N/A

ITCRH EPIT High Compare Data Register **Addr $2484_8024**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									ITC23	ITC22	ITC21	ITC20	ITC19	ITC18	ITC17	ITC16
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Table 55-19. ITCRH Description

Name	Description	Settings
Bits 15-8	**Reserved Bits**—Read as zero. Must be written with zero for future compatibility.	N/A
ITC[23:16] Bits 7-0	**EPIT Compare High Register** —This register is a 16-bit read/write register. it holds the 8 MSBs of the value the compare register.The write operation to ITCRH does not affect the counter operation. Only a write to EPIT Low Compare Register (see below) will affect the counter. When the 24 bit counter equals the concatenated value of ITCRH and ITCRL, the ITIF flag in EGPTSR is set, and an interrupt can be initiated to the MCU.	N/A

ITCRL EPIT Compare Low Register **Addr $2484_8026**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	ITC[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 55-20. ITCRL Description

Name	Description	Settings
ITC[15:0] Bits 15-0	**EPIT Compare Low Register** —This register is a 16-bit read/write register. it holds the 16 LSBs of the value the compare register. When the 24 bit counter equals the concatenated value of ITCRH and ITCRL, the ITIF flag in EGPTSR is set, and an interrupt can be initiated to the MCU.	N/A

55.6 Pinout

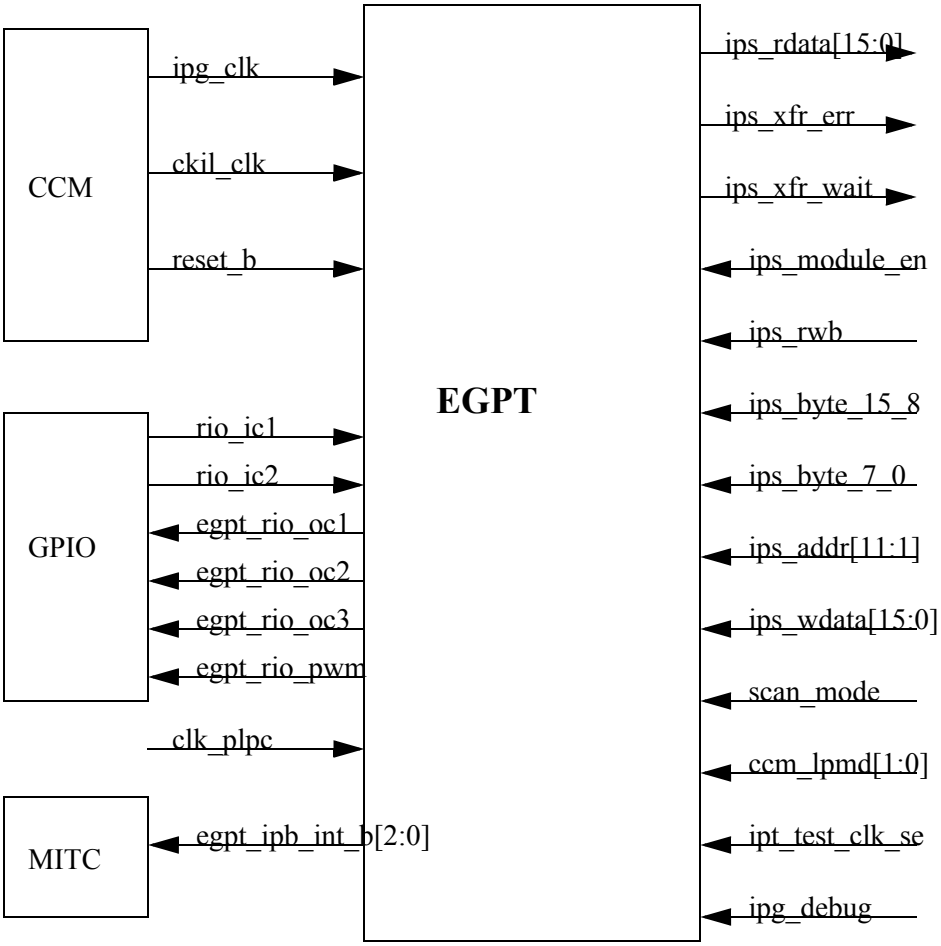
Table 55-21. EGPT Module Pin List

Name	Dir	Description
ipg_clk	input	IP bus peripheral clock
ckil_clk	input	Low-frequency reference clock (32 kHz)
clk_plpc	input	Clock controller's PLL low-power control bit when clr,indicates mcu _clk cease in wait or doze also
rio_ic1	input	Input compare #1 pin for General Purpose Timer
rio_ic2	input	Input compare #2 pin for General Purpose Timer
reset_b	input	Reset
IP Bus		
ips_addr[11:1]	input	IP Bus- Module address bus
ips_rwb	input	IP Bus- Module read/write
ips_module_en	input	IP Bus - Module select
ips_byte_15_8	input	IP Bus-write enable for Upper 8 bits
ips_byte_7_0	input	IP Bus-write enable for Lower 8 bits
ccm_lpmd[1:0]	input	CCM - Low power mode bits
ipg_debug	input	IP Bus - Debug Signal
ips_xfr_wait	output	IP bus Module transfer acknowledge

Table 55-21. EGPT Module Pin List

Name	Dir	Description
ips_xfr_err	output	IP Bus Module transfer error acknowledge
ips_rdata[15:0]	output	IP Bus - Module read data bus
ips_wdata[15:0]	input	IP Bus - Module write data bus
Scan		
scan_mode	input	Scan mode indicator
ipt_test_clk_se	input	IP Bus Scan shift enable
Output Compare		
egpt_rio_pwm	output	Output compare pin for PWM
egpt_rio_oc1	output	Ourput compare #1 pin for General Purpose Timer
egpt_rio_oc2	output	Ourput compare #2 pin for General Purpose Timer
egpt_rio_oc3	output	Ourput compare #3 pin for General Purpose Timer
egpt_ipb_int_b[2:0]	output	interrupt bus [0] = epit [1] = pwm [2] = gtim & pwm

55.7 Pin Diagram


CCM
ipg_clk
ckil_clk
reset_b
GPIO
rio_ic1
rio_ic2
egpt_rio_oc1
egpt_rio_oc2
egpt_rio_oc3
egpt_rio_pwm
clk_plpc
MITC
egpt_ipb_int_b[2:0]
EGPT
ips_rdata[15:0]
ips_xfr_err
ips_xfr_wait
ips_module_en
ips_rwb
ips_byte_15_8
ips_byte_7_0
ips_addr[11:1]
ips_wdata[15:0]
scan_mode
ccm_lpmd[1:0]
ipt_test_clk_se
ipg_debug

Chapter 56
Real Time Reference (RTR)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	03/19/02	Vinay Kumar P P	Made IP BUS compliant
0.2	05/07/03		Updated for LTE specification release.

The Real Time Reference (RTR) peripheral provides software with a time reference from the last system reset event and with a periodic interrupt source with programmable resolution for code profiling. RTR features include the following:

- Timer Counter
  - one second count unique over 136 years period
  - external clock source from 32.768 kHz crystal
  - 47-bit counter read as 32 (MSB) and 15 (LSB) bits by software
  - software detection of rollover condition
- Interrupt Generator
  - 30.5 micro-second to 1 second programmable rate
  - source is a 0-to-1 transition on one of the counter 16 LSB bits
- Low Power Support
  - IPG_CLK gated in WAIT, DOZE, and STOP modes
  - Software on/off control of CKIL_CLK source
- Implementation
  - 32-bit register interface to IP_BUS
  - Bus exceptions (ips_xfr_err assertion) for invalid read/write accesses
  - Fully scannable design

56.1 RTR Overview

A block diagram of the RTR module is shown in Figure 56-1. The module is partitioned into four functional blocks. These blocks consist of the register interface, the interrupt generator, and the 47-bit counter. The synchronizers between the CKIL_CLK and IPG_CLK (ips_rdata/ips_wdata) clock domains are not shown, but are necessary for a complete RTR implementation.

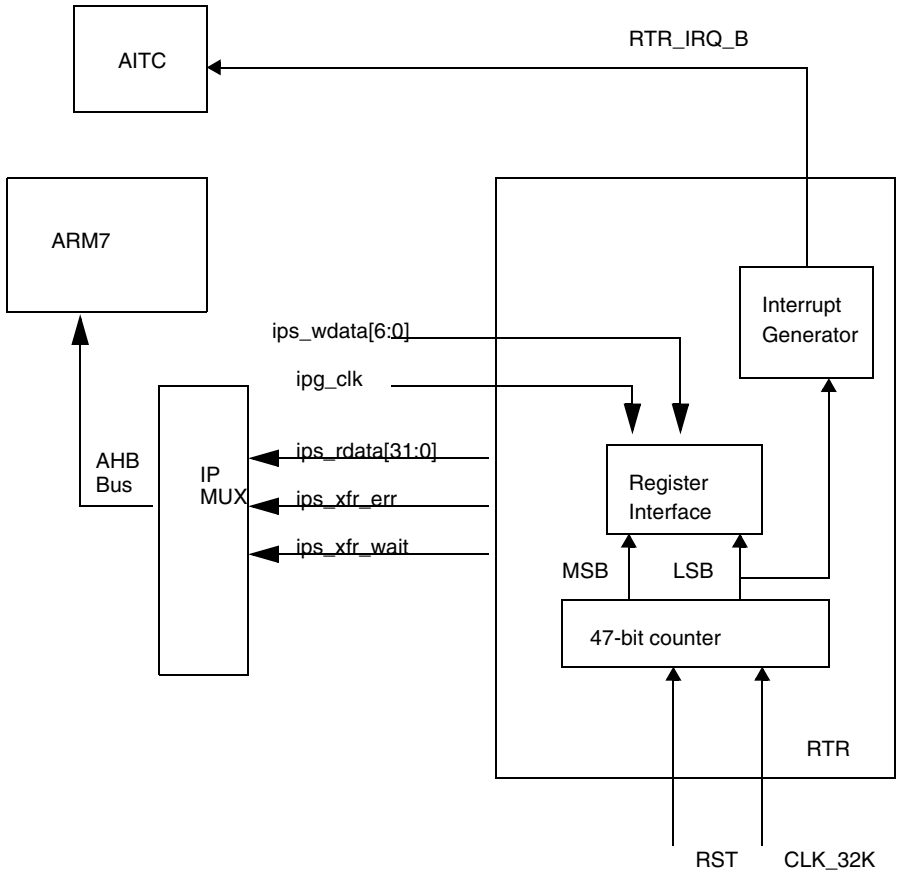


Figure 56-1. RTR Block Diagram

56.1.1 RTR Module Pin List

The RTR module connects to the PIG and AITC peripherals in the Neptune IC. Table 56-1 contains a description of the interface signals and their directions.

Table 56-1. RTR Module Pin List

Pin Name	Direction	Description
IP BUS Interface Signals		
ips_module_en	input	Module select, active low. Asserted when IPBUS decodes a valid address.
ips_addr[11:1]]	input	Module address bus.
ips_rwb	input	Module read/write signal
ips_byte_7_0	input	Byte enable for 7-0
ips_rdata[31:0]	output	Read data bus to IP Bus
ips_wdata[31:0]	input	Write data from IP Bus

Table 56-1. RTR Module Pin List

Pin Name	Direction	Description
ips_xfr_err	output	Module Transfer Error Acknowledge
MCU Interface Signals		
sys_rst_b	input	Active-low signal used to reset the Neptune IC.
ipg_clk	input	Source MCU Clock.
clk32k	input	32 kHz clock used by the 47-bit counter. Available during Deep Sleep mode.
AITC Interface Signals		
rtr_irq_b	output	Active low interrupt for 0-to-1 changes in any one of the 16 LSB counter bits. Maskable, write-one to clear.
Scan Divergence Signals		
ipt_scan_mode	input	Indication that we are in the scan mode.
ipt_clk_se	input	scan enable Used in clock gating cell

56.2 Functional Description

The RTR is a 47-bit read-only up counter that is intended to provide software with a time reference from the last system reset event and a periodic interrupt source with 30.5 micro-seconds resolution for code profiling. The counter is reset by the Neptune active-low Reset ($\overline{\text{RST}}$) signal. The RTR is clocked at a fixed 32.768 kHz rate (CKIL_CLK) during all Neptune operational modes, including the Deep Sleep mode. The counter reaches the zero value on roll-over (1193046.471 hours at 32.768 kHz) or two CKIL_CLK cycle after a system reset event. A 0-to-1 change in any one of the 16 least significant counter bits can generate a maskable interrupt on the RTR_IRQ_B line. Interrupt periods from 61 micro-seconds (RTR_LSB[17], **RTR[0]** source, see register map in Table 56-2) to 0.5 second (RTR_LSB[31], **RTR[14]** source) are supported. If a write access to the read-only RTR_LSB or RTR_MSB registers is attempted by software, an error signal (mtea_b_z) is generated to IP_BUS and an external exception is taken by the MCU . Similarly, a read access to an invalid RTR register address will cause the MCUto take an external exception. Counter rollover can be detected in software with a third read access to the RTR_MSB register. RTR_MSB contains the 32 most significant bits of the 47-bit value (seconds count), and RTR_LSB contains the remaining 15 least significant bits in RTR_LSB[31:17]. RTR_LSB[16:0] are always read as zeros. The RTR is a zero wait-state module and interfaces to the MCU via IPBUS. See the IPBUS specification for details on the IPBUS interface. The RTR counter can be disabled for power savings by setting the **disable** bit in the control register, RTR_CTRL (see Section 56.6, “Register Summary,”). This bit gates the external 32 kHz clock to the counter. The 47-bit value remains unchanged until software re-enables the 32 kHz clock (clears the **disable** bit) or triggers a system reset event. The RTR registers are not available during MCU DOZE, WAIT, and STOP low power modes.

56.3 Module Architecture

The RTR is accessed by the MCU via the IPBUS interface . The register data interface is 32 bits wide, and a read request to RTR_LSB returns the 15 lsb bits of the counter value in RTR_LSB[31:16], with zero in RTR_LSB[16]. Note that because of this requirement the upper 16 bits of a 32-bit ips_rdata bus transfer will not match the value contained in RTR_LSB[31:17]. The external 32 kHz clock source is synchronized to IPG_CLK, CLK_32K_sync in **Figure 56-2**, and the synchronized version is used to update the contents of the RTR_MSB and RTR_LSB registers on each read access to the module (ips_rwb and ips_module_en asserted) .

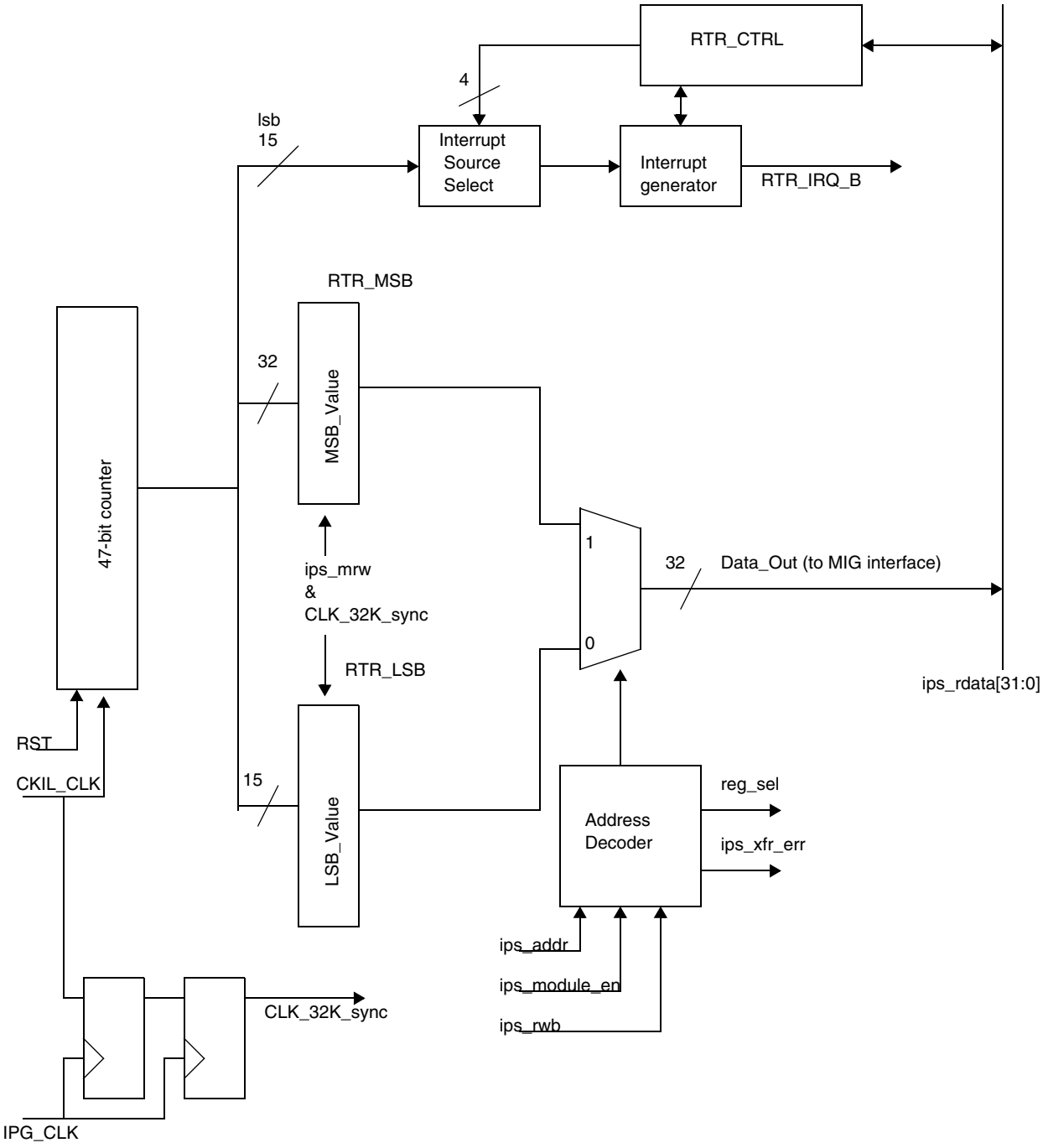


Figure 56-2. RTR Block Diagram Detail

Figure 56-3. RTR Read Cycle, Zero Module-inserted Wait States

56.4 Module Interrupts

The RTR generates an interrupt request to the MCU interrupt controller, AITC, via the RTR_IRQ_B active-low signal as show in **Figure 56-1**. The s**rc[3:0]** bits in the RTR_CTRL register (see register map in **Section 56.6, "Register Summary,"**) select one interrupt source from the 16 LSB bits in the RTR counter. A 0-to-1 transition on the selected interrupt source sets the status bit in the RTR_CTRL register. The RTR_IRQ_B interrupt line is asserted if the mask bit in RTR_CTRL has been cleared. The status bit is cleared by writing a one to it which also sets RTR_IRQ_B high. Interrupts will not be generated during the Neptune Deep Sleep state if masked by software. Interrupts can be cleared only if IPG_CLK is available to the module.

RTR_IRQ_B is synchronized to the ARM7 inside the AITC module.

56.5 RTR Clocking and power management

The RTR module uses an external 32 KHz clock source for the 47-bit counter and IPG_CLK for the register interface to the IP module. IPG_CLK frequency is programmable in the MCU DPLL, from 13 MHz to 52 MHz, and the RTR will gate this clock for power management in the DOZE, WAIT, and STOP modes. The ccm_lpmd[1:0] signals from the ccm module are used for gating IPG_CLK. The **disable** bit in the RTR_CTRL register can be used to disable the external 32 KHz clock source for further power savings. The RTR counter will continue to run in WAIT, DOZE, and STOP modes of the MCU if the **disable** bit has not been set. After exit from WAIT, DOZE, or STOP mode, the RTR counter value will not be valid until after a positive edge of the 32 kHz clock.

56.6 Register Summary

Figure 0-15.

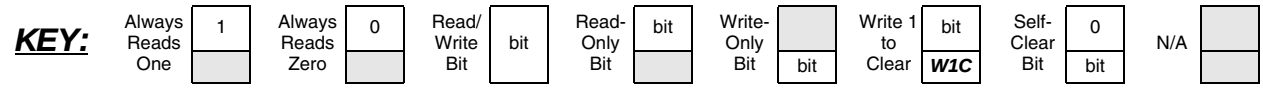


Table 56-2. RTR Register Map Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTR_MSB ($2484_A000)	R	RTR[46:31]															
	W																
	R	RTR[30:15]															
	W																
RTR_LSB ($2484_A004)	R	RTR[14:0]															0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
RTRC ($2484_A008)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	DIS	SRC[3:0]				STAT	MSK
	W															W1C	

56.6.1 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various RTR registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0.
  - **1**: Will reset to a logic 1.
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset.

RTR_MSB — RTR Most Significant Bits Register — Addr $2484_A000

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	RTR[46:31]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	RTR[30:15]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Name	Description	Settings
RTR[46:15] Bits 31-0	**RTR Most Significant Bits —** These are the 32 most significant bits of the 47-bit counter value. Bit 0 (**RTR[15]**) is the seconds count.	N/A

RTR_LSB — RTR Least Significant Bits Register — Addr $2484_A004

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	RTR[14:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Name	Description	Settings
RTR[14:0] Bits 31-17	**RTR Least Significant Bits —** These are the 15 least significant bits of the 47-bit counter value.	N/A
Bits 17-0	**Reserved bits—** The lower 17 bits are not implemented and read as zeros.	N/A

RTRC — RTR Control Register — Addr $2484_A008

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
										DIS	SRC[3:0]				STAT	MSK
TYPE	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	w1c	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Name	Description	Settings
Bits 31-7	**Reserved bits—** The upper 26 bits are not implemented and read as zeros.	N/A
DIS Bit 6	**CLK_32K disable—** This bit allows the user to turn off the external 32 kHz clock source for further power savings.	1 = disable external CLK_32K clock 0 = enable external CLK_32K clock
SRC[3:0] Bits 5-2	**RTR Interrupt source—** Selects one interrupt source from RTR[15] in RTR_MSB and RTR[14:0] in RTR_LSB. Bit RTR[0] provides the fastest available interrupt rate of 30.5 micro-seconds.	1111 = source is 0-to-1 transition on **RTR[15]** in RTR_MSB 0000 = source is 0-to-1 transition on **RTR[0]** in RTR_LSB
STAT Bit 1	**RTR Interrupt status— -** Sets on the first 0-to-1 transition of the selected interrupt source.	1 = a 0-to-1 transition has occurred 0 = a 0-to-1 transition has not occurred
MSK Bit 0	**RTR Interrupt mask—** Controls whether the RTR_IRQ_B interrupt is masked.	1 = masked 0 = unmasked

NOTE:

When the DIS bit is set, the RTR counter value is not valid until after the next positive edge of the 32 kHz clock.

Chapter 57
Real Time Clock (RTC)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	05/07/03		Updated for LTE specification release.

57.1 Introduction

The real time clock (RTC) module consists of several counters and registers used to maintain the day and time of day. The RTC operates even when the phone is powered down. An RTC alarm function is used to turn on the phone and alert the processor. If the phone is already on, the processor is interrupted. The module also contains a power-cut timer that monitors the duration of any temporary battery disconnection and manages the data back-up register bank and wake-up signal accordingly.

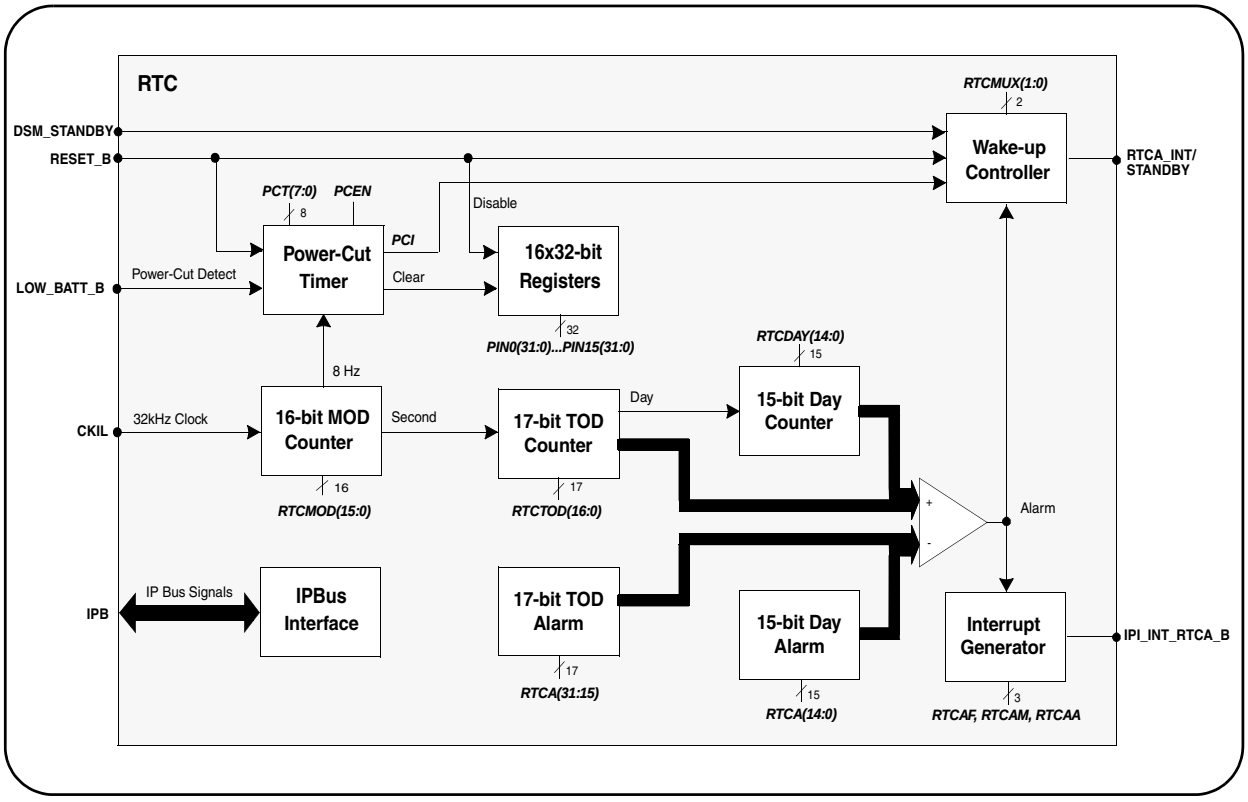


Figure 57-1. RTC Block Diagram

57.2 RTC Pin Description

In addition to the pins described below, this module uses a dedicated power pin, lvdd, which is powered even when other VDD pins are powered down. See Table 57-1 for a complete list of pins for the RTC module.

57.2.1 Input Pins

The RTC module is driven by a 32.768kHz clock signal on the CKIL pin. This is generated by a crystal oscillator on the Seaweed.

Although the RESET_B signal does not reset any registers or counters in the RTC, it is used to select between the RTCA_INT and STANDBY signals for the source of the RTCA_INT/STANDBY pin.

The DSM_STANDBY signal is an input from the Deep Sleep Module (DSM).

LOW_BATT_B is an input from the Seaweed that signals the detection of an under voltage at the start of a power cut.

57.2.2 Output Pins

The Real Time Clock Alarm function generates an interrupt request RTCA_INT which is output on IPI_INT_RTCA_B. It is used to interrupt the MCU when the phone is on.

The RTCA_INT/STANDBY pin may output the RTCA_INT or the STANDBY signals, to power-up or power-down the Neptune regulators on the Seaweed, respectively.

57.2.3 IP Bus

All registers in the RTC module are addressable for read and write operations by the MCU, through the IP Bus interface. This module supports 16-bit and 32-bit accesses conforming to the SRS version 2.0 IP bus specifications.

Table 57-1. RTC Pin List

Pin Name	Direction	Description
reset_b	Input	RESET_B (active low)
ipg_clk	Input	IP Bus Green
ipi_int_rtca_b	Output	IP Bus Indigo (active low)
ips_wdata[31:0]	Input	IP Bus Skyblue
ips_rdata[31:0]	Output	“
ips_addr[31:0]	Input	“
ips_module_en	Input	“
ips_byte_7_0	Input	“
ips_byte_15_8	Input	“
ips_byte_23_16	Input	“

Table 57-1. RTC Pin List (Continued)

Pin Name	Direction	Description
ips_byte_31_24	Input	“
ips_rwb	Input	“
ips_xfr_err	Output	“
scan_mode	Input	Test Interface
scan_enable	Input	“
test_reset	Input	“
scan_rtc_div_in	Input	“
tcm_scan_divergence	Input	“
scan_rtc_sdi_x[5:1]	Input	“
scan_rtc_sdo_x[5:1]	Output	“
rtc_sdo_1...5_bypass	Input	“
ckil	Input	32kHz Clock from Seaweed
dsm_standby	Input	STANDBY from DSM
rtca_int/standby	Output	Wake-up/Sleep signal
low_batt_b	Input	LOW_BATT* from Seaweed (active low)
lvdd	pwr	RTC Power
l_vss	gnd	RTC ground

57.3 RTC Programming Model

The RTC module has nineteen 32-bit registers: a status/control register (RTCSCR), a time register (RTCT), an alarm register (RTCA) and sixteen 32-bit registers (PCRAM0..15) designated as data back-up RAM. Only 32-bit read and write accesses are allowed for all registers in this module. In addition, the RTCSCR fields may be written by word access (RTCMOD) and byte access (PTC and control bits field only).

57.3.1 Register Summary

Figure 0-16.

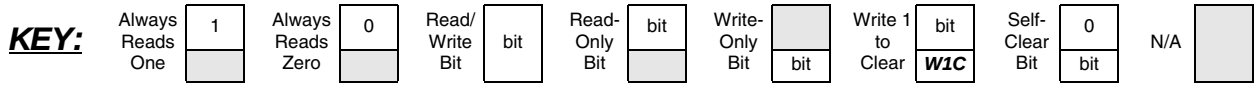


Table 57-2. RTC Register Summary
Figure 0-17.

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCSCR $2484_3000	R	RTCMOD[15:0]																PCT[7:0]								PCEN	PCI		RTCAF		RTCAM	RTCMUX[1:0]	
	W																													RTCAA			
RTCT $2484_3004	R	RTCDAY[14:0]																RTCTOD[16:0]															
	W																																
RTCA $2484_3008	R	RTCA[31:0]																															
	W																																
PCRAM0 $2484_300C	R	PCRAM0[31:0]																															
	W																																
...	R	...																															
	W																																
PCRAM15 $2484_3048	R	PCRAM15[31:0]																															
	W																																

57.3.2 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various RTC registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - — **r**: read only. Writing this bit has no effect.w: write only.
  - — rw: Standard read/write bit. Only software can change bit's value (other than hardware reset).
  - — rwm: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - — w1c: A status bit that can be read, and it cleared by writing a logic 1.
  - — slfclr: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- RESET: Gives the reset value of the bit. Possible values:
  - — 0: Will reset to a logic 0.
  - — 1: Will reset to a logic 1.
  - — ?: The reset state is unknown.

— u: Unaffected by reset.

RTCSCR **RTCSCR Register** **Addr $2484_3000**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	RTCMOD[15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PCT[7:0]								PCEN	PCI		RTCAF	RTCAA	RTCAM	RTCMUX[1:0]	
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	w	rw	rw	rw

Table 57-3. RTCSCR Description

Name	Description	Settings
RTCMOD [15:0] Bits 31-16	**RTC Modulus** — The RTCMOD register contains the modulus value at which the clock divider rolls over. Any inaccuracy in the crystal frequency may be compensated for by trimming this value.	See Appendix A for detailed trim calculations.
PCT[7:0] Bits 15-8	**Power Cut Time**r—These bits control the duration of the power cut timer, which can be programmed in 125ms increments from 0 to 31.875 seconds.	**PCT [7:0]** / **Duration (ms)** 00 / 0 01 / 125 02 / 250 ... / FF / 31,875
PCEN Bit 7	**Power Cut Enable**—The PCEN bit enables the power cut functionality on Neptune. Writing a 0 to this bit will also clear the backup RAM registers. This happens regardless of the current state of the PCEN bit.	PCEN = 0 → Power Cuts are disabled. Writing a "0" to this bit also clears the PCRAM registers. PCEN = 1 → Power Cuts are enabled.
PCI Bit 6	**Power Cut Interrupt**—The PCI bit indicates if a power cut has been detected. This bit is checked by the MCU during the power up sequence to determine if the power down was due to a temporary power cut or an extended one, which is considered as a power down. This bit is cleared when PCT[7:0] is written.	PCI = 0 → A power cut has not been detected. This would be the state of the bit during a normal power down, or if a power cut did occur, but the PC Timer expired before battery power was reapplied. PCI = 1 → A temporary power cut was detected.
Bit 5	**Reserved**	N/A
RTCAF Bit 4	**RTC Alarm Flag** — The RTCAF is a flag bit set by the alarm function. It asserts the interrupt line IPI_INT_RTCA_B to the AITC, if the RTCAM is cleared. This is a read-only bit. Writing it has no effect.	RTCAF = 0 → No RTC Alarm interrupt pending. RTCAF = 1 → RTC Alarm Interrupt pending.

Table 57-3. RTCSCR Description

<table>
<tr><td>RTCAA
Bit 3</td><td>RTC Alarm Acknowledge — The RTCAA bit is written to by the MCU as an acknowledgment of an interrupt. It clears the interrupt flag bit, RTCAF. This is a write-only bit. It always reads 0.</td><td>RTCAA = 0 → No interrupt acknowledgment.
RTCAA = 1→ Interrupt acknowledged, RTCAF cleared.</td></tr>
<tr><td>RTCAM
Bit 2</td><td>RTC Alarm Mask — The RTCAM is an interrupt mask bit to enable/disable the output of the RTCAF interrupt flag to the IPI_INT_RTCA_B pin.</td><td>RTCAM = 0 → RTCA_INT Interrupt enabled.
RTCAM = 1 → RTCA_INT Interrupt masked.</td></tr>
<tr><td>RTCMUX [1:0]
Bits 1-0</td><td>RTC Mux — The RTCMUX bits, in combination with the RESET_B signal, select the source of the signal on the RTCA_INT/STANDBY pin.</td><td>
<table>
<tr><th>RTCMUX[1:0]</th><th>MUX function</th></tr>
<tr><td>00</td><td>RTCA_INT</td></tr>
<tr><td>01</td><td>DSM_STANDBY</td></tr>
<tr><td>10</td><td>DSM_STANDBY if RESET_B=1,
RTCA_INT if RESET_B=0</td></tr>
<tr><td>11</td><td>Reserved</td></tr>
</table>
</td></tr>
</table>

RTCT RTCT Register **Addr $2484_3004**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	RTCDAY[14:0]															RTC TOD [16]
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

RTCT RTCT Register **Addr $2484_3006**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	RTCTOD[15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 57-4. RTCT Description

Name	Description	Settings
RTCDAY [14:0] Bits 31-17	**RTCDAY** — The RTCDAY register contains the count of days. It is incremented every time the RTCTOD counter rolls over.	N/A
RTCTOD [16:0] Bits 16-0	**RTCTOD** — The RTCTOD register contains the time-of-day, in seconds. It is incremented every time the RTC modulus counter rolls over. It counts from 0 to 86,399 and rolls over.	N/A

RTCA RTCA Register **Addr $2484_3008**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	RTCA[31:16]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	RTCA[15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 57-5. RTCA Description

Name	Description	Settings
RTCA [31:0] Bits 31-0	**RTCA** — The RTCA register contains an alarm value to be compared with the RTCT contents. The interrupt flag RTCAF is asserted when: RTCDAY[14:0]:RTCTOD[16:0] = RTCA[31:0].	N/A

57.4 RTC Operation

57.4.1 RTC Prescaler

A 16 bit modulo divider is used to divide the 32.768kHz external clock signal down to 1Hz. The modulo divider is used to enable trimming of the RTC frequency. This allows the use of a very inexpensive crystal. A ripple counter decrements from the 16-bit modulo register (RTCMOD) value down to zero, then reloads the modulo value and restarts again. A new value written into the modulo register will be loaded into the prescaler at the end of the cycle. The output of the modulo divider representing one second reloads the counter. See Appendix A for a detailed explanation of the trim value calculations.

57.4.2 RTC Counters

The 1Hz clock generated by the RTC Prescaler drives a 17-bit counter (RTCTOD) that increments the number of seconds in a 24 hour period. This counter increments from 0 to 86,399 then rolls over to zero and counts up again. The RTCTOD counter drives a 15-bit free running counter (RTCDAY) which is used for the day count. The day counter increments up to 32,767.

The RTCTOD[16:0] and RTCDAY[14:0] read and write registers share a common 32-bit address (RTCT). RTCTOD[0] is the least significant bit of the seconds count and is bit 0 of the 32-bit register. RTCDAY[0] is the least significant bit of the day count and is bit 17 of the 32-bit register. RTCDAY[14] is the most significant bit of the day count and is bit 31 of the 32-bit register. Data coherency between RTCDAY and RTCTOD is therefore guaranteed by using 32-bit accesses to the RTCT register.

57.4.3 RTC Alarm & Interrupt Generation

The RTCA register is used to set the alarm time (in seconds from 0 to 86,399). When the RTCDAY & RTCTOD counter combination is equal to the value in RTCA, the interrupt flag (RTCAF) is set in the RTCSCR:

$$\text{RTCDAY[14:0]:RTCTOD[16:0]} = \text{RTCA[31:0]} \qquad \textit{Eqn. 57-1}$$

The RTCAF bit generates an interrupt signal (RTCA_INT) output on the IPI_INT_RTCA_B pin, if the interrupt mask bit (RTCAM) in the RTCSCR is cleared. The flag bit is cleared by the MCU by writing to the interrupt acknowledge bit (RTCAA), also in the RTCSCR.

In addition to the RTCA_INT signal that always drives the AITC to generate MCU interrupts, this module also has a pin that drives out either the RTCA_INT, or the STANDBY signals. The STANDBY signal is generated by the DSM. The selection of the signal used depends on the RESET_B signal, as well as the RTCMUX[1:0] bits in the RTCSCR, as shown in Table 57-6.

NOTE:

As RESET_B does not affect the RTC, the initial state of the RTCA and the RTCAM mask bit, along with the other bits in the RTCSCR, is indeterminate. To prevent unintended interrupts from the RTCA module, the MCU must initialize the state of the RTC registers before enabling the RTCA interrupt in the AITC. Seaweed should also ignore the RTCA_INT/STANDBY pin when it has disabled the RTC regulator. This would occur during a low battery condition or power cut to the RTC.

Table 57-6. RTCA_INT/STANDBY output signal

RTCMUX[1:0]	MUX function
00	RTCA_INT
01	DSM_STANDBY
10	DSM_STANDBY if RESET_B=1, RTCA_INT if RESET_B=0
11	Reserved

57.5 Power Cut Timer

The PC Timer is an 8-bit writable and readable counter used to set the duration of the power cut time-out. The timer can be set to a duration of 0 to 31.875 seconds, in 125 ms increments. A power cut is detected, when the LOW_BATT_B and RESET_B signals are asserted. The PC timer counter is enabled and asynchronously decremented by one count on this event. Access to the backup PCRAM register bank is disabled when RESET_B is asserted.

During the power cut, the counter will be decremented by the 8 Hz clock from the RTC Prescaler. Since the event detection cannot be synchronized to the 8 Hz clock, the initial decrement may occur immediately after a clock edge, so that the count-down duration can be off by up to one count relative to the programmed value. For example, if the register is programmed for 3.0 seconds, the PC Timer will decrement to zero within 2.875 to 3.0 seconds.

The PC Timer will stop the countdown on the rising edge of the LOW_BATT_B signal, if it has not yet expired. The PCI bit in the RTCSCR is set on the next 8 Hz clock edge, to indicate that a temporary power-cut was detected and access to the PCRAM register bank is re-enabled. The PCI bit also drives a signal to the RTCA_INT/STANDBY pin which wakes up the Seaweed. If the rising edge of the LOW_BATT_B signal occurs after the PC Timer count-down expires, the PCI bit is not set and the PCRAM registers are cleared.

NOTE:

If implemented, the user RESET button must be pressed for a period longer than the time-out programmed in the PC Timer for the PCRAM to be cleared and a clean power up sequence to be initiated after its release.

The PC Timer value is uninitialized on RTC power up and the software must set this and the power cut enable bit (PCEN) in the RTCSCR. The power cut feature may be disabled by clearing the PCEN bit, in which case the PCRAM registers are cleared.

57.5.1 PCRAM Backup Register Bank

A RAM of 512-bits, implemented as a bank of 16 32-bit registers, is included in the RTC module for backup of critical data during a temporary power cut. Access to the PCRAM is disabled by the RESET_B signal and the contents are cleared if the PC Timer countdown expires. The contents are also cleared during a normal power down sequence by resetting the PCEN.

57.5.2 Signal Interface

A potential problem exists at the interface of the RTC module, which is normally always powered, and other modules, which may be powered down. To avoid spurious logic levels corrupting data on the RTC module and to eliminate the risk of undesirable leakage current across the interface, all susceptible signals are gated by RESET_B. This signal is asserted by the Seaweed to start the Neptune power down sequence. When the Neptune powers up, RESET_B is negated enabling the signal interface once again.

57.5.3 Real Time Clock Trim Calculations

It is desired that a very inexpensive and inaccurate crystal be used for the 32.768kHz reference. This has a target accuracy of +/-1%. This crystal will cause the Real Time Clock to drift by as much as 100 minutes over the course of 1 week:

86400s/Day x 7Days/Week x 1% = 6048s/Week ***Eqn. 0-1***

57.5.3.1 Obtainable Accuracy of Trim

If we allow the prescaler which divides the nominal 32.768kHz clock down to 1Hz to be a modulo divider, then we will have a nominal resolution equal to:

1Hz/32.768kHz = .003%, or 30ppm ***Eqn. 0-2***

If 30ppm resolution can give us +/-30ppm (+/-1 bits) accuracy then we should be able to keep time to within one quarter minute per week:

86400s/Day x 7Days/Week x 30ppm = 18s/Week ***Eqn. 0-3***

The 26MHz crystal to be used in the system has an accuracy around +/-30ppm, so if say for worst case the clock has an accuracy of +/-60ppm, then we would add this error to the error of the trim:

+/-30ppm (from trim function) + +/-60ppm (from reference) = +/-90ppm ***Eqn. 0-4***

With a 1Hz clock that is accurate to +/-90ppm we should be able to keep time to within 1 minute per week:

86400s/Day x 7Days/Week x 90ppm = 54s/Week ***Eqn. 0-5***

Further improvement could be obtained by trimming from the clock that is compensated to the GSM system clock.

57.5.3.2 Obtaining the Trim Value

In order for the trimmable RTC to do any good, we must be able to calculate the error in the 32.768kHz clock. The most obvious way is to compare it to the much more accurate 13MHz clock.

Because this is a GSM phone, there will be a Deep Sleep Module (DSM) available. This has a function by which the 13MHz CLIH is compared to the 32.768kHz CLIL. This is accomplished by counting the number of CLIH periods within a programmable number of CLIL periods. The REFCOUNT register provides this value.

So, we can simply set up the DSM to calculate the number of CLIH periods in a single CLIL period. Therefore:

REFCOUNT = CLIH/CLIL ***Eqn. 0-6***

The value which we want to place in the RTC modulo register is the number of 32.768kHz (nominal) CLIL periods in on second or:

RTCMOD = CLIL/1Hz ***Eqn. 0-7***

Since we know the frequency of CLIL very well, we can call it CLIHDes and store it as a constant. This would be 13MHz, for example. Now we can represent 1Hz as:

1Hz = CLIH/CLIHDes ***Eqn. 0-8***

Substituting, we can now arrive at RTCMOD:

RTCMOD = CLIHDes/REFCOUNT ***Eqn. 0-9***

This is the correct number to use if we had infinite resolution. However, the process of counting CLIH periods introduces reduces resolution that we must manage. We must ensure that the measurement resolution does not significantly degrade the trim. Let's assume that we want the measurement to be at 10x the resolution of the trim. So we have a 15 bit divider for the trim. We need 4 more bits of measurement, which will give us 16x more resolution for measurement than trim.

If we plug actual numbers into equation A-6, we see that there are only about 400 CLIH counts per CLIL count:

REFCOUNT = 13MHz/32.768kHz = 396.7... ***Eqn. 0-10***

By itself, this only represents about 9 bits of resolution (0x1,FF = 511). So we need to get this number to be larger than 2^19. (15 bits + 4 bits). So if we simply count for many cycles of CLIL instead of a single cycle, we can get greater resolution. This is a similar method as a dual slope ADC. More time means more resolution. To obtain 19 bits of resolution if each CLIL counts to 396, we should count 1324 CLIL cycles:

2^19/396 = 1324 ***Eqn. 0-11***

So the MEASTIME register would be set to the current COUNT32 value plus 1324 so that the REFCOUNT contains the number of CLIH periods in 1324 CLIL periods.

Because of this extra time, the value placed in RTCMOD will be changed. This will multiplied by 1324:

RTCMOD = CLIHDes x 1324 / REFCOUNT ***Eqn. 0-12***

Where:

- RTCMOD = value to be stored in RTC modulo register
- CLIHDes = constant describing speed of high frequency clock
- 1324= number of CLIL periods over which CLIH is accumulated
- REFCOUNT = number of CLIH periods counted in 1324 CLIL periods

Chapter 58
Keypad Port (KPP)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	03/19/02	Rajiv Mittal	Changed the Interface from MIG/PIG to IP Bus. Replaced the bidirectional pins (kpp_kpd) with two unidirectional pins. Removed the kpp_kddr_buf bus as it is no longer required.
0.2	05/21/02	Rajiv Mittal	Added description for xfr_wait signal assertion.
0.3	05/07/03		Updated for LTE specification release.

58.1 Introduction

The Keypad Port is a 16-bit peripheral which can be used either for keypad matrix scanning or as general purpose I/O. The block diagram of the KPP is given in Figure 58-1. Please see note in Section 58.2.2, “Output Pins,” regarding configuration of the GPIO module related to KPP functionality.

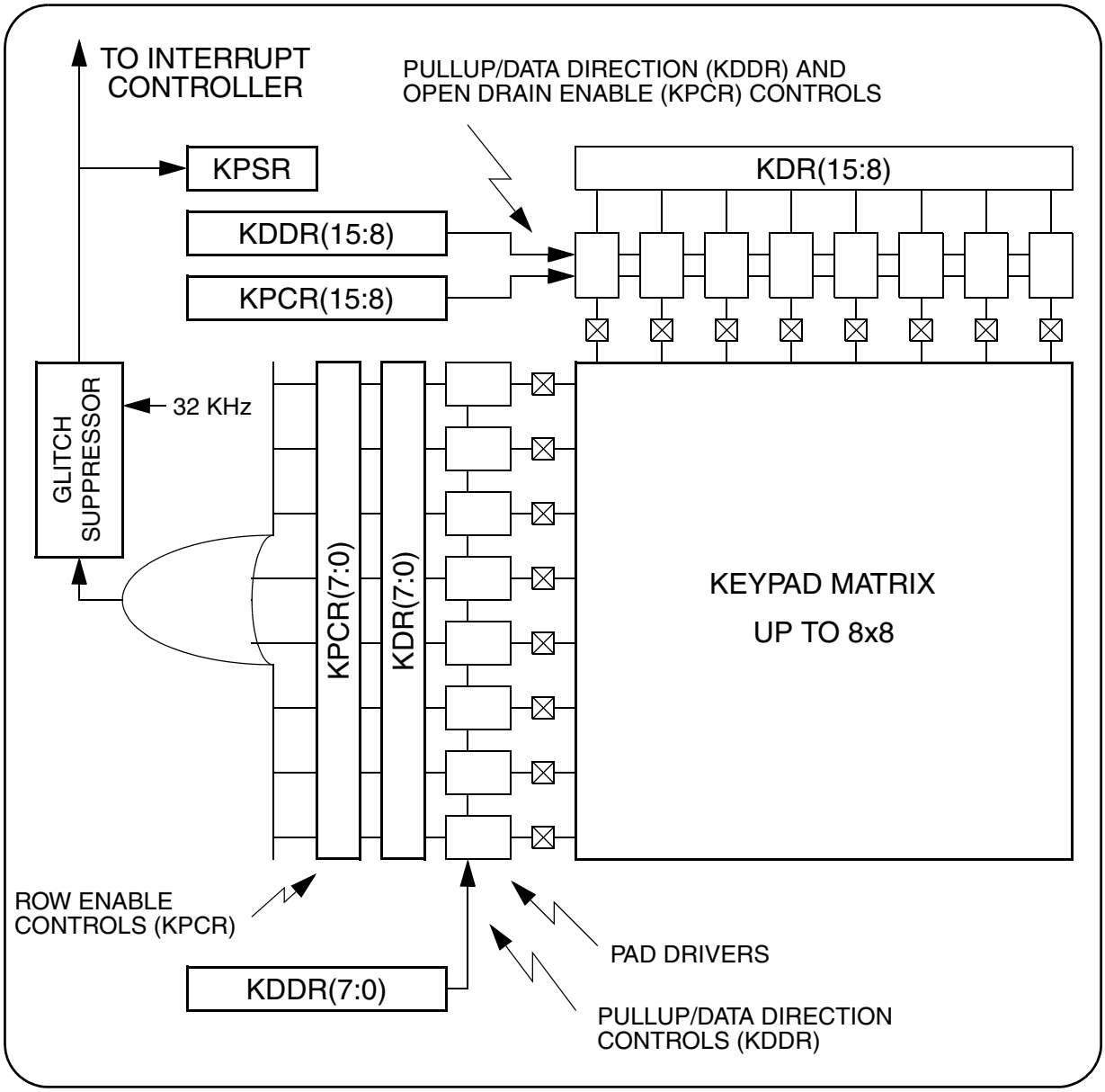


Figure 58-1. KPP Peripheral Block Diagram

58.2 Keypad Port Pin Description

Sixteen pins are dedicated to the KPP. Keypads of any configuration up to 8 rows and 8 columns are supported through software configuration of the peripheral pins. Any Column pin not used for the Keypad can be used as general purpose input/output and any Row pin not used for the Keypad can be used as general purpose input only.The registers are configured such that the Column pins can be treated as an I/O ports and Row pins can be treated as Input ports only.

58.2.1 Input Pins

All the 16 kpp_kpd_in pins associated with KPP are input ports.

58.2.2 Output Pins

KPP have 8 output pins (kpp_kpd_out) which reflect the upper 8 bits of KPDR register. Additionally, the 8 most significant bits (15 - 8) can be designated as open drain outputs by writing a one into the appropriate bits in KPCR.

Note : In Neptune LTE, ROWS are used as Input only and COLUMNS are Input/Output. Also only COLUMN0 and COLUMN1 are routed through GPIO to Neptune Chip boundary and other Columns(7-2) are not routed to Neptune Chip boundary.

58.2.3 KPP Module Pin List

Table 58-1 is a list of the KPP module pins.

Table 58-1. KPP Module Pin List

S. No.	Pin Name	Direction	Description
1	ips_rdata[15:0]	O	16 bit Read data bus from module
2	ips_xfr_wait	O	Wait signal
3	ips_xfr_err	O	Invalid access error signal
4	kpp_mint_b	O	Interrupt request at pin
5	kpp_kpcr_buf[15:8]	O	Control Register Data
6	kpp_kpd_out[15:8]	O	Kpp output data (written in KPDR register)
7	kpp_kddr_buf[15:8]	O	Keypad port direction signal to GPIO
8	ckil_clk	I	Buffered Low freq. clock (32 KHz)
9	ipg_clk	I	mcu peripheral clock (IP bus global clock)
10	ips_wdata[15:0]	I	16 bit write data bus to module
11	ips_module_en	I	Module enable
12	ips_addr[11:1]	I	Address bus
13	ips_rwb	I	1 - read, 0 - write

Table 58-1. KPP Module Pin List

S. No.	Pin Name	Direction	Description
14	ips_byte_15_8	I	Upper byte (15 down to 8) select
15	ips_byte_7_0	I	Lower byte (7 down to 0) select
16	kpp_kpd_in[15:0]	I	Kpp input data (on the boundary pins)
17	periph_soft_reset_b	I	MCU peripheral module reset
18	ipt_clk_se	I	scan enable
19	ipt_scan_mode	I	scan mode

58.2.4 KPP PIN DIAGRAMS

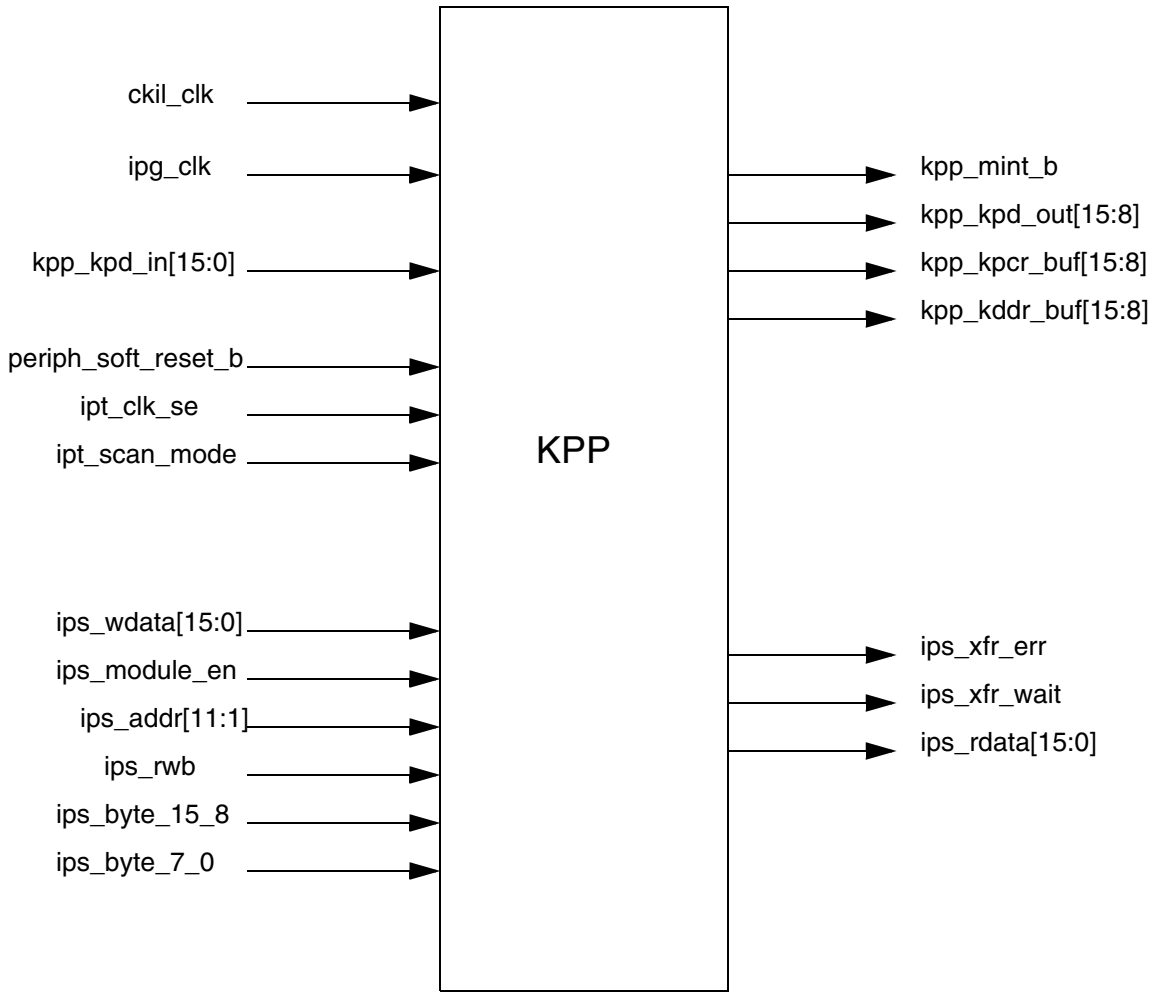


Figure 58-2.

58.3 Keypad Port Programming Model

58.3.1 Register Summary

The KPP module has four registers: a control register (KPCR), a status register (KPSR), a data direction register (KDDR) and a data register (KPDR).

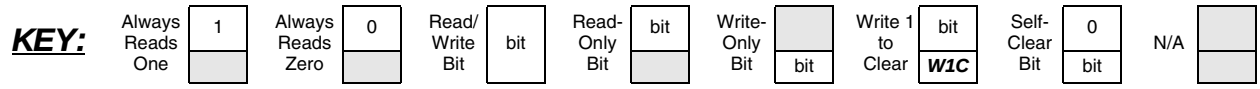


Table 58-2. Keypad Port Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KPCR ($2484_E000)	R	KCO[7:0]								KRE[7:0]							
	W																
KPSR ($2484_E002)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	KPKD
	W																W1C
KDDR ($2484_E004)	R	KCDD[7:0]								0	0	0	0	0	0	0	0
	W																
KPDR ($2484_E006)	R	KCD[7:0]								KRD[7:0]							
	W																

58.3.2 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various KPP registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

The Keypad Control Register determines which of the eight possible column strobes are to be open drain when configured as outputs and which of the eight row sense lines are considered in generating the interrupt to the MCU core. This register is byte addressable.

KPCR Keypad Control Register **Addr $2484_E000**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	KCO[7:0]								KRE[7:0]							
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 58-3. KPCR Description

Name	Description	Settings
KCO[7:0] Bits 15-8	**Keypad Column Strobe Open-Drain Enable**—Setting a column open-drain enable bit (KCO7 - KCO0) disables the P-channel driver on that pin. Clearing the bit allows the pin to drive to the high state. This bit has no effect when the pin is configured as an input.	KCOn = 0 → Column strobe output is totem-pole drive (P-Channel enabled). KCOn = 1 → Column strobe output is open drain (P-Channel disabled). This is only if the corresponding bit is defined as an output. **KDDR (15:8) / KPCR (15:8) / Pin Function** 0 / X / Input 1 / 0 / Totem-Pole Output 1 / 1 / Open-Drain Output
KRE[7:0] Bits 7-0	**Keypad Row Enable**	KREn = 0 → Row is not included in keypad key press detect. KREn = 1 → Row is included in keypad key press detect.

The Keypad Status Register reflects the state of the keypress detect circuit. This register is byte addressable.

KPSR Keypad Status Register **Addr $2484_E002**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
																KPKD
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	w1c
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 58-4. KPSR Description

Name	Description	Settings
Bits 15-1	**Reserved**	N/A
KPKD Bit 0	**Keypad Key Detect—** The keypad key detect (KPKD) status bit is set when one or more enabled rows are detected low. The status remains set until cleared by software. The bit will not be set again until the detect circuit senses no keys depressed followed by a key press. This action eliminates multiple interrupts for the same key press without additional software interaction. KPKD is cleared by first reading the bit set, followed by writing a logic one back into the bit.	KPKD = 0 → No key presses detected KPKD = 1 → A key has been depressed

The bits in the KDDR control the direction of Columns in GPIO and also the data read from KPDR register. This register is byte addressable.

KDDR Keypad Data Direction Register **Addr $2484_E004**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	KCDD[7:0]								KRDD[7:0]							
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 58-5. KDDR Description

Name	Description	Settings
KCDD [7:0] Bits 15-8	**Keypad Column Data Direction —** The upper eight bits in the register affect the pins designated as column strobes. Setting any bit in this register configures the corresponding pin as output in GPIO and clearing the bit configures the same as input. They also control the data read to KPDR register. Setting any bit in this register reads the corresponding bit in KPDR register. Clearing any bit in this register reads the corresponding input port pin (kpp_kpd_in). This register is cleared by reset.	KCDDn = 0 → Column will be configured as input in GPIO and data will be read from the corresponding input port pin.(n=0...7) KCDDn = 1 → Column will be configured as output and data will be read from the corresponding KPDR register.
KRDD [7:0] Bit 7-0	**Keypad Row Data Direction —** The lower eight bits affect the row sense pins.All the Rows are configured as Input only so these bits are always read as 0. Any read to KPDR register will read the corresponding input port pin (kpp_kpd_in).	KRDDn = 0 → Reading KPDR register will read the input port pins (kpp_kpd_in[7:0]).

This 16-bit register is used to access the column and row data. Data written to this register is stored in the internal latch, and if any pin is configured as an output the data is driven onto the pin. This register is not initialized by reset. Valid data should be written to this register before any bits are configured as outputs. This register is byte addressable.

KPDR Keypad Data Register **Addr $2484_E006**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	KCD[7:0]								KRD[7:0]							
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r
RESET:	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 58-6. KPDR Description

Name	Description	Settings
KCD[7:0] Bits 15-8	**Keypad Column Data —** A read of these bits returns the value on the input port pin if the corresponding bit in KDDR register is cleared. Otherwise, if the corresponding bit in KDDR register is set, then the value read is the value stored in the register.	N/A
KRD[7:0] Bits 7-0	**Keypad Row Data —** A read of these bits returns the value on the input port pins.	N/A

NOTE : As kpp_kpd_in[15:0] data bus is aynchronous signal coming from Neptune chip boundary, 2 flop synchronizer is used to synchronize the data with ARM clock (ipg_clk). So this data reaches the ips_rdata bus delayed by 2 ipg_clk cycles. Therefore whenever an access is done to KPDR register for reading the data, it generates a (2 ipg clock cycle) transfer wait signal for the mcu. The assertion of xfr_wait results in two additional clocks cycle wait before it finally reads the data, which is the mirror image of the port signals.

58.4 Keypad Operation

The keypad port is designed to ease the software burden of scanning a keypad matrix. With appropriate software support, the KPP is capable of detecting, debouncing and decoding one or two keys pressed simultaneously in the keypad.

Logic in the KPP is capable of detecting a key press even while the processor is in one of the low power standby modes. The KPP generates a CPU interrupt any time a key press is detected. This interrupt is capable of taking the processor out of the low power mode.

58.4.1 Keypad Matrix Construction

The keypad port is designed to interface to a keypad matrix which shorts the intersecting row and column lines together whenever a key is depressed.

58.4.2 Keypad Port Configuration

Software must initialize the Keypad Port for the size of the keypad matrix. Pins connected to the keypad columns should be configured as open-drain outputs. Pins connected to the keypad rows should be configured as inputs. On-chip pullup resistors are implemented for active keypad rows, as defined in the pins section of the spec.

The row inputs must also be enabled in the Keypad control register for that input to be active in the interrupt generation circuit.

Discrete switches which are not part of the matrix may be connected to any unused row inputs. The second terminal of the discrete switch is connected to ground. The hardware will detect closures of these switches without the need for software polling.

Please see note in Section 58.2.2 on page 58-3.

58.4.3 Keypad Matrix Scanning

Keypad scanning is simply a software loop which walks a zero across each of the keypad columns, reading the value on the rows at each step. The process is repeated several times in succession, the results of each pass compared with those from the previous pass. When several (3 or 4) consecutive scans yield the same key closures a valid key press has been detected. Software then can decode exactly which switch was depressed and pass the value up to the next higher software layer.

The basic debouncing period which has to be defined in the software routine (the basic period is the period between the scan of two consecutive columns, so the debounce time between two consecutive scans of the whole matrix shall be the number of columns multiplied by the basic period) is controlled by the MCU with his internal timers.

58.4.4 Keypad Standby

There is no need for the CPU to continually scan the keypad. Between key presses, the keypad can be left in a state which requires no software intervention until the next key press is detected. To place the keypad in a standby state, software should write all column outputs low. Row inputs are left enabled. At this point the CPU can attend to other tasks or revert to a low power standby mode. The Keypad Port will interrupt the CPU if any key is pressed.

Upon receiving a keypad interrupt, the CPU should set all the column strobes high, and begin a normal keypad scanning routine to determine which key was pressed. It is important that open-drain drivers be used to driving any of the columns high to prevent a possible DC path between power and ground through 2 or more switches.

58.4.5 Glitch Suppression on Keypad Inputs

A glitch suppression circuit qualifies the keypad inputs to prevent noise from inadvertently interrupting the CPU. The circuit is shown in Figure 58-3. It is a 4 state synchronizer clocked off the low frequency clock. The low frequency clock must continue to run in any low power mode that the keypad is a wakeup source, as the CPU interrupt is generated off the synchronized input. An interrupt is not generated until all 4 synchronizer stages have latched a valid key assertion, effectively filtering out any noise less than 91.5 us (3 32kHz clock periods) in duration. The interrupt output is latched in an S-R latch and remains asserted until cleared by software.

After scanning the keypad, it is necessary to wait for at least 153 us to allow the synchronizer to stabilize prior to clearing the interrupt status register.

58.4.6 Multiple Key Closures

One or two keys pressed simultaneously are easily detected by the software. When three or more keys are pressed, however, it is possible that errant key closures may be detected. As can be seen in Figure 58-4, three keys pressed simultaneously can short between the column currently “scanned” by the MCU and another column. Depending on the location of the third key pressed, a “ghost” key press may be detected.

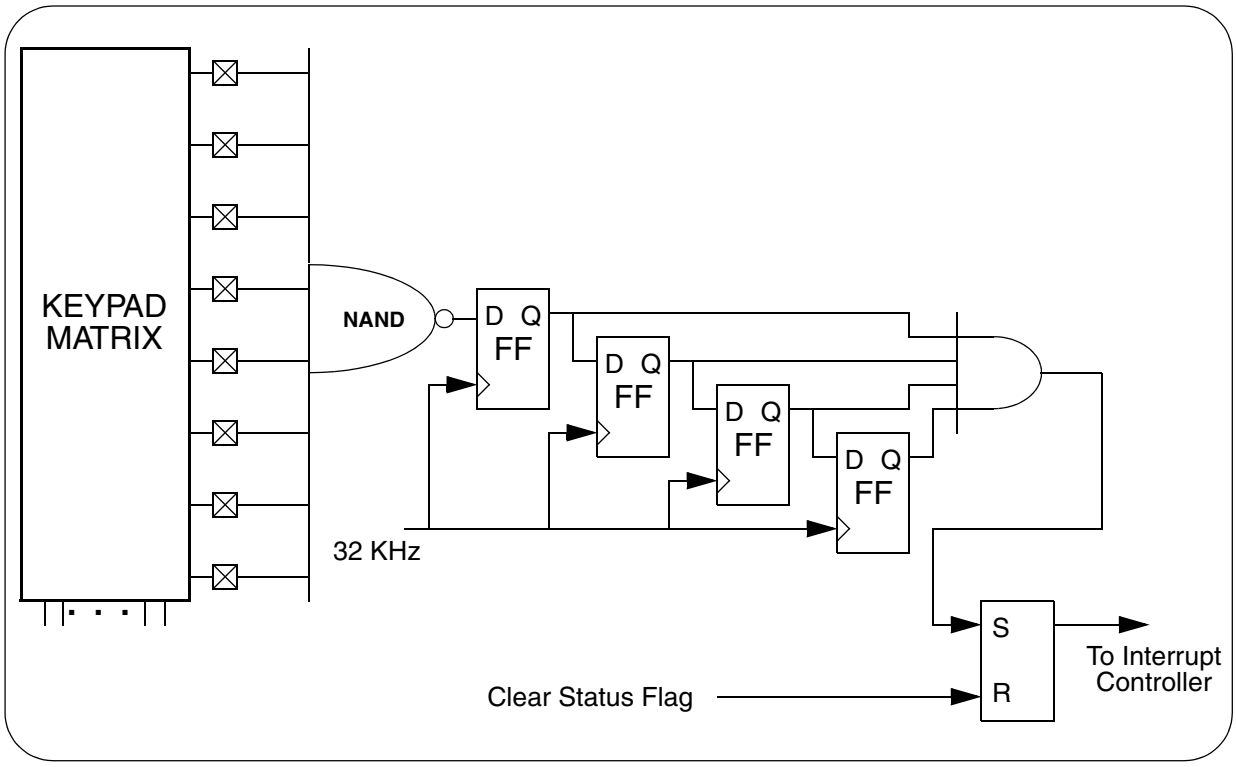


Figure 58-3. Keypress Synchronizer Functional Diagram

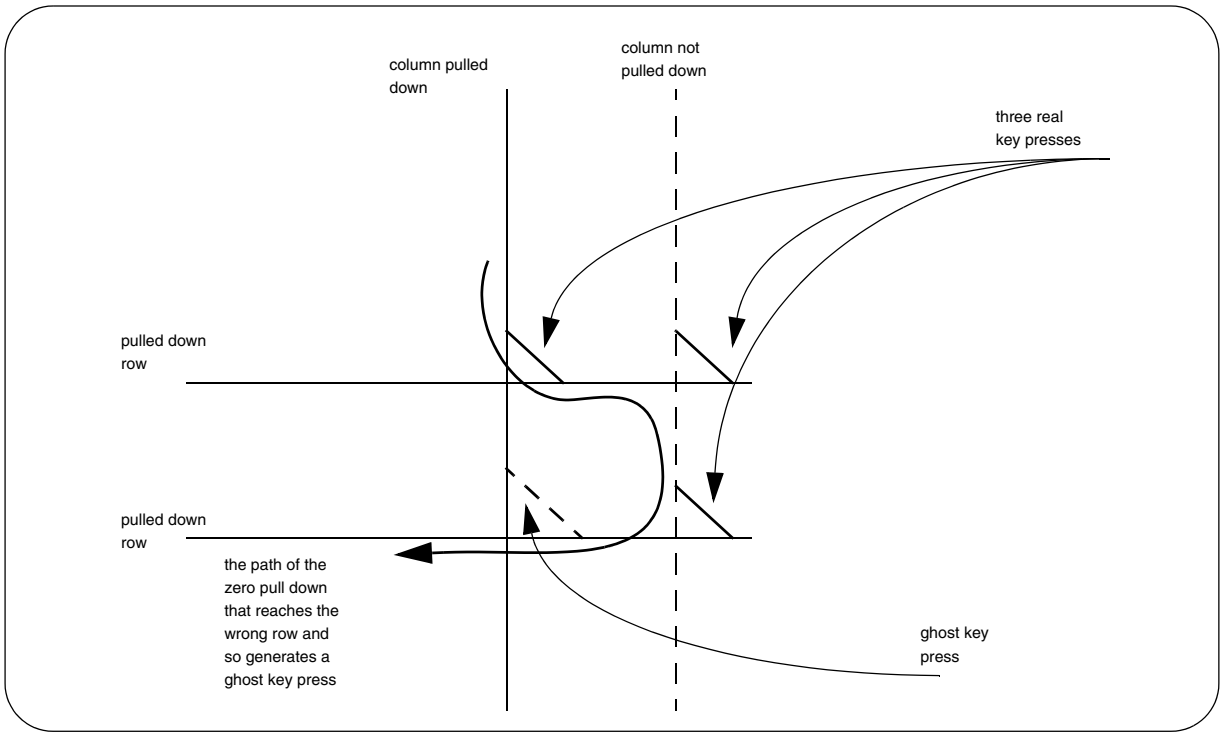


Figure 58-4. Decoding Wrong Three Key presses

58.4.7 Typical Keypad Configuration and Scanning Sequence

Configure Keypad

1. Enable number of rows in keypad (KPCR[7:0])
2. Configure keypad columns as open drain (KPCR[15:8])
3. Write 0's to KPDR[15:8]
4. Configure columns as output(KDDR[15:8])

(now in standby mode, awaiting a keypress...)

Keypress Interrupt Detected

Begin Keypad Scanning Routine

1. Disable keypad interrupt (optional)
2. Write a single column to 0, others to 1. Wait appropriate time for rows to rise due to RC delay.
3. Sample row inputs and save data. Multiple key presses can be detected on a single column.
4. Repeat steps 2 & 3 for remaining columns.
5. Return all columns to 0 in preparation for standby mode.
6. Wait at least 153 us for internal synchronizer to stabilize from prior scanning steps.
7. Clear interrupt status bit by reading KPSR set, and then writing it to a 1. Note: the status bit **must** be re-read during this step to ensure that the clearing process occurs, even if it has been previously read sometime during the routine.
8. Re-enable keypad interrupt.

Chapter 59
Display Memory Access Controller (DMAC)

Revision History Table

Revision	Date	Author	Changes
0.0	12/13/01	Rob Greenwood, Shannon Osgood	Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
1.0	2/25/02	Mindy Hsu	Changed from PIG interface to IP (Sky-Blue line) bus interface
2.0	4/18/02	Mindy Hsu	update scan signal names
3.0	4/29/02	Mindy Hsu	update port list for byte enable
4.0	6/19/02	Mindy Hsu	clarify ckih input clock
5.0	07/23/02	Mindy Hsu	Updated per DDTS# DSPhl15022.5.1
5.1	05/07/03		Updated for LTE specification release.

59.1 Overview

The Display Memory Access Controller module transfers data from the display memory buffer to the external display device. Direct Memory Access (DMA) transfers the data transparently with minimal software intervention. Bus utilization of the DMA is controllable and deterministic.

As cellular phone displays become larger and more colorful, demands on the processor increase. More CPU power is needed to render and manage the image. The role of the display controller is to reduce the CPU's involvement in the transfer of data from memory to the display device so the CPU can concentrate on image rendering. Direct Memory Access (DMA) is used to optimize the transfer. Embedded control information needed by the display device is automatically read from a second buffer in system memory and inserted into the data stream at the proper time to completely eliminate the CPU's role in the transfer.

A typical scenario for a cellular phone display is to have the display image rendered in main system memory. Once the image is complete, the CPU triggers the DMAC module to transfer the image to the display device. Image transfer is accomplished by burst DMA which steals bus cycles from the CPU. Cycle stealing behavior is programmable so bus utilization is kept within predefined bounds. Once the transfer is complete, a maskable interrupt is generated indicating the status. For animated displays it is suggested that a two-buffer ping-pong scheme be implemented so that the DMA is fetching data from one buffer while the next image is rendered into the other.

Several display sizes and types will be used in the various products that utilize the DMAC. The DMAC module has the capability of directly interfacing to the selected display devices. Both serial and parallel interfaces are supported. The DMAC module only supports writes to the display controller. ***DMAC read operations from the display controller are not supported!***

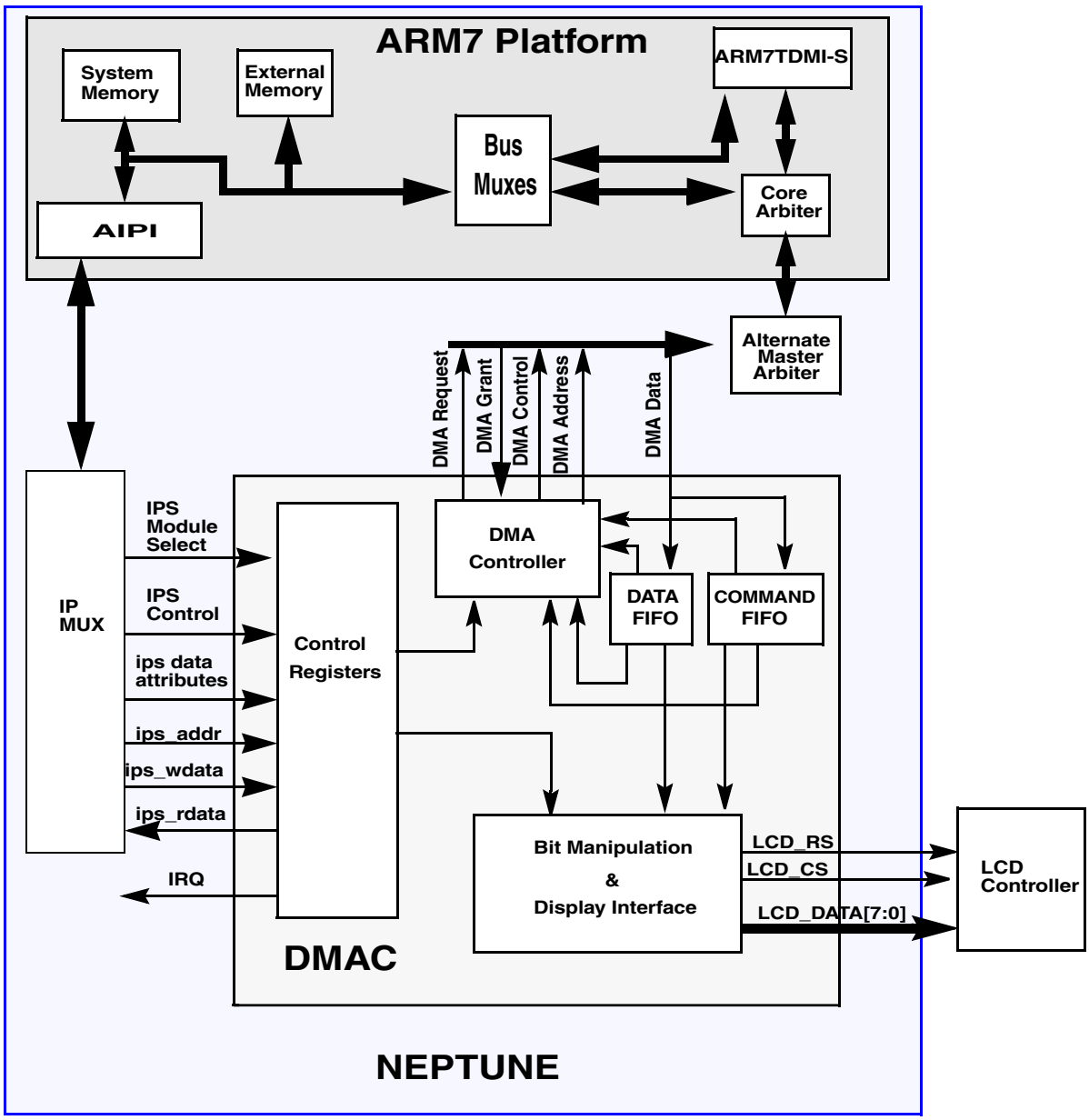


Figure 59-1. DMAC System Diagram

59.2 DMAC Module Pin List

Table 59-1 is a list of the DMAC module pins.

Table 59-1. DMAC Module Pin List

Pin Name	Direction	Description
AMBA Interface Signals		
dmac_haddr[31:0]	Output	R-AHB address bus
dmac_hwrite	Output	R-AHB read/write. 1 => Write; 0 => Read (tied to 0 by DMAC)
dmac_hsize[1:0]	Output	R-AHB transfer size (tied to 10 by DMAC for 32 bit reads)
dmac_htrans[1:0]	Output	R-AHB transfer type
dmac_hprot[1:0]	Output	R-AHB protection encoding (dmac_hprot[0] tied to 1 by DMAC)
dmac_br	Output	Bus request output to AMARB
amarb_bg	Input	Bus grant from AMARB
hrdata[31:0]	Input	R-AHB read data bus
hresp0	Input	Transfer response encoding (connected to HRESP[0] on R-AHB bus)
hready	Input	Transfer acknowledge
IP Bus Interface Signals		
ips_wdata[15:0]	Input	write data bus
ips_addr	Input	address bus
ips_module_en	input	module select
ips_byte_15_8	input	byte select bits 8 to 15
ips_byte_7_0	input	byte select bits 0 to 7
ips_rwb	input	read/write enable (asserted->read, negated->write)
ips_supervisor_access	input	supervisor mode access
dmac_ips_xfr_wait	output	transfer wait, transfer cannot be completed in the current cycle
dmac_ips_xfr_err	output	transfer error, an illegal access is attempted
dmac_ips_rdata[15:0]	output	read data bus
System Signals		
dmac_irq_b	output	Active low DMAC Interrupt (mod_irq_b[x])
ipt_scan_mode	input	Module scan mode
ipt_clk_se	input	Scan enable
mcu_clk	input	mcu clock
reset_b	input	Active LOW system Reset (m_rst_b)
ckih[1]	input	clock input to the module
LCD Interfaces		
dmac_lcd_data[7:0]	output	Data bus used to write information to external LCD controller

Table 59-1. DMAC Module Pin List (Continued)

Pin Name	Direction	Description
dmac_lcd_cs	output	Used as a chip select for the external display controller in serial mode. In parallel mode, used as write strobe for the external display controller. The polarity of this signal is programmable.
dmac_lcd_rs	output	LCD Register Select signal that indicates to the LCD device whether the data being written is display data or control data. The polarity of this signal is programmable.

1. ckih is the clock used to calculate and provide clock signals to the external LCD controller. It is tied to mcu_clk on Neptune LTE. The maximum speed provided to LCD controller is about half of the mcu_clk speed. It is running in wait mode and high in stop and doze mode.

59.3 Functional Description

The purpose of the DMAC is to transparently and efficiently transfer image data from system memory to an external LCD controller. The system memory can be either internal or external memory. Figure 59-1 on page 59-2 shows a block diagram of the DMAC module within the system. The DMAC module contains a DMA controller that is designed to operate as an alternate master on the reduced ARM high performance bus (R-AHB) specified for Neptune. The data interface to the R-AHB is 32 bits wide and operates as described in the AHB arbitration section of the AMBA specification. The DMAC DMA interface requests control of the AHB in order to do 32-bit reads from system memory.

The purpose of the DMA within the DMAC is to transfer image and control data from system memory to the DMAC FIFO where it is formatted and sent out to the external LCD controller. The DMA only performs read accesses from system memory. The data is collected in 32-bit words and stored in two internal FIFOs. Data is pulled from the FIFOs, as needed, and put in the correct format for the external LCD controller.

The DMAC has both control and status registers which are accessible via the IP bus. These registers are used to store information about the type of LCD controller attached to the system.

The DMAC can be configured to write image data to an external LCD controller via a 4- line serial, 3-line serial, or an 8-bit parallel interface.

During automatic transfers, the DMAC is responsible for properly sequencing display data and control strings in a data stream that is sent to the LCD controller to fill its internal display RAM. Display data is typically written to the external controller in pages. Each write to the controller fills one column of the current page. The DMAC assumes that the controller automatically increments its display RAM column pointer after each write.

NOTE:

Changing the DMAC configuration in the middle of a transfer can cause corruption of the data stream output to the LCD. The DMAC configuration should only be changed when the GO bit and BUSY bit of the DMAC control/status register are cleared.

59.3.1 Accessing the LCD Controller

The DMAC provides two methods of accessing the external display device. The first, and preferred method, is through automatic writes handled by the DMAC. The second, is through direct IP register access via the LCD Write Data register.

59.3.1.1 Automatic DMAC Transfers

The DMAC is designed to take a block of data from system memory and write it, without software intervention, to an external display controller. This is done by giving the DMAC a starting address and a data block size (in bytes). This method of data transfer can be used for both LCD display data as well as command data. Transfers are initiated by setting the GO bit in the DMAC Control/Status register. The data transfer is done in the background. Upon completion, a maskable interrupt is generated.

Automatic transfers can be done in one of three ways:

- A block of data can be transferred to the LCD controller with control strings automatically inserted to navigate through the LCD controller's internal RAM. This mode makes use of a second buffer in system memory to store the software commands necessary for LCD controller RAM addressing.
- A block of data can be transferred from system memory to the display controller while tying the LCD_RS pin low.
- A block of data can be transferred from system memory to the display controller while tying the LCD_RS pin high.

The Automatic Transfer Mode control bits (AUTOMODE[1:0]) of the DMAC Control/Status Register configure the DMAC for one of the three supported automatic transfer modes.

59.3.1.1.1 Automatic Display Data Transfers (AUTOMODE[1:0][1]=10)

The DMAC is designed to require no software intervention when transferring a display frame buffer from system memory to the external LCD controller. To accomplish this, the DMAC must automatically transmit control strings for LCD controller RAM addressing at appropriate times in the frame buffer data stream. LCD controller RAMs are typically organized into pages. After the page of RAM is filled, the controller's page address must be set to the next page. The DMAC contains counters that monitor the number of display data bytes written to the LCD controller to determine when the current page of controller RAM is filled. After the page is filled, the DMAC inserts a control string into the data stream to set the controller RAM's page address to the next page. The DMAC requires that this control string be stored in a command buffer in system memory. The organization of the command buffer is discussed later in this section.

In order to correctly move a complete frame of image data from system memory to the LCD controller, the DMAC must be programmed with the following LCD information:

- Number of bytes in a page of LCD image data
- Number of bytes in the LCD frame data buffer
- Type of LCD interface (serial or parallel)
- Data clock polarity of the LCD (for serial interfaces only)
- Chip select polarity for the LCD
- Transfer clock speed requirements of the LCD
- Control string needed at the start of each page of LCD display data (stored in a separate command buffer in system memory).

The DMAC uses the number of bytes per page of LCD image data information (stored in the BYTPPAGE[12:0] bits of the LCD CONFIG register) to determine when a set of addressing commands must be sent to the LCD. When the current display RAM page is full, the DMAC must increment the page address and reset the column address of the LCD controller to 0.

1. Automode [1:0] = 11 is reserved.

This is accomplished by reading the next control string from the command buffer and transmitting it to the LCD controller. The DMAC continues to send display data and command strings (as needed) until the controller's display RAM is filled. The transfer is finished when the number of bytes transferred equals the number of bytes in the buffer as defined by the DATABUFSIZ bits in the DATA BUFFER SIZE registers.

Figure 59-2 "Sample LCD Controller Memory Mapping (Monochrome)" on page 59-7 shows an example of how the external LCD controller RAM is filled during automatic DMAC writes. In this example, the LCD controller requires three "command" bytes to set the page and column address. Setting the GO bit of the control/status register begins the transfer of data from system memory to the LCD controller. The first byte written to the LCD sets the page address to 0. The next two bytes written are control strings that set the LCD column address to 0. Byte write #4 starts the transfer of image data to the LCD. The first byte of display data maps to column 0 of the display. The most-significant bit of the byte maps to row 7 and the LSB maps to row 0.

Bytes of display data continue to be written to the LCD until the DMAC's internal byte counter equals the value stored in the BYTPPAGE[12:0] bits of the LCD CONFIG register. At this point, the DMAC issues a command string, taken from the command buffer, to increment the page address and reset the column address of the LCD.

This sequence continues until the LCD controller's data RAM is filled. Once the display buffer has been filled, the IRQ bit of the DMAC Control/Status register becomes set. Also, the BUSY bit and GO bit clear. An interrupt will be generated if the IRQEN bit is set and the system is configured for DMAC interrupts.

Figure 59-3 "DMAC LCD Controller Memory Mapping (2-bit color/gray scale)" on page 59-8 shows the mapping for a 2-bit color/gray scale display. For this configuration, each column of data within the page requires 2 byte writes. The sequencing remains the same as the 1-bit per pixel case show in Figure 59-2 on page 59-7, but the number of writes to fill the page of display data nearly doubles.

The DMAC is designed to deal with a large variety of LCD controllers from multiple vendors. While the hardware interface to the external controller is somewhat standard, the software interface can vary from chip to chip. To deal with the software differences between LCD controllers, the DMAC makes use of a separate command buffer stored in system memory to go along with the display data buffer. Figure 59-4 "Memory Configuration for Automatic Display Data Transfers" on page 59-9 shows the organization of the information in system memory.

The LCD page and column command buffer should be set up to contain the commands necessary to navigate through the LCD controller's internal RAM. Each command in the buffer is preceded by a tag. The DMAC uses the tags to determine the value of the LCD_RS pin during the transmission of the command byte.

Figure 59-5 "Command Buffer Tag Organization" on page 59-10 shows how the commands and tags are organized in system memory. In this example, the command string length is 3. That is, three bytes are required to be transmitted to the LCD controller at the start of each page of LCD RAM. Each of the command bytes has an 8-bit tag placed adjacent to it in memory. The command is placed immediately after its tag in memory. Currently the DMAC only uses the least-significant bit of the tag field. This bit, labeled "RS" in Figure 59-5, sets the value of the LCD_RS pin while the command byte is being transferred to the LCD controller. If the RS tag bit is set to "1", LCD_RS will be driven to 1 during the transfer of the corresponding command byte. Similarly, if the RS tag bit is set to "0", LCD_RS will be driven to 0 during the transfer of the corresponding command byte.

Figure 59-6 "DMAC Automatic Mode Write Sequence (monochrome display)" on page 59-11 and Figure 59-7 "DMAC Automatic Mode Data Flow (AUTOMODE[1:0] = 10)" on page 59-12 show the sequence in which the bytes of data are written to the LCD controller when AUTOMODE[1:0] =10. These figures illustrate how command strings, taken from the command buffer, are interleaved with display data taken from the data buffer.

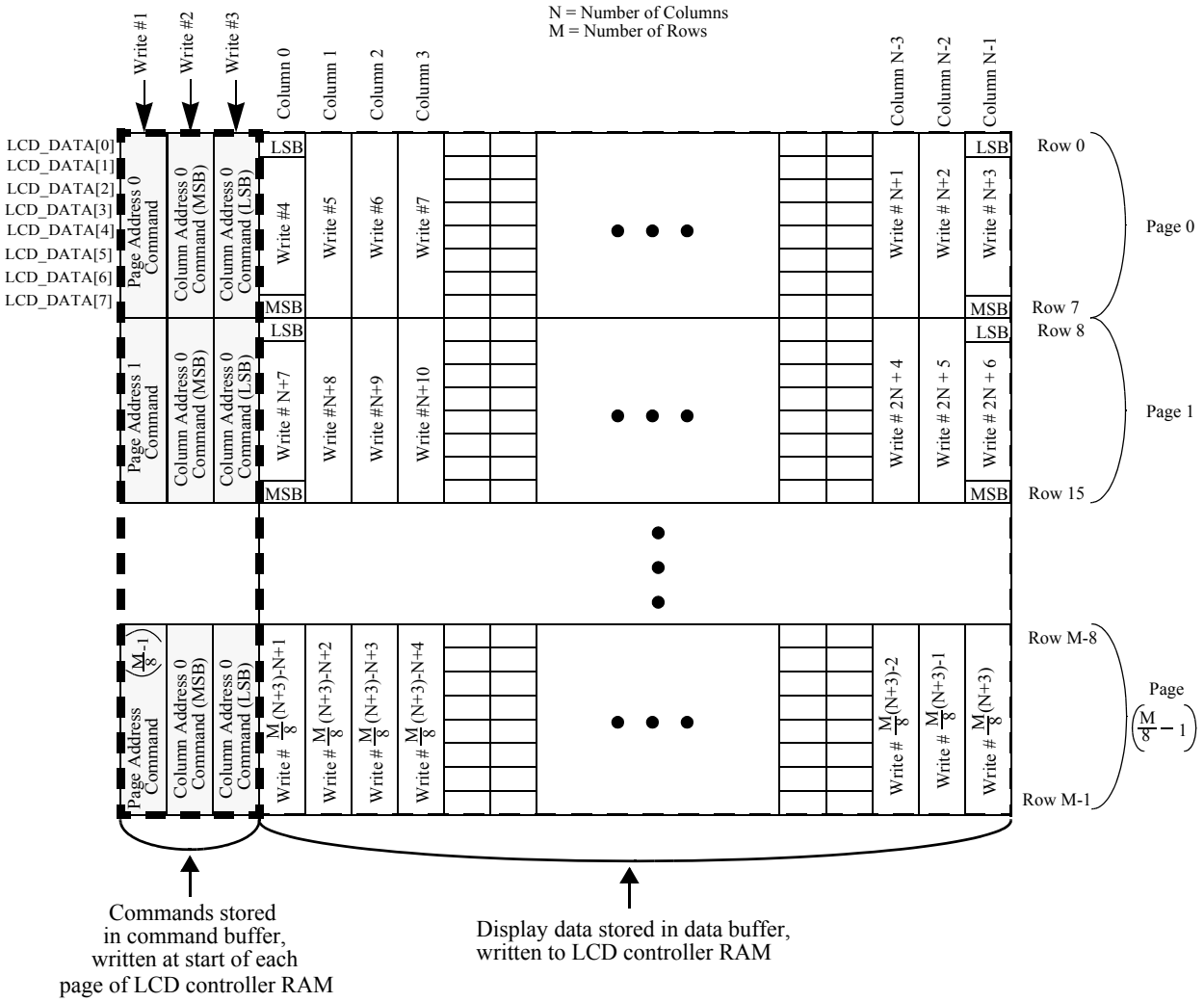


Figure 59-2. Sample LCD Controller Memory Mapping (Monochrome)

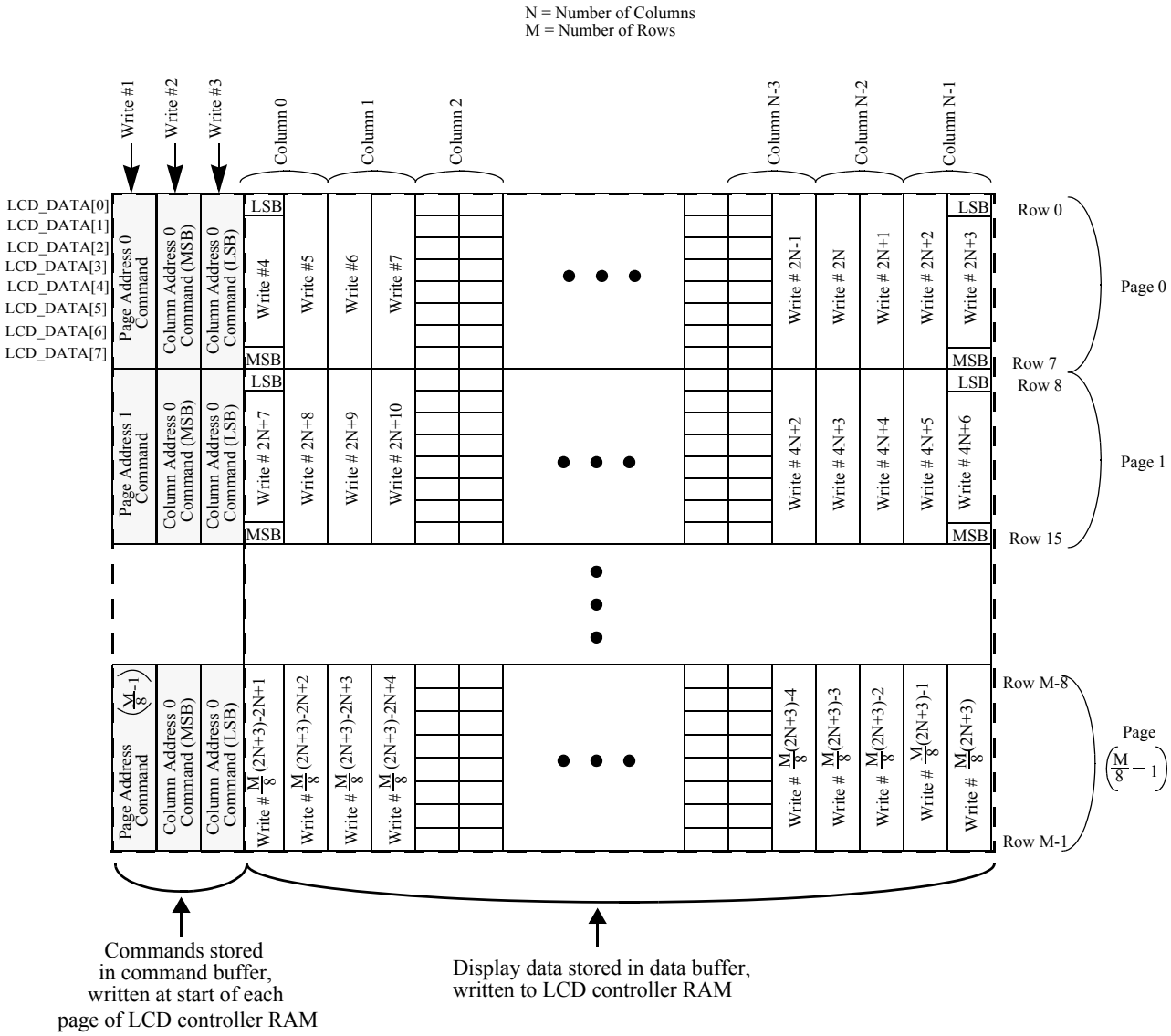


Figure 59-3. DMAC LCD Controller Memory Mapping (2-bit color/gray scale)

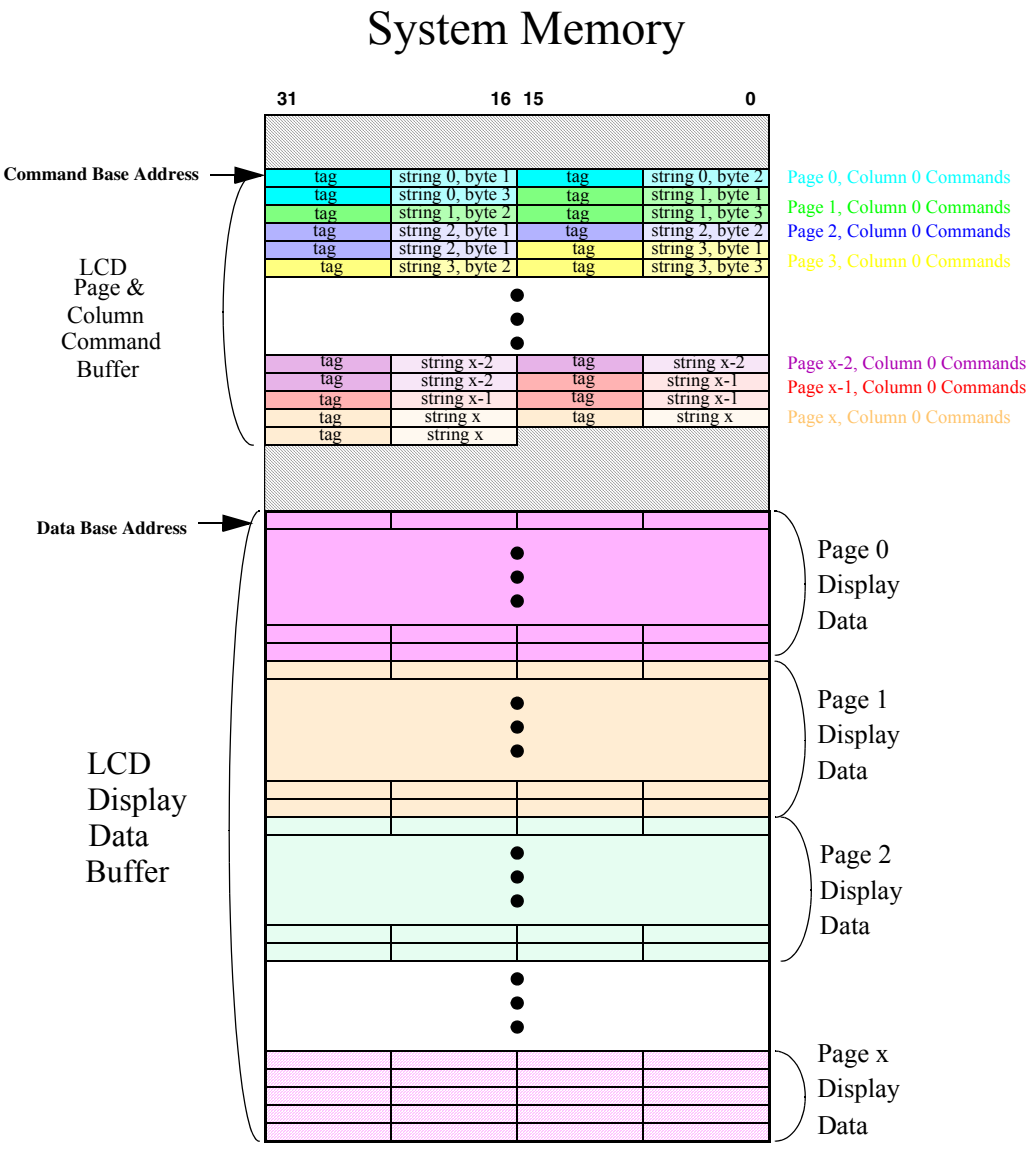


Figure 59-4. Memory Configuration for Automatic Display Data Transfers

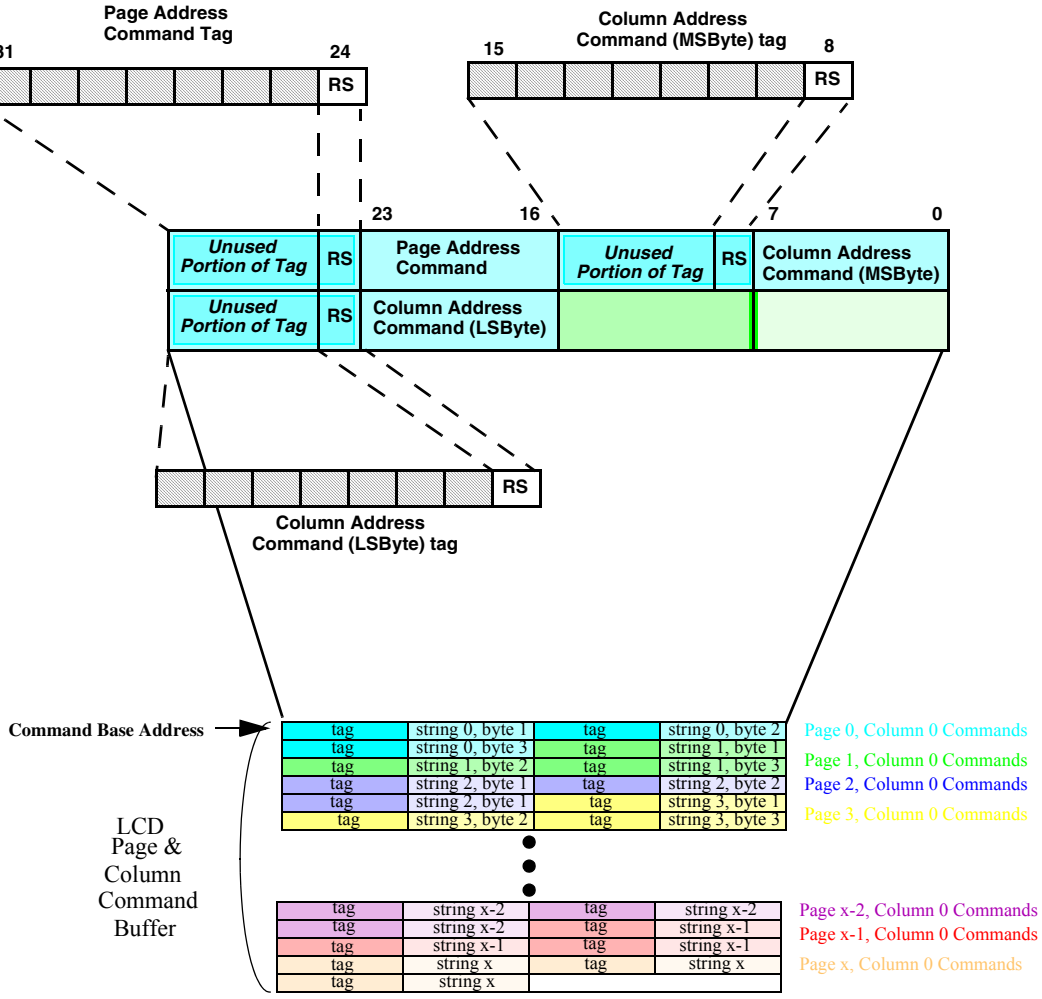


Figure 59-5. Command Buffer Tag Organization

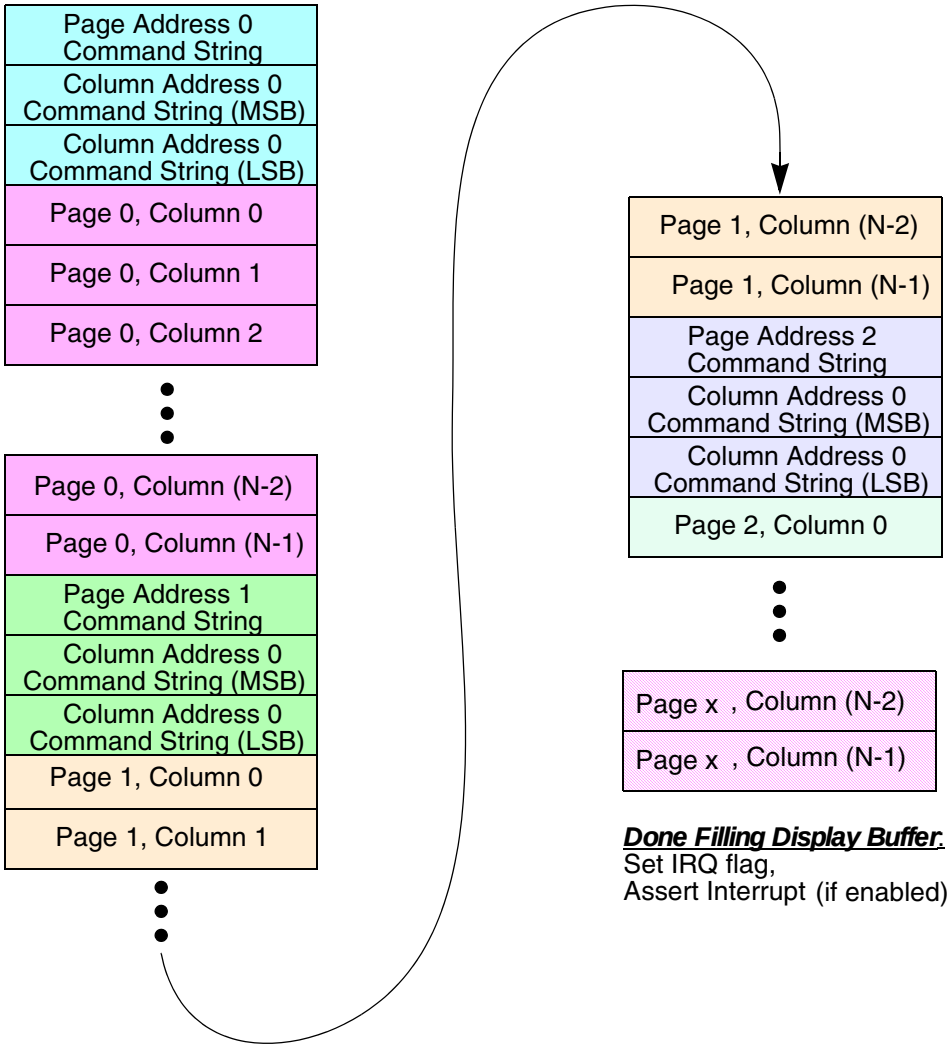


Figure 59-6. DMAC Automatic Mode Write Sequence (monochrome display)

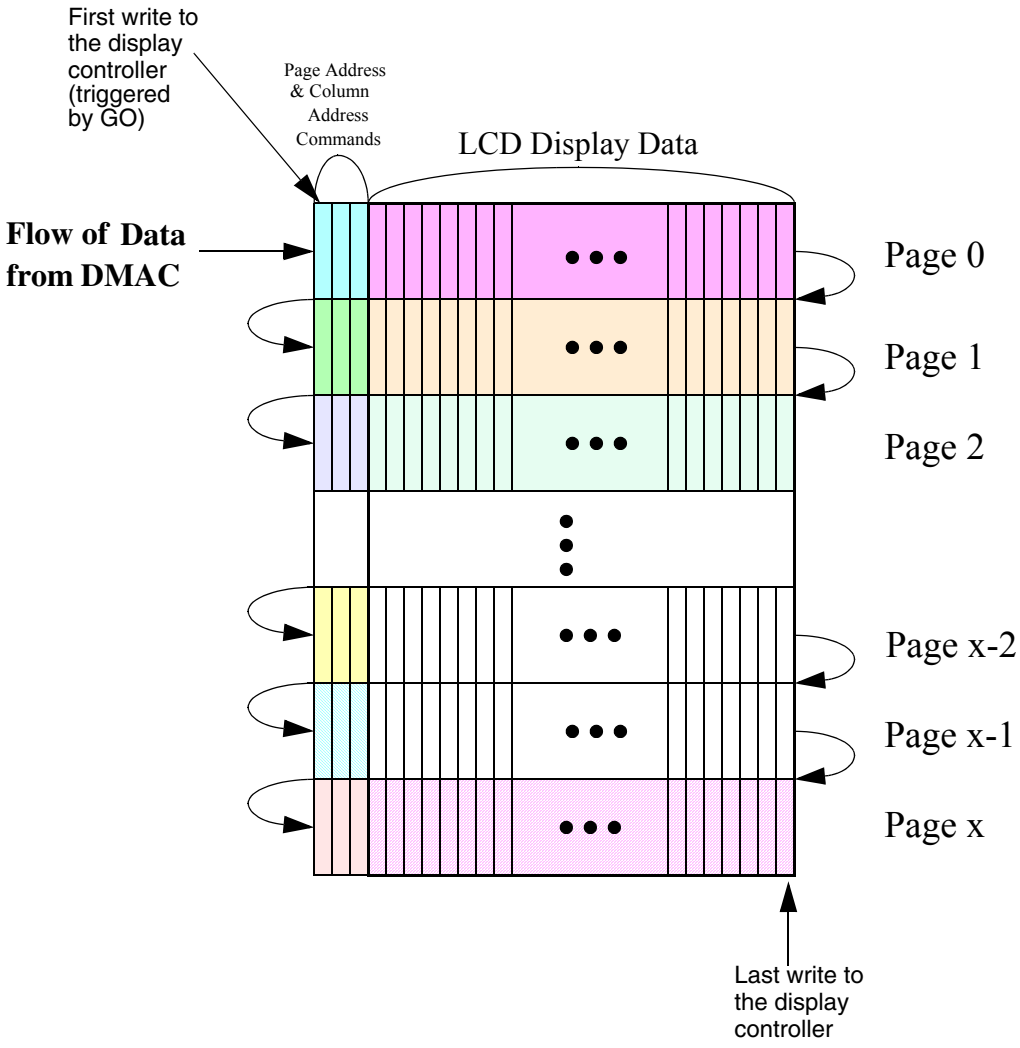


Figure 59-7. DMAC Automatic Mode Data Flow (AUTOMODE[1:0] = 10)

59.3.1.1.2 Automatic Command Data Transfers (AUTOMODE[1:0]=00)

The DMAC provides a convenient method of moving data from system memory to an external device, such as an LCD controller. When the AUTOMODE[1:0] bits = 00, the DMAC ignores the command buffer and transfers the contents of the specified data buffer to the DMAC output pins without inserting page and column addressing control strings. When AUTOMODE[1:0] = 00, the LCD_RS output is held to 0 during the entire transfer.

This method can be used for sending a block of commands to the LCD controller for initialization. The buffer transmitted is defined by the bits of the DATA BASE ADDRESS registers and the bits of the DATA BUFFER SIZE registers. Setting the GO bit of the DMAC Control/Status Register starts the

transfer and sets the BUSY status bit. After the number of bytes transferred to the external device is equal to the buffer size defined by the DATA BUFFER SIZE registers, the BUSY bit will clear, the GO bit will clear, and the IRQ flag will set. A system interrupt will be generated if the IRQEN bit is set.

59.3.1.1.3 Automatic Command Data Transfers (AUTOMODE[1:0]=01)

The DMAC provides a convenient general purpose method of moving data from system memory to an external device, such as an LCD controller. When the AUTOMODE[1:0] bits = 01, the DMAC ignores the command buffer and transfers the contents of the specified data buffer to the DMAC output pins without inserting page and column addressing control strings. When AUTOMODE[1:0] = 01, the LCD_RS output is held to 1 during the entire transfer.

The buffer transmitted is defined by the DATABASEADR[31:0] bits of the DATA BASE ADDRESS registers and the DATABUFSIZ[16:0] bits of the DATA BUFFER SIZE registers. Setting the GO bit of the DMAC Control/Status Register starts the transfer and sets the BUSY status bit. After the number of bytes transferred to the external device is equal to the buffer size defined by the DATA BUFFER SIZE registers, the BUSY bit will clear, the GO bit will clear, and the IRQ flag will set. A system interrupt will be generated if the IRQEN bit is set.

59.3.1.2 Direct IPAccess

Single bytes can be written to the external LCD controller via the 9-bit LCD WRITE DATA register. Any write to the LCD WRITE DATA register will result in a write to the external display controller, provided that the DMAC is not currently in the middle of a transfer (indicated by BUSY=1 in the Control/Status register). The value written to the LCDDAT[7:0] bits of the LCD Write Data register will be transmitted according to settings stored in the LCD transfer configuration register (i.e. the transfer is done in accordance with the settings of the XFRMODE, CSPOL, and SCKPOL bits of the LCD TRANSFER CONFIG register).

Typical LCD controllers use a register select signal to distinguish between display data writes and control command writes. The RS bit of the LCD WRITE DATA register determines whether a write is presented to the external controller as a command string or a display data string. The LCD_RS pin will be held to the value stored in the RS bit during the byte transfer. Since the transfer to the external device is slow compared to the system clock rates, the BUSY bit in the Control/Status Register is used to indicate that the transfer is in progress. A transfer initiated by a write to the WRITE DATA register will cause an interrupt if the IRQEN bit of the DMAC Control/Status register is set.

Another direct write cannot be done until the previous transfer has completed (BUSY=0).

59.3.2 Aborting DMAC Transfers

The DMAC Control/Status Register contains a bit, ABORT, which can be used to terminate a DMAC transfer that is in progress. Termination of the transfer occurs gracefully. Any byte transfer that is in progress when ABORT is asserted will be completed before the DMAC goes to an idle state. The BUSY status bit will clear upon entry to the idle state. The IRQ flag will also assert, indicating that the abort has completed. A DMAC interrupt will be generated, if enabled.

59.3.3 Low Power Mode Operation

The DMAC does not contain any control registers to program behavior in low power modes. The DMAC can be used in low power modes provided that ckih clock is available and the system clock is running at frequency greater than or equal to ckih clock. The DMAC requires ckih for the LCD clock generation and a system clock rate equal to or greater than ckih to be able read from system memory at a rate that is fast enough for the DMAC bit manipulator.

NOTE:

In the Neptune LTE configuration, the ckih clock is tied to system clock, mcu_clk, therefore the two clocks will always be equal.

59.4 DMAC Register Summary

The Register Summary lists all registers of the module by ascending address. The absolute address of each register is given, and the value of each bit for reads and writes is given using the conventions in the Legend.

Figure 0-18.

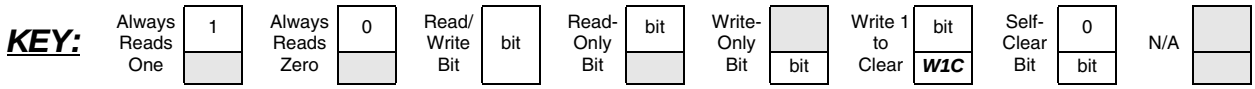


Table 59-2. DMAC Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA BASE ADDRESS_H ($2485_4000)	R	DATABASEADR[31:16]															
	W																
DATA BASE ADDRESS_L ($2485_4002)	R	DATABASEADR[15:2]														0	
	W																
DATA BUFFER SIZE_H ($2485_4004)	R	0															DATA BUF SIZ[16]
	W																
DATA BUFFER SIZE_L ($2485_4006)	R	DATABUFSIZ[15:0]															
	W																
COMMAND BASE ADDRESS_H ($2485_4008)	R	COMBASEADR[31:16]															
	W																
COMMAND BASE ADDRESS_L ($2485_400a)	R	COMBASEADR[15:2]														0	
	W																
COMMAND BUFFER SIZE_H ($2485_400c)	R	0															COM BUF SIZ[16]
	W																
COMMAND BUFFER SIZE_L ($2485_400e)	R	COMBUFSIZ[15:0]															
	W																
COMMAND STRING SIZE ($2485_4010)	R	0								COMSTRINGSIZ[7:0]							
	W																
FIFO CONFIG ($2485_4012)	R	0														BURST[1:0]	
	W																

Table 59-2. DMAC Register Summary (Continued)

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCD CONFIG ($2485_4014)	R	0			BYTPPAGE[12:0]												
	W																
LCD TRANSFER CONFIG ($2485_4016)	R	0													XFR MODE	CS POL	SCK POL
	W																
DMAC CONTROL/STATUS ($2485_4018)	R	0			AUTOMODE [1:0]		0		PROT1	IRQEN	IRQ	UNDR FLOW	TEA	0	BUSY	0	0
	W										W1C	W1C	W1C			ABORT	GO
LCD CLOCK CONFIG ($2485_401a)	R	0										DIVIDE[5:0]					
	W																
LCD WRITE DATA ($2485_401c)	R	0							RS	LCDDAT[7:0]							
	W																

59.5 DMAC Register Descriptions

The following figures and associated text give detailed descriptions of the various DMAC registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - — **r**: read only. Writing this bit has no effect.
  - — **w**: write only.
  - — **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - — **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - — **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - — **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - — **0**: Will reset to a logic 0.
  - — **1**: Will reset to a logic 1.
  - — **?**: The reset state is unknown.
  - — **u**: Unaffected by reset.

DATA BASE ADDRESS_H — Data Buffer Base Address — **Addr $2485_4000**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	DATABASEADR[31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 59-3. DATA BASE ADDRESS_H Descriptions

Name	Description
DATABASE ADR [31:16] Bits 15-0	**Data Buffer Base Address—** This register contains the most-significant 16 bits of the pointer in the R-AHB address space to the start of the LCD data buffer. This value is used in conjunction with DATA BASE ADDRESS_L to form the 32-bit data buffer base address.

DATA BASE ADDRESS_L — Data Buffer Base Address — **Addr $2485_4002**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	DATABASEADR[15:2]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 59-4. DATA BASE ADDRESS_L Descriptions

Name	Description
DATABASE ADR [15:2] Bits 15-2	**Data Buffer Base Address—** This register contains the least-significant 16 bits of the pointer in the R-AHB address space to the start of the LCD data buffer. This value is used in conjunction with DATA BASE ADDRESS_H to form the 32-bit data buffer base address. The data buffer base address value is forced to be word-aligned since the 2 lsb's are forced to 0. That is, data buffer base address [1:0] = 00.

NOTE:

The DMAC DMA address generator width is 17 bits. Therefore, it is possible that a data buffer can be incorrectly addressed by the DMAC if it extends beyond a 128 kbyte boundary. Therefore, it is recommended that the data buffer base address be set at the beginning of a 128 kbyte boundary. The sum of DATABASEADR[16:0] and DATABUFSIZ[16:0] must not exceed a multiple of 128K.

DATA BUFFER SIZE_H — Data Buffer Size — **Addr $2485_4004**

BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
															DATA BUF SIZ[16]
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 59-5. DATA BUFFER SIZE_H Descriptions

Name	Description
DATA BUFSIZ[16] Bit 0	**Data Buffer Size—** This register contains the most-significant bit of DATABUFSIZ[16:0], which defines the length (in bytes) of the LCD image or control buffer stored in system memory. This value is used to determine when the DMA transfer of the data buffer from system memory to the LCD controller is complete.

DATA BUFFER SIZE_L — Data Buffer Size — **Addr $2485_4006**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	DATABUFSIZ[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 59-6. DATA BUFFER SIZE_L Descriptions

Name	Description
DATA BUFSIZ [15:0] Bits 15-0	**Data Buffer Size—** This register contains the least-significant bits of DATABUFSIZ[16:0], which defines the length (in bytes) of the LCD image or control buffer stored in system memory. This value is used to determine when the DMA transfer of the data buffer from system memory to the LCD controller is complete.

COMMAND BASE ADDRESS_H Command Buffer Base Address **Addr $2485_4008**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	COMBASEADR[31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 59-7. COMMAND BASE ADDRESS_H Descriptions

Name	Description
COM BASEADR [31:16] Bits 15-0	**Command Buffer Base Address—** This register contains the most-significant 16 bits of the pointer in the R-AHB address space to the start of the LCD command buffer that contains the LCD controller page and column address commands. This buffer is used when the DMAC is configured for transfers with automatic page address and column address command insertion. This value is used in conjunction with COMMAND BASE ADDRESS_L to form the 32-bit data buffer base address.

COMMAND BASE ADDRESS_L — Command Buffer Base Address — Addr $2485_400a

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	COMBASEADR[15:2]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 59-8. COMMAND BASE ADDRESS_L Descriptions

Name	Description
COM BASEADR [15:2] Bits 15-2	**Command Buffer Base Address—** This register contains the least-significant 16 bits of the pointer in the R-AHB address space to the start of the LCD command buffer that contains the LCD controller page and column address commands. This buffer is used when the DMAC is configured for transfers with automatic page address and column address command insertion. This value is used in conjunction with COMMAND BASE ADDRESS_H to form the 32-bit command buffer base address. The command buffer base address value is forced to be word-aligned since the 2 lsb's are forced to 0. That is, command buffer base address [1:0] = 00.

NOTE:

The DMAC DMA address generator width is 17 bits. Therefore, it is possible that a command buffer can be incorrectly addressed by the DMAC if it extends beyond a 128 kbyte boundary. Therefore, it is recommended that the command buffer base address be set at the beginning of a 128 kbyte boundary. The sum of COMBASEADR[16:0] and COMBUFSIZ[16:0] must not exceed a multiple of 128K.

COMMAND BUFFER SIZE_H — Command Buffer Size — **Addr $2485_400c**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
																COMBUFSIZ[16]
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 59-9. COMMAND BUFFER SIZE_H Descriptions

Name	Description
COMBUFSIZ[16] Bit 0	**Command Buffer Size—** This register contains the most-significant bit of COMBUFSIZ[16:0], which defines the length (in bytes) of the LCD image or control buffer stored in system memory. This value is used to determine when the DMA transfer of the command buffer from system memory to the LCD controller is complete.

COMMAND BUFFER SIZE_L — Command Buffer Size — **Addr $2485_400e**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	COMBUFSIZ[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 59-10. COMMAND BUFFER SIZE_L Descriptions

Name	Description
COM BUFSIZ[15:0] Bits 15-0	**Command Buffer Size—** This register contains the least-significant 16 bits of COMBUFSIZ[16:0], which defines the length (in bytes) of the LCD image or control buffer stored in system memory. This value is used to determine when the DMA transfer of the command buffer from system memory to the LCD controller is complete.

COMMAND STRING SIZE — Command String Size — **Addr $2485_4010**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
									COMSTRINGSIZ[7:0]							
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 59-11. COMMAND STRING SIZE Descriptions

Name	Description
COM STRING SIZ[7:0] Bits 7-0	**Command String Size—** This register contains COMSTRINGSIZ[7:0] bits, which define the length (in bytes) of the LCD command string made up of the LCD controller page and column address commands. This command string is used when the DMAC begins to fill a new page of the LCD controller's RAM. It should be noted that these strings are only transmitted to the LCD controller when the DMAC is configured for transfers with automatic page and column address command insertion. The command strings are stored along with tags in the LCD command buffer located in system memory. The COMSTRINGSIZ[7:0] bits represent the number of bytes that must be sent to the LCD controller to set the page and column address, NOT the number of bytes needed to store the corresponding commands and tags in system memory.

FIFO CONFIG FIFO Configuration Register **Addr $2485_4012**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
															BURST[1:0]	
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 59-12. FIFO CONFIG Descriptions

Name	Description	Settings
BURST [1:0] Bits 1-0	**DMA Burst Length—** These bits select the length (in 32-bit words) of the DMA burst. That is, these bits determine the number of 32-bit word reads that occur when the DMAC has ownership of the R-AHB bus.	00 = Burst length set to 1 32-bit word 01 = Burst length set to 2 32-bit words 10 = Burst length set to 3 32-bit words 11 = Burst length set to 4 32-bit words

LCD CONFIG — LCD Controller Configuration — Addr $2485_4014

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
				BYTPPAGE[12:0]												
TYPE	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 59-13. LCD CONFIG Descriptions

Name	Description	Settings
BYTPPAGE [12:0] Bits 12-0	**LCD Bytes Per Page—** These 13 bits set the number of bytes per page for the external LCD controller connected to the DMAC module. These bits are used by the DMAC to determine when control characters must be sent to the controller. The number of bytes per page is determined from the number of columns on a display and the number of bits per pixel (color depth).	0 = 0 bytes per LCD page 1 = 1 byte per LCD page 2 = 2 bytes per LCD page . . . 8191 =8191 bytes per LCD page

LCD TRANSFER CONFIG LCD Controller Transfer Configuration **Addr $2485_4016**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
														XFR MODE	CS POL	SCK POL
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 59-14. LCD TRANSFER CONFIG Descriptions

Name	Description	Settings
XFRMODE Bit 2	**Image Data Transfer Width—** This bit selects the width of the LCD display interface databus. Data transfers to the LCD controller can be done serially or in parallel.	0= 1 bit serial transfers 1 = 8 bit parallel transfers
CSPOL Bit 1	**Chip Select Polarity—** This bit selects the polarity of the LCD_CS signal. This bit affects the behavior of the LCD_CS pin in both serial and parallel mode.	0 = LCD_CS is asserted low. Therefore, the DMAC will drive LCD_CS to 0 during a serial transfer of data to the external LCD controller. In parallel mode, LCD_CS will normally be high. LCD data will change when LCD_CS goes low. The rising edge of LCD_CS is used by the LCD controller to latch the parallel data. 1 = LCD_CS is asserted high. Therefore, the DMAC will drive LCD_CS to 1 during a serial transfer of data to the external LCD controller. In parallel mode, LCD_CS will normally be low. LCD data will change when LCD_CS goes high. The falling edge of LCD_CS is used by the LCD controller to latch the parallel data.
SCKPOL Bit 0	**Serial Data Clock Polarity—** This bit selects whether the serial data will transition on the rising or falling edge of the serial data clock during transfers of data in serial mode. This bit has no effect during parallel data transfers to the external LCD display device	0 = Serial data will transition on the rising edge of the serial clock. Therefore, data should be latched by the display device on the falling edge of the serial clock. 1 = Serial data will transition on the falling edge of the serial clock. Therefore, data should be latched by the display device on the rising edge of the serial clock.

DMAC CONTROL/STATUS Control/Status Register **Addr $2485_4018**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
				AUTOMODE [1:0]				PROT 1	IRQE N	IRQ	UNDR FLOW	TEA		BUSY	ABOR T	GO
TYPE	r	r	r	rw	rw	r	r	rw	rw	w1c	w1c	w1c	r	r	slfclr	slfclr
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 59-15. DMAC Control/Status Descriptions

Name	Description	Settings
AUTO MODE [1:0] Bits 12-11	**Automatic Transfer Mode—** These bits control how the DMAC transfers the data buffer to the LCD controller upon being triggered by the GO bit of the DMAC Control/Status Register. The DATA BASE ADDRESS bits designate the location of the transferred data buffer in system memory. Transfers triggered by the GO bit can be configured to automatically insert page address and column address commands in the data stream to navigate through the LCD controller's RAM. This is done without CPU intervention. The DMAC can be also be configured to send the data buffer to the LCD controller without inserting any command strings. In this mode, the value of the LCD_RS signal can be configured to be either 1 or 0 during the transfer.	00 = DMAC transfers the data buffer defined by DATA BASE ADDRESS and DATA BUFFER SIZE without inserting control strings. With this setting, the LCD_RS signal is tied to 0 during the entire transfer. 01= DMAC transfers the data buffer defined by DATA BASE ADDRESS and DATA BUFFER SIZE without inserting control strings. With this setting, the LCD_RS signal is tied to 1 during the entire transfer. 10 =When transmitting the data buffer defined by DATA BASE ADDRESS and DATA BUFFER SIZE, the DMAC inserts command strings to set the LCD controller's page and column addresses. These command strings are taken from the buffer defined by COMMAND BASE ADDRESS and COMMAND STRING SIZE. This insertion of command strings is done at the beginning of each page of LCD controller RAM. 11 = Reserved
PROT1 Bit 8	**R-AHB Protection Code—** This bit controls the value is driven onto HPROT[1] signal of the R-AHB each time a DMAC access is done. This value controls the type of access being done on the bus (user or privileged). The dmac_hprot[0] output of the DMAC is tied to 1 to indicate a data access. The PROT1 bit resets to 0.	0 = dmac_hprot[1:0] driven to 01 indicating DMAC accesses will be user data accesses. 1 = dmac_hprot[1:0] driven to 11 indicating DMAC accesses will be privileged data accesses.

Table 59-15. DMAC Control/Status Descriptions (Continued)

Name	Description	Settings
IRQEN Bit 7	**Interrupt Enable—** This bit controls whether a DMAC interrupt is generated upon completion of a DMAC transfer. DMAC transfers can end three ways: 1) When a transfer error occurs on the R-AHB. 2) When the DMAC completes the transfer of data to the LCD controller. 3) When a DMAC transfer is aborted by setting the ABORT bit in the DMAC Control/Status Register. DMAC transfers are initiated by assertion of the GO bit or a write to the LCD WRITE DATA register. Clearing the IRQEN bit will prevent an interrupt from being generated when the DMAC completes a transfer.	0 = DMAC interrupt is masked. No interrupt will be generated from the DMAC. 1 = DMAC interrupt will be generated upon the completion of a DMAC transfer.
IRQ Bit 6	**DMAC Interrupt Flag—** This bit indicates that an interrupt condition occurred in the DMAC. The interrupt flag is set a DMAC transfer is completed or aborted. DMAC transfers are initiated by assertion of the GO bit or a write to the LCD WRITE DATA register. This bit is cleared by writing a 1 to the IRQ bit of the Control/Status register.	0 = DMAC interrupt is not pending 1 = Interrupt condition occurred in the DMAC.
UNDR FLOW Bit 5	**DMAC FIFO Underflow—** This bit indicates that the DMAC FIFO became empty during the transfer of data from system memory to the LCD controller. Although FIFO underflow does not cause an error in the DMAC transfer, it does indicate the output portion of the DMAC had to wait for the FIFO to be filled during the transfer. This bit is cleared by writing a 1 to the UNDRFLOW bit of the Control/Status register. NOTE: The underflow flag will not set until after some data has been read into the DMAC fifos. Underflow only sets after a condition where the fifos have data, they empty, and then do not refill fast enough for the DMAC bit manipulator. In this case, the DMAC bit manipulator stalls while it waits for data to be read from system memory into the fifos. The underflow flag will not set in a case where a DMAC transfer is initiated, but the DMAC is never granted control of the bus, and thus able to fill the fifos. The fifos have to go from a non-empty state to an empty state in order for UNDERFLOW to set.	0 = No underflow occurred in DMAC FIFO. 1 = DMAC FIFO became empty during the transfer of data to the LCD controller. This caused the output portion of the DMAC to wait, delaying transfer of data to the LCD controller.

Table 59-15. DMAC Control/Status Descriptions (Continued)

Name	Description	Settings
TEA Bit 4	**DMAC DMA Transfer Error—** This bit indicates that the DMAC DMA received a transfer error acknowledge from the R-AHB bus while trying to read data from system memory. This is indicated by HRESP0=1 and HREADY=1 on the R-AHB bus. This is a fatal condition and will cause the DMAC to terminate its transfer. This bit is cleared by writing a 1 to the TEA bit of the DMAC control/status register	0 = No transfer error acknowledge error occurred. 1 = A transfer error acknowledge was received by the DMAC DMA from the R-AHB.
BUSY Bit 2	**DMAC Busy—** This bit indicates that the DMAC is in the process of transferring the image buffer from system memory to the LCD controller. Data transfer is initiated by setting the GO bit of the DMAC Control/Status Register or by writing data to the LCD Write Data register. This bit is a read-only bit and will clear once the transfer of data to the LCD controller has completed.	0 = DMAC idle. 1 = DMAC in process of transferring image buffer.
ABORT Bit 1	**Abort DMAC Transfer—** This bit causes the DMAC to abort the data transfer currently underway. Termination of the transfer occurs gracefully. That is, any byte transfer to the LCD controller in progress will be completed before the DMAC goes to an idle state. This bit will be cleared automatically when the DMAC has transitioned to the idle state.	0 = Do not abort current data transfer. 1 = Abort the DMAC data transfer currently in progress.
GO Bit 0	**Start DMAC Transfer—** This bits starts the automatic transfer of image data from system memory to the external LCD controller. This bit will always read 1 until the data transfer has completed. After the data transfer has completed or been aborted, the GO bit will read 0. **Note:** Writing 1 to the GO bit will have no effect if the BUSY bit of the DMAC Control/Status Register is set.	0 = Do not start DMAC transfer. 1 = Start DMAC image data transfer.

LCD CLOCK CONFIG — LCD Clock Configuration Register — **Addr $2485_401a**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
											DIVIDE[5:0]					
TYPE	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 59-16. LCD CLOCK CONFIG Descriptions

Name	Description	Settings
DIVIDE [5:0] Bits 5-0	**LCD Clock Divide Value—** This bit sets the divide ratio used to generate the LCD clock from the ckih clock **(See LCD Clock Configuration on page -37.)** for a detailed explanation of the fractional divider programming). Setting DIVIDE[5:0] to 0 disables the LCD_CLK by gating ckih clock from the fractional divider. These bits must be set to some non-zero value before transfers to the LCD can occur.	Frequency relations: 0 = LCD_CLK off 1 = LCD_CLK = ckih clock/128 2 = LCD_CLK=ckih clock*2/128 . . . 62 = LCD_CLK=ckih clock*62/128 63 = LCD_CLK=ckih clock*63/128

LCD WRITE DATA — LCD Write Data — **Addr $2485_401c**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
								RS	LCDDAT[7:0]							
TYPE	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 59-17. LCD Write Data Descriptions

Name	Description	Settings
RS Bit 8	**LCD Register Select—** This bit determines the value placed on the LCD_RS pin during a byte transfer from the WRITE DATA register.	0 = Data written to LCDDAT[7:0] is transferred to the LCD controller as command data (the LCD_RS signal is tied to 0 during the transfer). 1 = Data written to the LCDDAT[7:0] register is transferred to the LCD controller as display data (the LCD_RS signal is set to 1 during the transfer).
LCDDAT [7:0] Bits 7-0	**LCD Controller Data—** Data written to these 8 bits is transferred to the external LCD controller. This register gives direct write access to the LCD controller. The data is determined to be command or display data via the RS bit of the LCD Write Data register. Data written to this register will only be transferred to the LCD controller if the DMAC is not in middle of another transfer (indicated by the BUSY bit of the Control/Status register). A read of these bits will give the last value written to this register (or the reset value), NOT a value from the LCD controller.	

59.6 LCD Controller Interface

The DMAC transfers data from the display memory buffer in system memory to the external display device. Transfers can be done via a 4-wire serial, 3-wire serial, or an 8-bit parallel interface.

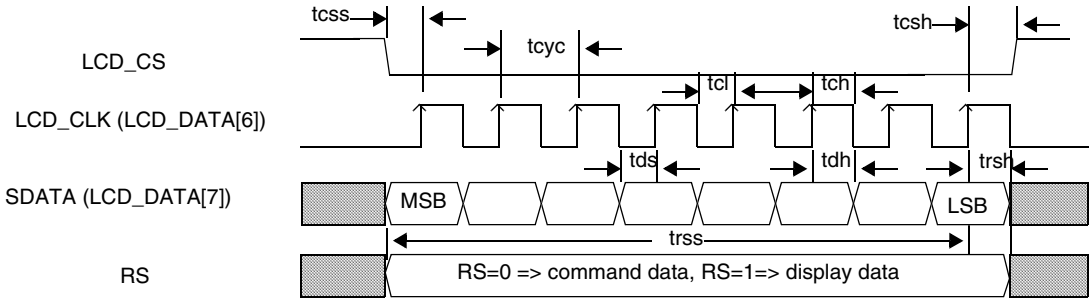
59.6.1 Serial Interface

In serial mode (XFRMODE bit in LCD TRANSFER CONFIG register set to 0), data is transmitted to the display device via four signals: LCD_CS, LCD_DATA[7:6], and LCD_RS. The data is transmitted on the MSB of the LCD_DATA bus (LCD_DATA[7]), the serial clock toggles on LCD_DATA[6], and LCD_RS is used to tell the display whether the data being transmitted is display data or command data. LCD_CS is used as a chip select signal. The LCD_CS signal will be asserted during all transfers, and will be held asserted for sequential bytes being transmitted. The polarity of the LCD_CS signal is programmable using the CSPOL bit of the LCD TRANSFER CONFIG register.

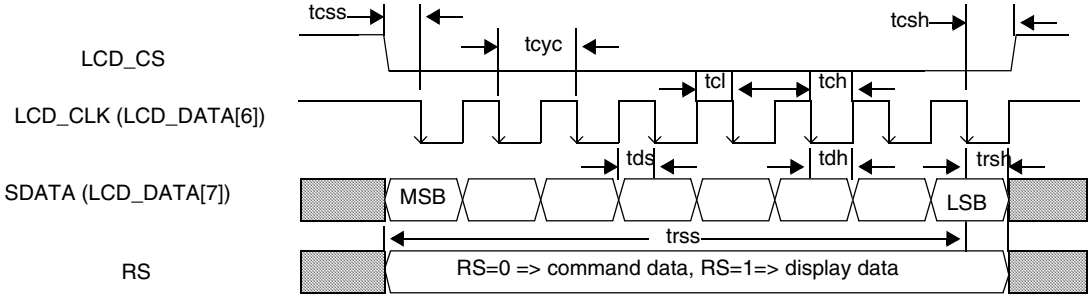
Figure 59-8 "DMAC Serial Transfers to LCD Device" on page 59-34 shows the timing associated with the transfer of one byte of data to the display. The polarity of the serial data clock on the LCD_DATA[6] pin can be configured using the SCKPOL bit of the LCD TRANSFER CONFIG register. This bit will need to be set in accordance with the external display device requirements. If the external device latches the serial data on the rising edge of the serial clock, SCKPOL needs to be set to 1. If the external device latches the serial data on the falling edge of the serial clock, SCKPOL needs to be cleared.

Table 59-18. DMAC Serial Interface Timing

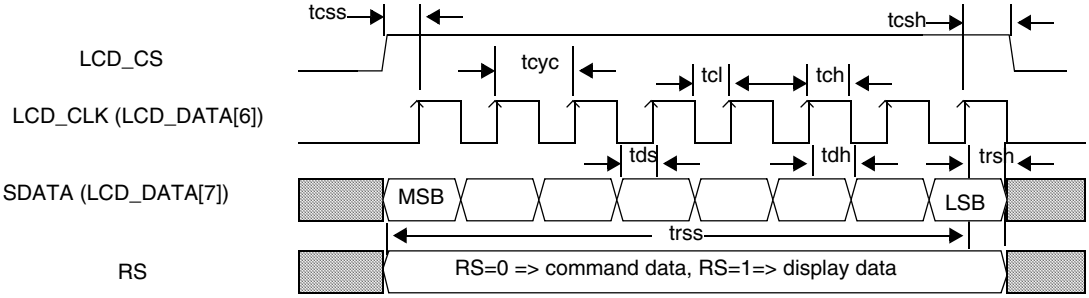
Symbol	Parameter	Min (ns)	Typ (ns)	Max (ns)
t_{css}	Chip select setup time	t_{cyc} / 2		
t_{csh}	Chip select hold time	t_{cyc} / 2		
t_{cyc}	Serial clock cycle time	39		2641
t_{cl}	Serial clock low pulse	18		
t_{ch}	Serial clock high pulse	18		
t_{ds}	Data setup time	t_{cyc} / 2		
t_{dh}	Data hold time	t_{cyc} / 2		
t_{rss}	Register select setup time	15 * t_{cyc} / 2		
t_{rsh}	Register select hold time	t_{cyc} / 2		



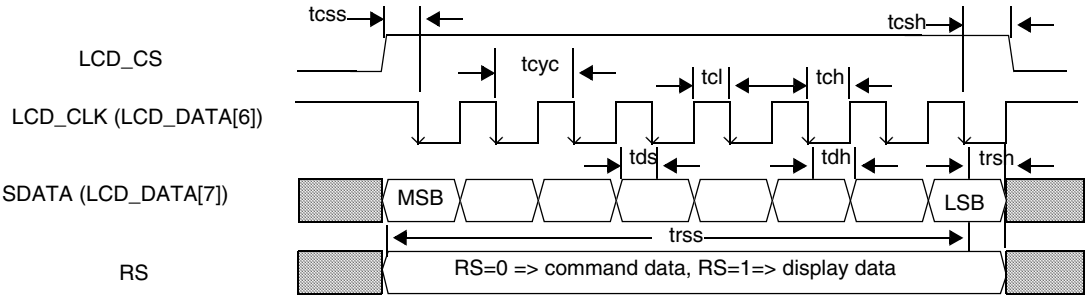
The diagram above illustrates the timing when the SCKPOL = 1, CSPOL = 0



The diagram above illustrates the timing when the SCKPOL = 0, CSPOL = 0



The diagram above illustrates the timing when the SCKPOL = 1, CSPOL = 1



The diagram above illustrates the timing when the SCKPOL = 0, CSPOL = 1

Figure 59-8. DMAC Serial Transfers to LCD Device

59.6.2 Parallel Interface

The DMAC module can also be configured to execute parallel transfers of display data from system memory to an external display device. Figure 59-9 "DMAC Parallel Transfers to LCD Device" on page 59-36 shows the timing associated with a DMAC parallel transfer to an external LCD device. The timing is based on the frequency of LCD_CLK, which is programmable via the LCD CLOCK CONFIG register. In parallel mode, the LCD_CS pin is used as a write strobe by the external LCD controller. The latching edge of LCD_CS occurs at least one LCD_CLK cycle after the data is made available to the display on the LCD_DATA[7:0] pins. The polarity of the LCD_CS signal is programmable using the CSPOL bit in the LCD TRANSFER CONFIG register.

The parallel mode timing of the DMAC is slightly modified from the original DMAC timing found in Neptune LT and LTS. For Neptune LTE and ULS, LCD_CS signal is delayed one mcu_clk cycle. This provides a non-zero set-up time for the LCD_RS and LCD_DATA pins with respect to the edge up LCD_CS.

Table 59-19 on page 59-37 summarizes the resulting timing from the proposed change. t_{skew} is intended to represent timing skew between the signals due to top-level chip routing. The t_{skew} value that is specified below assumes equal loading on the LCD pins. Additional skew can be introduced by imbalanced loads on the LCD pins. The Min setup and values documented in Table 59-19 are achieved when the DMAC is programmed for maximum LCD interface speed. This is done by programming the DIVIDE[5:0] bits of the DMAC LCD CLOCK CONFIG register to 0x3f. The LCD_RS setup time with respect to LCD_CS (tRSS) is independent of the programmed DIVIDE[5:0] value. tRSS is determined by the mcu_clk frequency and t_{skew} at all values of DIVIDE[5:0]. All other timing parameters are dependent on the programmed value of DIVIDE[5:0].

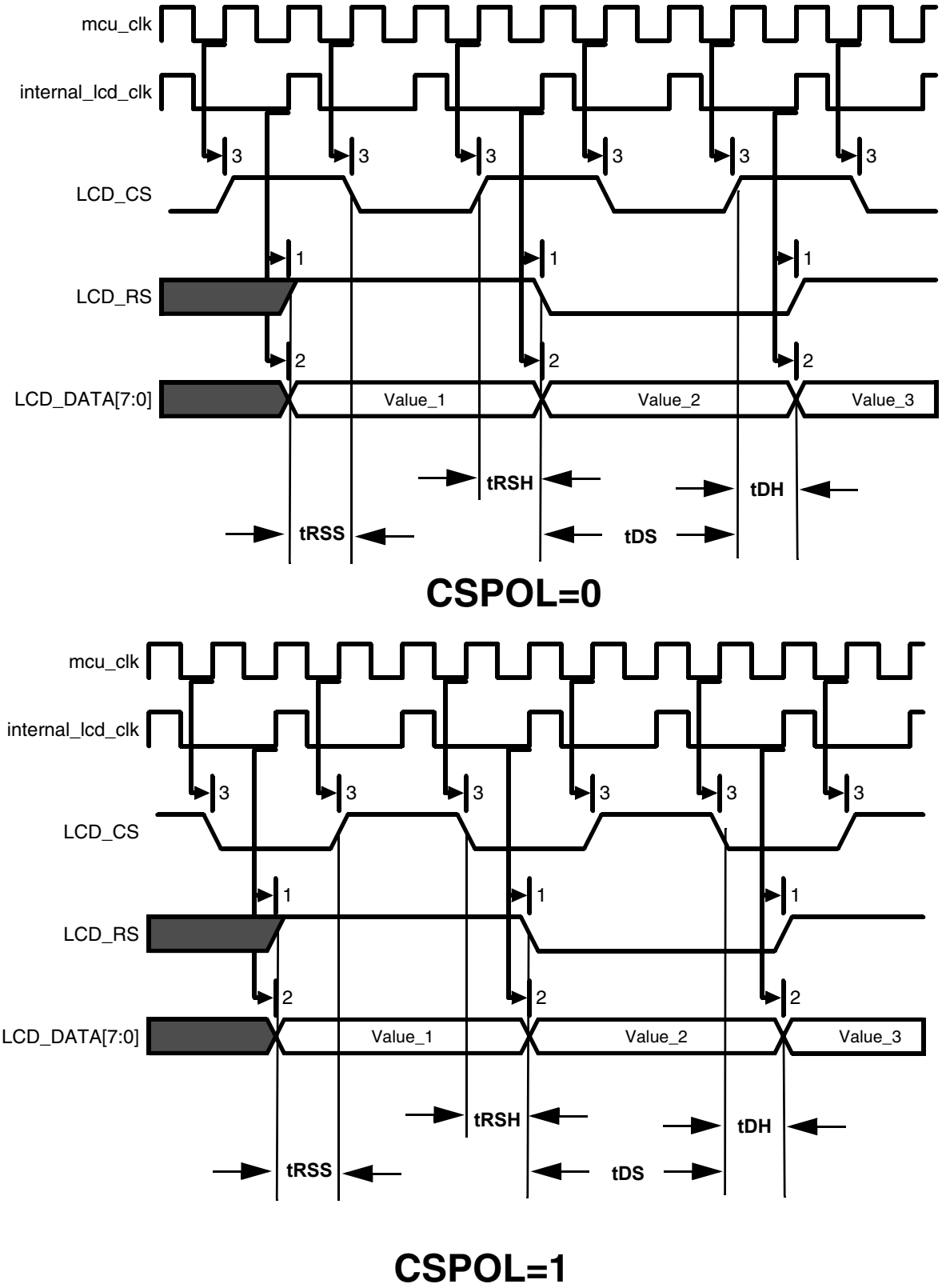


Figure 59-9. DMAC Parallel Transfers to LCD Device

Table 59-19. Parallel DMAC Timing Parameters

Symbol	Parameter	Min (ns)	Typ (ns)	Max (ns)
t_{skew}	Signal Skew	0	3	5
tRSS	LCD_RS setup time	(mcu_clk period) - 5ns	(mcu_clk period) +/- 3ns	(mcu_clk period) + 5ns
tRSH	LCD_RS hold time	(mcu_clk period) - 5ns	Determined by DIVIDE value	127 * (mcu_clk period) + 5ns
tDS	DATA setup time	3 * (mcu_clk period) - 5ns	Determined by DIVIDE value	129 * (mcu_clk period) + 5ns
tDH	DATA hold time	(mcu_clk period) - 5ns	Determined by DIVIDE value	127 * (mcu_clk period) + 5ns

59.7 LCD Clock Configuration

The DMAC uses a fractional divider to generate the LCD data clock from the ckih clock signal. The fractional divider is programmed by setting a divide value using the DIVIDE[5:0] bits of the LCD CLOCK CONFIG register. The frequency of LCD_CLK is calculated according to the following formula:

$$\text{LCD\_CLK} = \frac{CKIH \times DivideValue}{128}$$

The divide value is taken directly from the value stored in the DIVIDE[5:0] bits of the LCD CLOCK CONFIG register. Setting DIVIDE[5:0] to 0 disables the LCD_CLK signal. These bits must be set to some non-zero value before transfers to the LCD can occur.

Table 59-20. LCD_CLK Frequency Range

CKIH Clock Frequency (MHz)	Minimum LCD_CLK frequency (MHz)	Maximum LCD_CLK frequency (MHz)	Resolution of step (kHz)
52	0	25.59	406.25
26	0	12.8	203.13
16.8	0	8.27	131.25

59.8 R-AHB Interface and DMAC FIFOs

The DMAC uses a DMA interface to the R-AHB to access system memory without CPU intervention. Therefore, the DMAC DMA steals cycles from the CPU. When the DMAC needs data, the DMA will request control of the R-AHB from the ARM7TDMI-S processor and read 32-bit words from system memory. The number of 32-bit words read during the burst is determined by the value stored in the BURST[1:0] bits of the FIFO CONFIG register. Control of the R-AHB is relinquished by the DMA after

the burst is complete. The amount of R-AHB bandwidth required by the DMAC can be controlled by system software. The frequency of LCD frame updates and the rate of the LCD_CLK will affect how often the DMAC requests use of the R-AHB.

The data read by the DMA interface is stored in two 32-bit x 4 word FIFOs that are used to buffer data between the DMA and the DMAC display interface. One FIFO is used to buffer data from the command buffer and the other buffers data from the display data buffer. When one of the FIFOs has enough room to store the data read during a DMA burst, it will request servicing. Since the DMA burst length is programmable, the occupancy level of the FIFOs can vary when servicing is requested.

Chapter 60
Hash Acceleration Module (HAC)

Revision History

Revision	Date	Author	Description
0.0	02/22/02	Pritam Kabe	Initial Version: Taken from Patriot Indy Baseband IC specification.
0.1	03/14/02	Pritam Kabe	Updated the initial version for Neptune LTS: Made the MCORE to ARM edits. Modified the block diagram - AITC for INT, AEIM for EIM, AHB for MLB, IP Bus for PIG Bus. Modified the HAC Alternate Master Behaviour Table. Modified the memory map.
0.2	10/16/02	Pritam Kabe	Updated the figure for the Padding Value Creation. Corrected the Block Count value (19 bits wide) and the number of zero padding bits.
0.3	05/07/03		Updated for LTE specification release.

60.1 Module Overview

The Hash Acceleration Module (HAC) is a hardware accelerator designed to assist in the hashing of the external FLASH during boot. It does so by being able to quickly run large amounts of data through a SHA-1 algorithm to produce a 160 bit hash. To facilitate rapid access to data, the HAC is an alternate bus master to the microcontroller. Control and status information is passed between the HAC to the microcontroller via registers accessible over the microcontroller's peripheral bus. In addition, there is a single interrupt from the HAC to the microcontroller to indicate the completion of the hash.

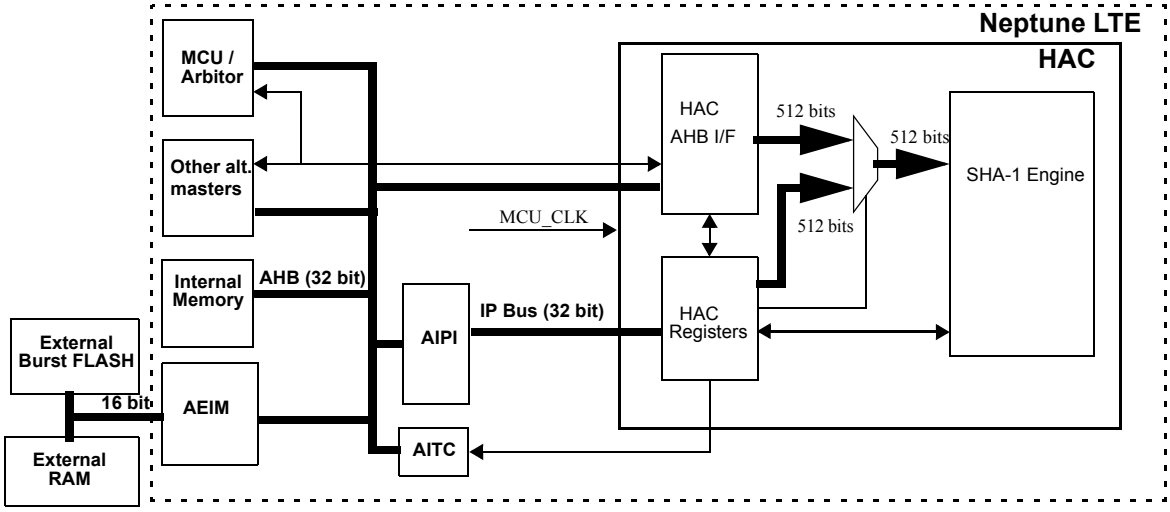


Figure 60-1. Top Level HAC Diagram

60.2 Functional Description

The Hash Acceleration Module (HAC) accelerates the creation of a SHA-1 hash over selected FLASH contents. The SHA-1 algorithm is a one-way hash algorithm that creates a 160 bit hash of any length input. By its nature, SHA-1 works on 512 bits at a time, and in the end will pad a final block with a defined sequence so that the final block is also a 512 bit entity.

60.2.1 HAC Usage

The intention of the HAC is to allow software to calculate a SHA-1 hash over a number of large, potentially non-contiguous segments of FLASH. The general method for executing this process would be to enable the HAC interrupt, and then add each segment to the hash. In this context, a segment is any amount of contiguous FLASH space which is assumed to start and end on 512 bit boundaries as shown in Figure 60-3. For the first segment, the values in the START_ADDR register and BLOCK_COUNT register (number of 512 bit blocks in the given segment) are set, and then the START bit in the HAC's CTL register is set to begin the process. Once this process completes, the HAC's DONE bit is set, which can also create an interrupt. Following this, the software can once again set the START_ADDR and BLOCK_COUNT for the next FLASH segment to be added, and then set the CONTINUE bit in the HAC's CTL register which would indicate to it to add this FLASH data to the already accumulated hash value. Again, the HAC's DONE bit will be set upon completion. Note, you cannot change the START_ADDR and BLOCK_COUNT values while processing is in progress. Also, you can't START or CONTINUE when the BLOCK_COUNT is zero.

Once all segments of FLASH have been hashed in this fashion, the software would need to write the total number of 512 bit blocks that were hashed into the BLOCK_COUNT register, and then set the PAD bit in the HAC's CTL register. This will cause the HAC to add one more 512 bit value to the hash as defined by the SHA-1 algorithm (a 1 in the MSB and the total message bit count in the 64 LSBs with zeros padding the rest to make the final block 512 bits--see Figure 60-2). Once complete, the DONE bit will again be set at which time the software can read the 160 bit result from the 5 HSH registers. A more detailed description of each of these bits is given in Section 60.4, "Register Map," on page 60-7, and the overall flow itself is summarized in Section 60.3, "Programmer's Model," on page 60-6.

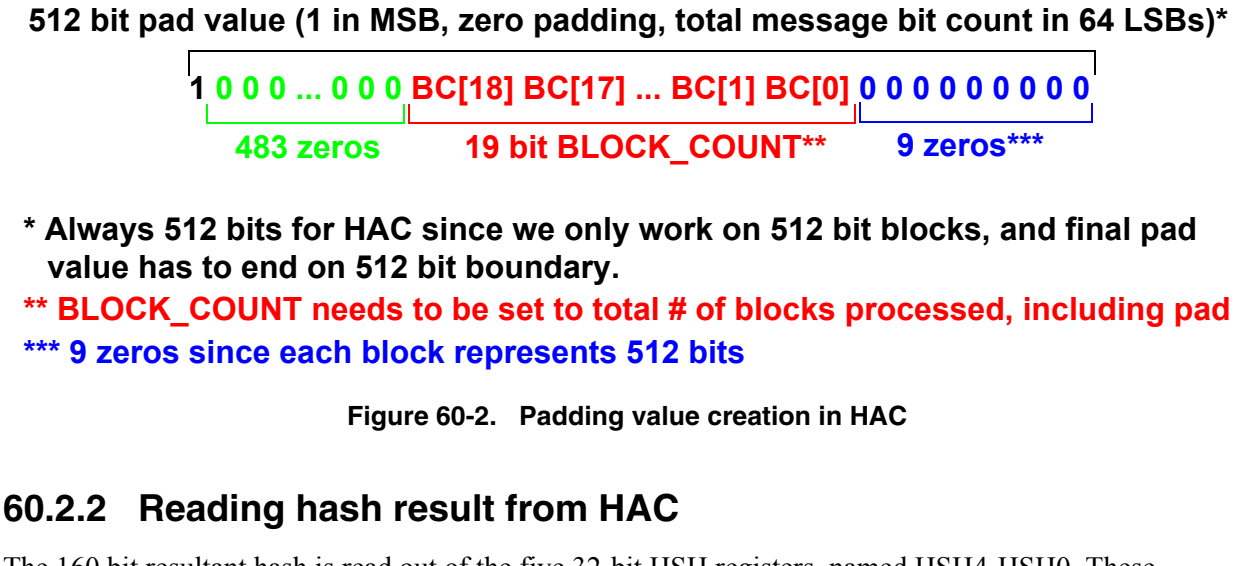


Figure 60-2. Padding value creation in HAC

60.2.2 Reading hash result from HAC

The 160 bit resultant hash is read out of the five 32-bit HSH registers, named HSH4-HSH0. These represent the 160 bit result from left to right. That is, HSH4 contains bits 159 to 128, HSH3 contains bits 127 to 96, etc. In some SHA-1 standards, the bits 159 to 128 would be label H0 or sha_A, HSH3 as H1 or sha_B, etc. The naming of H0 versus HSH4 should not be confused. The HSH4 is a convention used here as it is normal practice in IC specifications to list the most significant register to least significant register in this way.

As an example, if the HAC was to apply SHA-1 to the following 512 bit block, the result after padding is given below.

Table 60-1. Single 512 block of example input data (in hexadecimal)

A50E3CC8	ED7ABBE7	1991A93B	51A3BEAA
0B0329B1	10EA5962	E4A63775	DD370C75
012A451A	945C393E	02582C2B	63DF1CDC
3F165EC7	71CADA1C	19386403	0AFED421

HSH4 = 0FD4E942; HSH3 = 0F6EC6F7; HSH2 = 05494050; HSH1 = A1C3B546

HSH0 = 18AE7C08

60.2.3 HAC Data Interface

In order to rapidly hash the FLASH space, the HAC is an alternate master to the MicroController Unit (MCU).This means the HAC is capable of stealing cycles away from the MCU on its main bus (the AHB) so that the software doesn't need to copy the data into the module, but instead the HAC can directly access

the data from the FLASH to be added to the hash. The software control of the HAC is still done via the standard memory-mapped peripheral bus used to control other hardware modules (see Section 60.3, “Programmer’s Model,” on page 60-6).

Because the HAC is only going to be used to hash the FLASH, and the FLASH memory blocks are always very large spaces, the starting address of any given block of FLASH to be added to the hash is assumed to begin and end on 512 bit boundaries as shown in Figure 60-3. This assumption simplifies the hardware by reducing the size and processing requirements of the HAC itself.

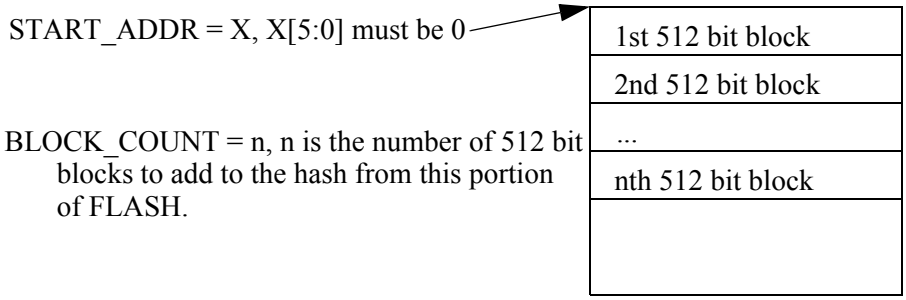


Figure 60-3. Parameters used to identify a block of FLASH to be hashed

In case the data to be added to the hash doesn’t exist in one contiguous block, the HAC does support continuing the hash from another block area. To do this, the software would just need to set the START_ADDR and BLOCK_COUNT values to the next block, and then tell the HAC to continue its hashing. This can repeat as often as needed to obtain all blocks to be added to the hash. Once all blocks have been included, the software tells the HAC to add the appropriate SHA-1 padding block to the end of the sequence, and once complete, the entire 160 bit hash will be available to the software via 5 memory mapped registers in the HAC.

The HAC does accesses over the AHB system bus of the Neptune LTE IC. Table 60-2 is a summary of the AHB interface and the HAC alternate master behaviour.

Table 60-2. HAC Alternate Master Behaviour

HAC signals	Description	How driven by HAC
haddr[31:0]	Address bus	Address of current word being read
htrans[1:0]	Transfer type	Drive 00b (Internal Cycle) when not requesting bus Drive 10b (Non-Sequential) when requesting bus and then drive 11b (Sequential)
hwrite	Transfer direction	Always drive 0(since only do reads)
hsize[1:0]	Transfer size	Always drive 10b (since only read words)
hresp0	Transfer Abort	Indicates current transfer invalid
hready	Transfer Complete	Indicates current transfer done
hprot[1:0]	Protection Encoding	Always drice 11b (Previledged Mode and Data Fetch only)
hac_br	Bus request	HAC request for bus arbitration
amarb_bg	Bus grant	HAC owns bus for accesses

60.2.3.1 Calculating HAC Processing Time and Percentage Bus Cycles Stolen

The amount of time it takes the HAC to do the requested actions will depend on the frequency of the HAC clock, the amount of information to be processed, possibly the speed of the FLASH being hashed, and any access delays incurred when attempting to fetch data or acquire the bus. The SHA-1 accelerator in the HAC module takes 80 cycles to process 512 bits. While a given 512 bits are being processed, the next 512 bits are being fetched if possible. There is a 1 cycle overhead for each transfer of 512 bits. Hence, the number of MCU cycles it takes to process a given segment of n 512-bit data blocks is

*((cycles it takes to acquire first 512 bit block) + (n*81) + (any misc. overhead))*

This equation will be true unless the FLASH accesses for the next 512 bits takes longer than 80 cycles it takes to process the current 512 bits. If the FLASH being hashed has burst capability, the HAC will take advantage of it. Assuming a 16 bit interface, the first read of each 512 bit block will be assumed to be non-sequential, but then the next 31 reads will be assumed to be sequential. As an example, assuming 5-1-1-1 speeds, this would be 5 + 31*1 = 36 cycles which is less than the 80 cycles required. However, if burst FLASH is not used, and assuming a 4 cycle read for each access, it takes 128 cycles to fetch 512 bits which would in fact then limit the overall speed of the HAC module.

This same information can also be used to determine the percentage of bus cycles that the HAC module would steal from the MCU. Going back to the burst FLASH example given above, the HAC module would take 36 out of every 81 cycles, leaving the rest for the MCU to do other things. If there is nothing else to be done, the MCU should be in WAIT, and these extra cycles will just be dead time on the bus to save power. In the example of the non-burst FLASH with 4 wait states, the HAC would use virtually 100% of the MCU bus in trying to keep data flowing just as fast as possible, as one would expect.

Note, the padding operation also takes 81 cycles to complete. This operation does not fetch any data, so the wait states of the memory will have no effect. Also note that the equation given above for processing time does not include the time for the padding since padding is really a separate step from the HAC point of view. That is, an operation started with a START or CONTINUE will not include the padding.

60.2.3.2 Aborting, Resetting, and Stopping HAC Processing

Once the HAC begins its processing, there are two methods to interrupt it-- software reset, or assertion of software stop bit by software.

If for some reason the hashing currently being done is no longer desired or needed, its processing can be aborted by setting the SWRST bit in the CTL register. Setting this bit will immediately reset the HAC state machine, halt any pending HAC data access, return all HAC register bits to their reset state, and return the HAC back to its idle state. The DONE flag will not be set. It is recommended that the user assert the STOP bit prior to writing to the SWRST bit to ensure the HAC is not in the middle of a bus request when the reset occurs.

The SWRST bit can also be used to simply put the HAC in a known reset state if desired.

If the software simply wants to delay the HAC's processing of a given block, it can set the STOP bit in the CTL register. The HAC will finish any currently pending access and then prevent any further accesses until after STOP is cleared. It will not drop down to its low power state. Once software clears the STOP bit, processing will resume from the point it was frozen. If the STOP bit is written when the START or CONTINUE bits are written, processing will begin once the STOP bit is cleared. STOP does not affect the PAD operation since the PAD operation does not require a bus access.

There is hardware that prevents the START_ADDR and BLOCK_COUNT registers from being changed when the HAC is processing (BUSY bit is set). This includes the time when it has been halted via the setting of the STOP bit. There is also hardware that prevents the START and CONTINUE bits from being set when the BLOCK_COUNT is zero.

60.2.4 HAC Clock Usage and Power Management

The HAC module runs at the same clock frequency that the MCU does. The HAC will automatically gate as many internal clocks as possible when it is not processing. There is no software bit that needs to be set to reduce the power consumption of the module when it is not in use. It is possible to read and write the HAC registers with the HAC in this low power state.

The HAC will continue processing in any MCU low power mode in which it receives a clock from the system.

60.2.5 HAC disable via LASER fuse

To ensure the HAC does not interfere with previous generation functionality, a laser fuse output is routed to the module. If this fuse is left unblown, the HAC will be disabled. Disabling the HAC will gate the clock to the module, prevent the module's registers from being access (an attempt at accessing will result in a bus time-out just as if the module didn't exist), and gate the system bus request output to ensure there is no way the HAC would take over the system bus by mistake.

If this fuse is blown, the HAC will be enabled to behave as described in the remainder of this specification.

60.2.6 Running HAC with various burst memory configurations

To maximize bus efficiency, the HAC is capable of doing burst reads from memory. This may or may not be supported by the memory being accessed. For the M310S on Neptune LTE, the HAC will always attempt to burst read 16 accesses. There is no known reason to attempt to limit this.

Since the HAC design is also used on other ICs, it has bits to configure various burst modes of the HAC, the burst_config bits in the HAC's CTL_STAT register. These can come into play if the MCU has a cache accessing flash in critical word first mode, and so the burst lenght has to be limited to the length of the cache line fill. Since the Neptune LTE does not have a cache, these bits will not have any affect on the HAC's operation.

For completeness, the various setting are given in Table 60.2-3.

Table 60.2-3. Mapping of burst config bits (no affect on Neptune LTE)

BURST_CONFIG	Meaning
00	16 WORD burst reads are done
01	4 WORD burst reads are done
10	8 WORD burst (ARM cores sometimes require this)
11	No burst reads will be done (no known use case)

60.3 Programmer's Model

This section will describe the normal method of using the module. If ERROR bit is ever set, the software should take the appropriate action.

1. Set IMSK as desired (normally, clear it to enable interrupts so software can go into WAIT while HAC is running).
2. Put starting address of 1st block to be processed into START_ADDR.

3. Put number of 512 bit blocks in 1st block to be processed into BLOCK_COUNT.
4. Set START bit, and either poll for DONE or wait for HAC interrupt.
5. Check for ERROR bit, and clear DONE bit.
6. If no more blocks to process, go to step 11.
7. Put starting address of next block to be processed into START_ADDR.
8. Put number of 512 bit blocks of next block to be processed into BLOCK_COUNT.
9. Set CONTINUE bit, and either poll for DONE or wait for HAC interrupt.
10. Goto step 5.
11. Put total number of 512 bit blocks processed into BLOCK_COUNT (note, if only 1 block was processed, this value should already be correct)
12. Set PAD bit, and either poll for DONE or wait for HAC interrupt.
13. Clear DONE bit (no chance ERROR bit was set since no system bus reads were done)
14. Read HSH4 - HSH0 to obtain the 160 bit SHA-1 result.

60.4 Register Map

The HAC's registers can be accessed only as 32 bit values. If any register in the HAC's address space that is outside of the memory map defined here is accessed, a bus exception will be generated. No bus exception is generated for writing to unimplemented bits. Bus exceptions will also occur if 1) one attempts to write a 1 to either the START bit or CONTINUE bit in the CTL_STAT register when the BLOCK_COUNT register is zero OR 2) if one tries to change either the START_ADDR or BLOCK_COUNT values when the HAC is busy.

KEY: Always Reads One (1) | Always Reads Zero (0) | Read/Write Bit (bit) | Read-Only Bit (bit) | Write-Only Bit (bit) | Write 1 to Clear (bit / *W1C*) | Self-Clear Bit (0 / bit) | N/A

Table 60-4. HAC Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTL_STAT ($2485_8000)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	BURST CONFIG		0	0	0	0	STOP	IMSK	BUSY	ERROR	DONE
	W								PAD	CON TINUE	START	SWRST				W1C	W1C
START_ADDR ($2485_8004)	R	START_ADDR[31:16]															
	W																
	R	START_ADDR[15:6]										0	0	0	0	0	0
	W																
BLOCK COUNT ($2485_8008)	R	0	0	0	0	0	0	0	0	0	BLOCK_COUNT[22:16]						
	W																
	R	BLOCK_COUNT[15:0]															
	W																
HSH4 ($2485_800c)	R	HSH[159:144]															
	W																

Table 60-4. HAC Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	R	HSH[143:128]															
	W																
HSH3 ($2485_8010)	R	HSH[127:112]															
	W																
	R	HSH[111:96]															
	W																
HSH2 ($2485_8014)	R	HSH[95:80]															
	W																
	R	HSH[79:64]															
	W																
HSH1 ($2485_8018)	R	HSH[63:48]															
	W																
	R	HSH[47:32]															
	W																
HSH0 ($2485_801c)	R	HSH[31:16]															
	W																
	R	HSH[15:0]															
	W																

60.4.1 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various HAC registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read and is cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

CTL HAC Control Register **Addr $2485_8000**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	8	6	5	4	3	2	1	0
						BURST CONFIG		PAD	CON TINUE	START	SWRST	STOP	IMSK	BUSY	ERROR	DONE
TYPE	r	r	r	r	r	rw	rw	slfclr	slfclr	slfclr	slfclr	rw	rw	r	w1c	w1c
RESET	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Table 60-5. CTL_STAT Description

Name	Description	Settings
Bits 31-11	**Unimplemented Bits—** Always reads as zero; writing has no effect.	N/A
BURST CONFIG Bits 10:9	**Burst Configuration —**These read/write bits configure the burst read nature of the HAC's data interface. These bits really have no affect on the operation of the HAC for Neptune LTE (see Section 60.2.6, "Running HAC with various burst memory configurations," on page 60-6 for more information). The burst_config bits reset to 00.	00 =Do 16 word burst. 01 =Do 4 word burst. 10 = Do 8 word burst. 11 = Do not burst read.
PAD Bit 8	**PAD hash—**This self-clearing bit indicates that the HAC should add the appropriate padding to the end of the data structure. Adding padding is equivalent to running one more block at the end of the existing hash. The DONE bit will be set when this is complete. The BLOCK_COUNT register must contain the total number of 512 bit blocks that were contained in the hash before the PAD bit is set for the padding data structure to be correct. If the SWRST bit is set on the same cycle the PAD bit is set, the software reset will take precedence and no hashing will take place. Writing a one to the PAD bit has no function if the DONE bit is set. If the STOP bit is high when the PAD bit is set, the HAC will start the processing anyway since the PAD operation does not require bus accesses.	0 = Do not add SHA-1 padding to data structure. 1 = Add SHA-1 padding to data structure using value in block count register.
CONTINUE Bit 7	**Continue HAC Processing —**This self-clearing bit starts the processing of the HAC. The bit always reads as 0. The new values read in will be added to the existing hash. If the SWRST bit is set on the same cycle the CONTINUE bit is set, the software reset will take precedence and no hashing will take place. Writing a one to the CONTINUE bit has no function if the DONE bit is set. If the STOP bit is high when the CONTINUE bit is set, the HAC will not start the processing until the STOP bit is cleared. Once the STOP bit is cleared, processing will immediately begin. If software attempts to write a 1 to the CONTINUE bit when the BLOCK_COUNT register is zero, the write will not work, and a bus exception will be issued instead.	0 = Do not continue the HAC processing 1 = Continue the HAC processing at the start address by adding the new values read to the existing hash.

Table 60-5. CTL_STAT Description (Continued)

Name	Description	Settings
START Bit 6	**Start HAC Processing —**This self-clearing bit starts the processing of the HAC. The bit always reads as 0. Setting this bit indicates to the hardware to initialize the hash machine before beginning. If the SWRST bit is set on the same cycle the START bit is set, the software reset will take precedence and no hashing will take place. Writing a one to the START bit has no function if the DONE bit is set. If both the START and CONTINUE bit are set in the same write, the START bit functionality will take precedence (i.e. bit counter will be cleared). If the STOP bit is high when the START bit is set, the HAC will not start the processing until the STOP bit is cleared. Once the STOP bit is cleared, processing will immediately begin. If software attempts to write a 1 to the START bit when the BLOCK_COUNT register is zero, the write will not work, and a bus exception will be issued instead.	0 = Do not start the HAC processing 1 = Start the HAC processing a new hash value.
SWRST Bit 5	**Software Reset of HAC —** This self-clearing bit initiates a software reset of the entire HAC module. The bit always reads as 0. A software reset will reset all state machines back to their default states thereby halting hashing function in progress, as well as all register values back to their reset states. It will clear all status bits (BUSY/ERROR/DONE) and any pending interrupts. If both the SWRST and START, CONTINUE, or PAD bits are set in the same cycle, the software reset will take precedence and no hashing function will take place. The SWRST is automatically released internally via two clocks of the HAC's clock, so the clock must run at least two cycles before the user can write new values to any register bits. It is required that if a hash function is currently in progress that the STOP bit be set prior to setting the SWRST bit to ensure no bus accesses are in progress when the SWRST occurs.	0 = Do not perform a software reset of the HAC 1 = Perform a software reset of the module
STOP Bit 4	**Stop HAC Processing —** This read/write bit controls whether the HAC can continue an active process. Once the STOP bit is set, the HAC will halt its processing as soon as any pending data transfers are complete. Software must clear the STOP bit manually to resume processing. STOP must be cleared for a START or CONTINUE. If either of these is issued with STOP set, and the STOP is later taken away, the processing will immediately begin. A PAD operation will occur with the STOP bit set since it doesn't require any bus accesses. The SWRST bit can be used to prevent the processing from proceeding if desired. The STOP bit is cleared by a SWRST. STOP resets to a 0.	0 = HAC allowed to process data 1 = HAC will stop processing following the completion of any pending access
IMSK Bit 3	**HAC Interrupt Mask—** Controls whether the HAC interrupt is masked. The only source of the HAC interrupt is the DONE bit. If the DONE bit is set, and the IMSK is cleared, the HAC interrupt will be asserted. The IMSK bit resets to a 1.	0 = HAC interrupt is unmasked 1 = HAC interrupt is masked

Table 60-5. CTL_STAT Description (Continued)

Name	Description	Settings
BUSY Bit 2	**HAC is busy—**When this bit is set, it indicates that the HAC is currently hashing the FLASH. It does not create an interrupt. Since the start bit is self-clearing, this bit is the only way for software to know that a hashing is currently being done or not. This is a read only bit that reflects the state of the HAC. This bit will remain high when a process has been frozen due to the setting of the STOP bit. This bit is cleared by a software reset (SWRST).	0 = The HAC is not currently hashing the FLASH. 1 = The HAC is currently hashing the FLASH.
ERROR Bit 1	**Error occurred when hashing FLASH —** When this write 1 to clear bit is set, it indicates that the hashing was aborted due to a bus error during a HAC access over the AHB. If a bus error occurs, the HAC will set the ERROR bit and then end the processing by setting the DONE bit. This bit is cleared by a software reset (SWRST).	0 = The hashing process finished without any AHB access errors. 1 = The hashing process was halted abnormally due to a AHB access error.
DONE Bit 0	**Done processing —** This write 1 to clear bit is set when the selected processing initiated by the START bit has completed. It will create an interrupt if the IMSK bit is clear. It is cleared by writing a 1 to this bit. This bit is cleared by a software reset (SWRST). **Note:** The DONE bit must be cleared for the setting of the START, CONTINUE, or PAD bits to have any effect.	0 = HAC interrupt is not pending. 1 = HAC has finished most recent selected processing. Interrupt is pending if unmasked.

START_ADDR — HAC Start Address Register — Addr $2485_8004

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	START_ADDR[31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	START_ADDR15:6]															
TYPE	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 60-6. START_ADDR Description

Name	Description	Settings
START_ADDR [31:6] Bits 31-6	**Starting address block to hash —** This register contains the starting address of the block to be added to the hash. Note, the starting address is assumed to be on a 512 bit boundary, hence the lower 6 bits cannot be written. There is hardware in the HAC that will prevent reads to this register from occurring when the HAC is processing (HAC's BUSY bit is set). Any attempt to write to the register when BUSY is set will result in a bus exception.	N/A

BLOCK COUNT — HAC BLOCK COUNT Register — Addr $2485_8008

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
										BLOCK_COUNT[22:16]						
TYPE	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	BLOCK_COUNT[16:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 60-7. BLOCK_COUNT Description

Name	Description	Settings
Bits 31-23	**Unimplemented Bits—** Always reads as zero; writing has no effect.	N/A
BLOCK_COUNT [22:0] Bits 22-0	**Block count —** This register contains the number of 512 bit blocks that will be hashed. When either START or CONTINUE are set, this register indicates the number of 512 bit blocks that are run through the hash machine. When PAD is set, this number is used as part of the padding value (see Figure 60-2 on page 60-3). There is hardware in the HAC that will prevent reads to this register from occurring when the HAC is processing (HAC's BUSY bit is set). Any attempt to write to the register when BUSY is set will result in a bus exception. Note, If software attempts to write a 1 to the START or CONTINUE bits when the BLOCK_COUNT register is zero, the write will not work, and a bus exception will be issued instead.	N/A

HSH4 — HAC HSH4 Register — Addr $2485_800c

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	HSH[159:144]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	1	1	0	0	1	1	1	0	1	0	0	0	1	0	1

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	HSH[143:128]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	1

HSH3 — HAC HSH3 Register — Addr $2485_8010

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	HSH[127:112]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	1	1	0	1	1	1	1	1	1	0	0	1	1	0	1

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	HSH[111:96]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	0	1	0	1	0	1	1	1	0	0	0	1	0	0	1

HSH2 — HAC HSH2 Register — Addr $2485_8014

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	HSH[95:80]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	0	0	1	1	0	0	0	1	0	1	1	1	0	1	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	HSH[79:64]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	1	0	1	1	1	0	0	1	1	1	1	1	1	1	0

HSH1 — HAC HSH1 Register — Addr $2485_8018

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	HSH[63:48]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	1	0	0	0	0	0	0	1	1	0	0	1	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	HSH[47:32]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	1	0	1	0	1	0	0	0	1	1	1	0	1	1	0

HSH0 — HAC HSH0 Register — Addr $2485_801c

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	HSH[31:16]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	1	0	0	0	0	1	1	1	1	0	1	0	0	1	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	HSH[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	1	1	0	0	0	0	1	1	1	1	1	0	0	0	0

Table 60-8. HSH Description

Name	Description	Settings
HSH [159:0] HSH4 Bits 31-0 HSH3 Bits 31-0 HSH2 Bits 31-0 HSH1 Bits 31-0 HSH0 Bits 31-0	**Hash Result—** This set of registers holds the 160 bit Hash result of the SHA-1 process. It becomes valid once the DONE bit is set. It will remain valid until the START, CONTINUE, or SWRST bits are set. See Section 60.2.2, “Reading hash result from HAC,” on page 60-3 for more information on how to properly interpret these registers. **Note:** The registers have a strange reset value, but this value should be of no interest to the user.	N/A

Chapter 61
Data RAM Arbiter Module (DRA)

Revision History

Revision	Date	Author	Changes
0.0	11/8/2001	Steven Olsen	Initial Release
0.1	11/25/2001	Steven Olsen	Add port list. Generic port names used in place of peripheral names. Changed arbitration description. Spare port is added. Time Outs defined for both read and write.
0.2	11/28/2001	Khandaker Azad	Change of names and removal of some I/O pins Change in configuration register Typo
0.3	12/07/2001	Khandaker Azad	Insertion of SPMBIF. Changes in some pin description. Removal of ndbgctl pin.
0.4	12/17/01	David Liddle	Added in Low power mode behavior. Changed registers to support a combined read/write timeout. Added description of what happens when the timeout values are updated.
0.5	12/21/01	David Liddle	Updated because of the interleaved memory approach. Removed support for X memory muxing. Changed ports from 5 to 4. Removed the X select bit from the registers. Made the read timeout 1 bit wider.
0.6	1/10/02	David Liddle	Updates after review in Australia/NIDC - Changed port timeouts to active low - Clarified the low power behavior - Clarified the purpose of the px_rl_cont_b signals
0.7	1/11/02	David Liddle	Removed the read data pins and scan signals.
0.8	2/19/02	Khandaker Azad	Insertion of gclkw
0.9	05/07/03		Updated for LTE specification release.

61.1 Introduction

The Data RAM Arbiter Module (DRA) is designed to provide access to Y Data RAM for modules requiring direct memory access (DMA and VIAC). The MDI will have direct access to the X memory. This module works in the new S-OnyxU DSP memory architecture that is being developed to support higher operating speeds (133 MHz). Previously, the shared memory peripherals had access through dedicated ports on special shared memories. In the new architecture, these peripherals access the Data RAM via a single dedicated connection to each of the X and Y memory spaces. The MDI will have dedicated access to X memory, the other peripherals will be limited to accessing Y memory through the DRA. The DRA module arbitrates access to the Y-Data bus. See , "".

The fixed priority can be overridden if a peripheral access has timed out. This will only be overridden until the peripheral access is complete.

61.2 DRA Block Diagram

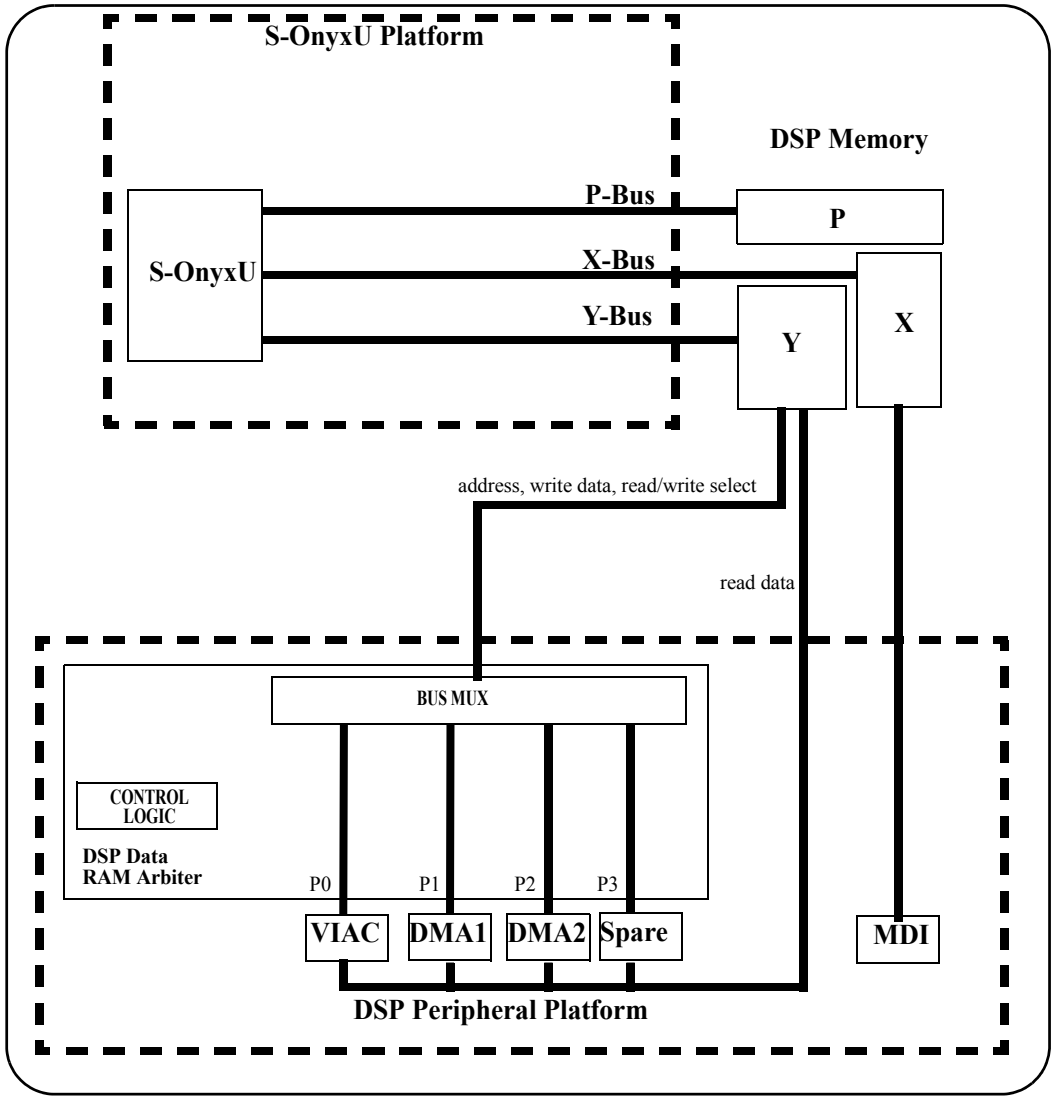


Figure 61-1. DRA Block Diagram

61.3 DRA Pin List

Table 61-1. DRA Module Pin List

Pin Name	Direction	Description
Misc.		
gclkw	Input	Global clock for DRA module
gclkw_b	Input	Inverted global clock
res	Input	Peripheral Module's Hardware reset.
Y Data Memory Interface		
dra_ydb_wr[15:0]	Output	Y Data Write data bus
dra_yab [15: 0]	Output	Y Data address bus
dra_aryrd	Output	Y Data read enable
dra_arywr	Output	Y Data write enable
dmem_y_rl_cont_b	Input	Asserted when a contention occurs in the Y memory space between the DRA and the core.
Port0 Memory Interface (VIAC)		
p0_ydb_wr[15:0]	Input	Port0 write data bus
p0_yab [15: 0]	Input	Port0 address
p0_aryrd	Input	Port0 read control
p0_arywr	Input	Port0 write control
p0_to_b	Input	Port0 timeout has occurred.
p0_rl_cont_b	Output	Asserted when Port0 access has not yet completed (due to contention with core or higher priority peripheral, or the 1 cycle delay due to data sampling).
Port1 Memory Interface (DMA1)		
p1_ydb_wr[15:0]	Input	Port1 write data bus
p1_yab [15: 0]	Input	Port1 address
p1_aryrd	Input	Port1 read control
p1_arywr	Input	Port1 write control
p1_to_b	Input	Port1 timeout has occurred.
p1_rl_cont_b	Output	Asserted when Port1 access has not yet completed (due to contention with core or higher priority peripheral, or the 1 cycle delay due to data sampling).
Port2 Memory Interface (DMA2)		
p2_ydb_wr[15:0]	Input	Port2 write data bus

Table 61-1. DRA Module Pin List

Pin Name	Direction	Description
p2_yab [15: 0]	Input	Port2 address
p2_aryrd	Input	Port2 read control
p2_arywr	Input	Port2 write control
p2_to_b	Input	Port2 timeout has occurred.
p2_rl_cont_b	Output	Asserted when Port2 access has not yet completed (due to contention with core or higher priority peripheral, or the 1 cycle delay due to data sampling).
Port3 Memory Interface (Spare)		
p3_ydb_wr[15:0]	Input	Port3 write data bus
p3_yab [15: 0]	Input	Port3 address
p3_aryrd	Input	Port3 read control
p3_arywr	Input	Port3 write control
p3_to_b	Input	Port3 timeout has occurred.
p3_rl_cont_b	Output	Asserted when Port3 access has not yet completed (due to contention with core or higher priority peripheral, or the 1 cycle delay due to data sampling).

61.4 Arbitration Scheme

The arbitration scheme is based on a fixed priority with Port0 first, Port1 second, Port2 third, and Port3 last. If a peripheral access has timed out, it should generate an interrupt to the S-OnyxU core to free the bus, allowing its access to occur. When a peripheral has timed out, it will be given the highest priority allowing its access to occur first. If multiple peripherals have timed out, then the priority will be according to the fixed priority.

The minimum latency in the DRA module is one cycle. The maximum will depend on whether there is contention with the core or with higher priority peripherals.

61.5 Low Power Mode

The DRA will not be receiving a clock when the S-OnyxU DSP core is in stop mode. This restricts the peripherals that can be connected to the DRA to those peripherals that are inactive in stop mode. It is the peripherals responsibility to ensure that no accesses are started on the cycle before stop mode is entered. If an access is started on the cycle before stop mode is entered, it will pause during stop mode (since none of the registers within the DRA will be updated), and will continue once stop mode is exited.

Chapter 62
Memory Separation Unit (MSU)

Table 62-1. Revision History

Revision	Date	Author	Comments
1.0	12 April 2001	Glen Zoerner	Initial release
1.1	18 April 2001	Glen Zoerner WITC-Austin	Remove DMA Add power saving during Privileged mode Add programming example Add connection diagram and functional diagram Change Memory Size from 5 bits to 28 bits
1.2	23 April 2001	Glen Zoerner WITC-Austin	Updated spec per software team's comments
1.3	24 April 2001	Glen Zoerner WITC-Austin	Reset change bars Add ID register remove Status Bit Separate comparator enables from attributes
1.4	26 April 2001	Glen Zoerner WITC-Austin	Reset change bars text clarifications
1.5	3 May 2001	Glen Zoerner WITC-Austin	Reset change bars add LDCL instruction support add ENABLE input signal
1.6	8 May 2001	Glen Zoerner WITC-Austin	Reset change bars add instruction details
1.7	16 May 2001	Mike Fitzsimmons EPS-Austin	Reset change bars. Modify to make R-AHB slave and per SW feedback.
1.8	17 May 2001	Mike Fitzsimmons	Minor clarifications.
1.9	21 May 2001	Mike Fitzsimmons EPS-Austin	Reset change bars. Clarifications and optimizations per feedback from SW team.
2.0	21 May 2001	Mike Fitzsimmons	Reset change bars. Minor corrections.
2.1	31 May 2001	Mike Fitzsimmons	Reset change bars. Minor clarifications.
2.2	4 June 2001	Mike Fitzsimmons	Reset change bars. Corrected Table 1.3-1.
2.3	15 June 2001	Mike Fitzsimmons	Add disable feature and simple blk diag. Clarified abort handling in FAR/FSR. Added "MSU Caveats" section.
2.4	30 July 2001	Mike Fitzsimmons	Corrected bit number in FSR. Added alignment note to Section 1.2.7 Added explanation of address Masking. Added Note to MSU Caveats Section 1.3

Table 62-1. Revision History

Revision	Date	Author	Comments
2.5	17 August 2001	Mike Fitzsimmons	Changed disable_msu input to enable_msu
2.6	25 March 2002	Mike Fitzsimmons	Updated to support 8 Areas.
2.7	7 June 2002	Russ Oertel	Fixed typos
2.8	05/07/03		Updated for LTE specification release.

Comments, corrections, and enhancements should be directed to:

Mike Fitzsimmons: Mike.Fitzsimmons@Motorola.com or RA2164@email

62.1 Introduction

The Memory Separation Unit (MSU) is a hardware mechanism to keep software user-mode tasks physically separated in memory space.

Features of the MSU:

- Fully synthesizable, pos-edge clock, mux-D design
- Implemented as R-AHB slave device (within existing AHBMUX module)
- Supports 8 programmable memory areas
- 1KB address granularity
- Generation of ABORT on disallowed memory access
- Automatic power-down during Privileged execution
- Unused comparators can be disabled

62.2 Operation

The MSU grants or denies access to user-mode application programs for up to eight areas in memory by comparing the current address and attributes on the ARM's bus. Base Address, Size and Permissions are selectable for each memory area. Each active application has an operating system-maintained map (data structure) of memory areas, each with separate privilege attributes of Read, Write and Execute. Just prior to an application being granted "running" status by the operating system, the application's structure of memory areas and permissions are loaded into the MSU and bounds/attribute checking is enabled. User-mode applications must make memory accesses only to their allowed spaces. Accesses beyond these spaces are terminated, before completing, with an ABORT. Privilege mode applications are granted access to the complete memory map.

A simple block diagram of the MSU is shown in Figure 62-1 on page 62-3.

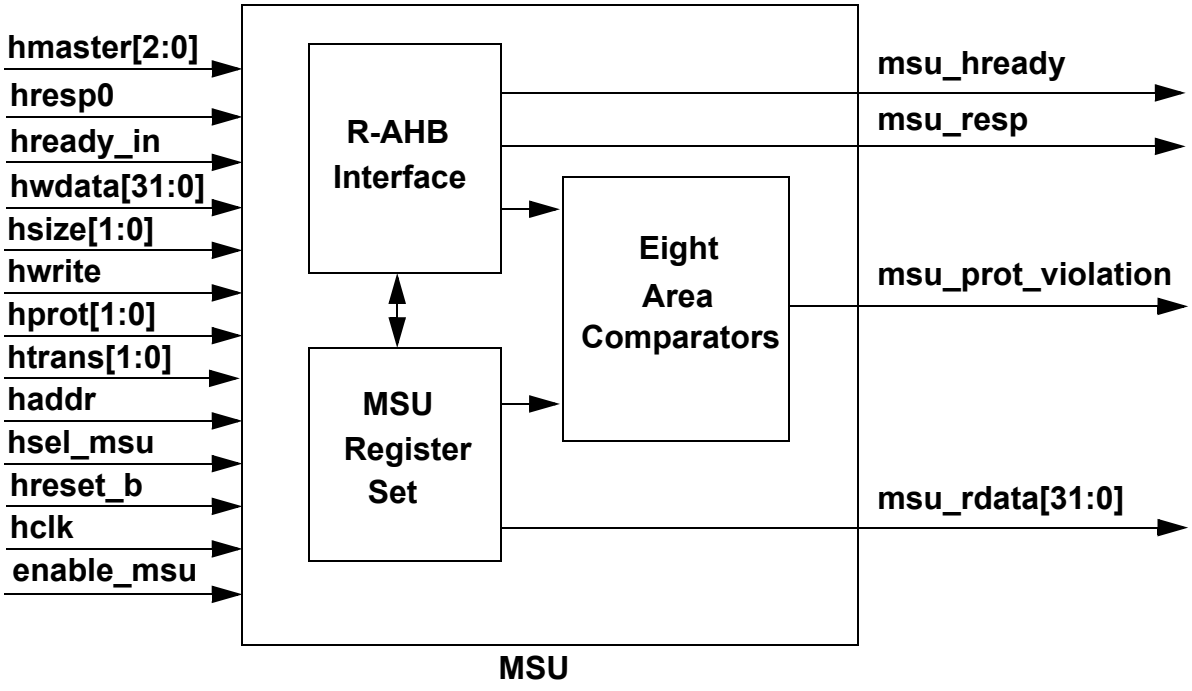


Figure 62-1. MSU Block Diagram

The MSU is implemented as an R-AHB slave device within the AHBMUX module. The ARM7 processor will program the MSU register set via an R-AHB slave interface. The register set supports 8 Areas for permission checking. Each area has a comparator which can generate a protection violation based on the programmed register permissions and the R-AHB bus cycle type.

A block diagram of a single comparator within the MSU is show in Figure 62-2 on page 62-4.

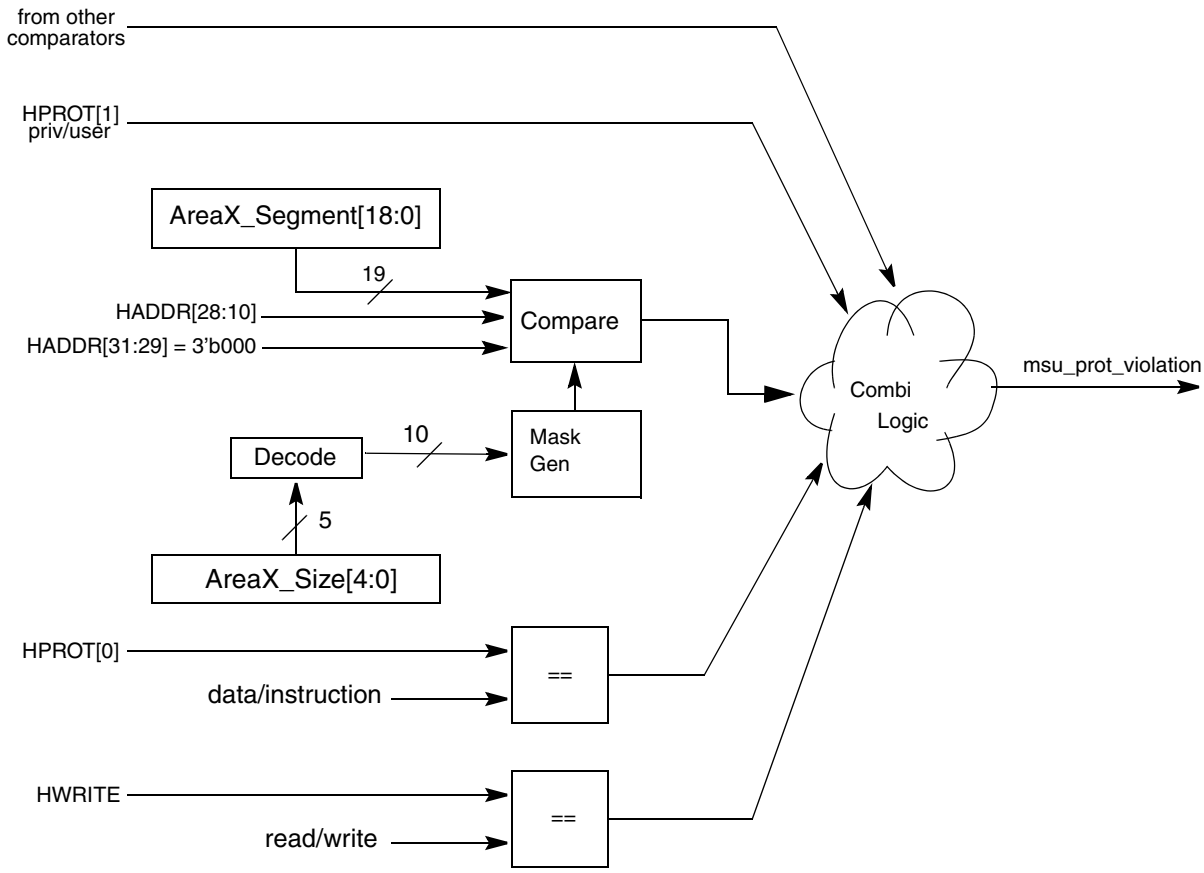


Figure 62-2. MSU Comparator (one of eight)

62.2.1 Disabling the MSU

62.2.1.1 Complete and Total Hardware Disable

An **enable_msu** input is provided to the MSU from an external fuse. When this signal is negated (low), the MSU will not respond to R-AHB bus cycles and will not cause any bus cycles to be aborted.

62.2.1.2 Disabling Abort Generation

When the **enable_msu** input is asserted (high), the MSU can be effectively disabled from generating aborts in either of three ways:

- Asserting system reset (**hreset_b**) or by
- Writing a "1" to ALL valid bit locations in the Clear Enable Register or by
- Writing a "0" to ALL valid bit locations in the Area Enable Register.

62.2.2 Enabling the MSU

Immediately after system reset, the MSU will be effectively disabled from generating aborts since the registers have not yet been programmed to enable protection checking. If the **enable_msu** input signal is asserted, the MSU can be enabled to abort bus transactions in either of three ways:

- Writing to any Area Control register automatically sets the corresponding bit in the Area Enable Register, which enables the corresponding area and the MSU.
- Writing a “1” to a valid bit location in the Set Enable Register automatically enables the corresponding Area and the MSU.
- Setting any valid bit in the Area Enable Register will enable the MSU.

When the MSU is enabled, unallowed bus transactions will result in the assertion of the **msu_prot_violation** output.

62.2.3 The msu_prot_violation Output

When asserted, the **msu_prot_violation** signal will gate off the **hsel_ram**, **hsel_rom**, **hsel_pram**, and the **hsel_cs[5:0]** signals in the AHBMUX module. In addition, the MSU will be responsible for asserting the **msu_hready** and **msu_hresp** signals in order to abort the transaction. That is, unallowed accesses will be aborted by the MSU, and not by the targeted slave device.

62.2.4 Unassigned Memory Area Behavior

If a user-mode access is made to an area that is not explicitly defined by entries in the MSU’s registers, and the MSU is enabled, the access is denied with an abort.

62.2.5 Privileged Mode Operation (Automatic Power-down)

Privileged state programs have complete access to the entire memory map. While HPROT[1] (Privileged/User attribute) indicates a privileged mode access, the comparators in the MSU are gated off to reduce power consumption. When a user-mode bus master resumes control, the signals are again allowed to propagate into the MSU’s logic.

62.3 MSU Registers

This section describes the registers associated with the MSU. All MSU registers are accessible in supervisor mode only. All registers in the MSU are accessed via 32-bit transactions only. The registers are mapped onto the R-AHB bus starting at address $2C00_0000. The address map for the MSU register set is shown in Table 62-2.

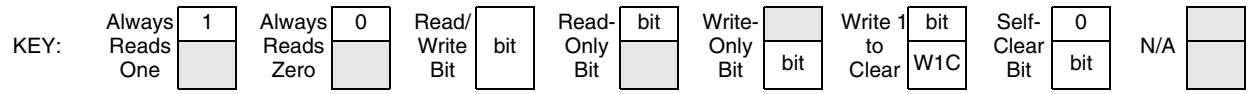


Table 62-2. MSU Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID $2C8C_0000	R	0	0	0	0	0	0	Largest Area Size[4:0]					Smallest Memory Area[4:0]				
	W																
	R	0	0	0	Number of Memory Areas[4:0]					Version Number[2:0]			Revision Number[4:0]				
	W																
Set Enable $2C8C_0004	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	Set Enable[7:0]							
	W																
Clear Enable $2C8C_0008	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	Clear Enable[7:0]							
	W																
Faul Address $2C8C_000C	R	Fault Address[31:16]															
	W																
	R	Fault Address[15:0]															
	W																
Fault Status $2C8C_0010	R	0	0	0	0	0	0	0	0	0	Prefetch_Abort_Status[22:16]						
	W																
	R	0	0	0	Data_Abort_Status[12:0]												
	W																
Area Enable $2C8C_0030	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	Area Enable[7:0]							
	W																
Area n* Control $2C8C_0038-$2C8C_0050	R	0	0	0	0	0	0	Area n Segment[18:8]									
	W																
	R	Area nSegment[7:0]								Area n Size[4:0]					Execute	Read	Write
	W																

*n = 0 to 7

Note: All registers are 32-bit access only. User mode and non 32-bit accesses to MSU registers will have no affect and will be aborted. Accesses to **Reserved** registers within the MSU register space will have no affect and will not be aborted.

62.3.1 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the MSU registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

62.3.2 ID Register

The MSU contains an Identification Register. The ID register is read-only and contains information about the MSU's capabilities, including the number of Areas, the size of the smallest Area, the size of the largest Area, and MSU module revision/version numbers. Writes to this register will be ignored but not aborted. Unused bits read 0. For Neptune LTE, the ID register is 32'h0249_0800.

ID ID Register **Addr $2C8C_0000**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
							Largest Area Size[4:0]					Smallest Memory Area[4:0]				
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
				Number of Memory Areas[4:0]					Version Number[2:0]			Revision Number[4:0]				
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 62-3. ID Register Description

Name	Description	Settings
Largest Area Size Bits 25-21	**Largest Area Size —** The size of the largest memory Area supported by the MSU. Refer to Table 1.2-5 on page 1-10 for size encoding. **Note:** The Neptune LTE implementation of the MSU will support a Largest Memory Area of 512 KB (0x12).	
Smallest Area Size[4:0] Bits 20-16	**Smallest Area Size[4:0]** - The size of the smallest memory Area supported by the MSU. Refer to Table 62-11 on page 62-16 for size encoding. **Note:** The Neptune LTE implementation of the MSU will support a Smallest Memory Area of 1K bytes (0x09).	
Number of Memory Areas[4:0] Bits 12-8	**Number of Memory Areas[4:0] -**This field indicates the number of memory areas that are supported in the MSU. The Neptune LTE implementation of the MSU will support 8 Areas (0x08).	
Version Number[2:0] Bits 4-0	**Version Number[2:0] -**This field indicates the version number of the MSU implementation. The initial implementation version number will be 0x00.	

62.3.3 Set Enable Register

Set Enable Set Enable Register **Addr $2C8C_0004**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									Set Enable[7:0]							
TYPE	r	r	r	r	r	r	r	r	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 62-4. Set Enable Register Description

Name	Description	Settings
Set Enable[7:0] Bits 7-0	This register is used to set the MSU Area enables. Writing a “1” to an Area’s Set Enable bit will set the corresponding bit in the Area Enable Register ($2C00_0030). Writing a “0” to an Area’s Set Enable bit will have no effect on that enable. This register will return all zeros when read. The actual value of all Area Enable bits can be determined by reading the Area Enable Register.	

62.3.4 Clear Enable Register

Clear Enable Clear Enable Register **Addr $2C8C_0008**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									Clear Enable[7:0]							
TYPE	r	r	r	r	r	r	r	r	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 62-5. Clear Enable Register Description

Name	Description	Settings
Clear Enable Bits 7-0	**Clear Enable —** This register is used to clear the MSU Area enables. Writing a “1” to an Area’s Clear Enable bit will clear the corresponding bit in the Area Enable Register ($2C00_0030). Writing a “0” to an Area’s Clear Enable bit will have no effect on that enable. This register will return all zero’s when read. The actual value of all Area Enable bits can be determined by reading the Area Enable Register.	

62.3.5 Fault Address Register

Fault Address Fault Address Register **Addr $2C8C_000C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	Fault Address[31:16]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Fault Address[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 62-6. Fault Address Register Description

Name	Description	Settings
Fault Address Bits 25-21	**Fault Address —** This register will contain the address of the last data-side abort caused by any R-AHB bus cycle mastered by the ARM7 processor (whether the MSU generated the abort or not). The contents of this register will correspond to the contents of the lower 16-bits of the Fault Status Register. Refer to Section 62.3.6, “Fault Status Register.” This register is read only. It will be cleared at reset. Thereafter, it will only be loaded on data-side aborts. Writes to this register will be ignored, but will not be aborted.	

62.3.6 Fault Status Register

This register's lower 16 bits will contain status information pertaining to the last data-side abort caused by any R-AHB bus cycle mastered by the ARM7 processor (whether the MSU generated the data abort or not). The contents of this register's lower 16-bits will correspond to the address contained in the Fault Address Register. Refer to Section 62.3.5, "Fault Address Register."

The upper 16 bits of this register, Fault Status Register bits [22:16], will provide information regarding prefetch aborts caused by any R-AHB bus cycle mastered by the ARM7 processor (whether the MSU generated the prefetch abort or not). Specifically, bit[16] is used to determine if the last prefetch abort was or was not generated by the MSU. If the last prefetch abort was indeed generated by the MSU, bits [22:17] then indicate which defined Area the aborted address resided in. Alternatively, the aborted address may reside outside any defined Areas (a "Null" address aborted). If the prefetch abort was not generated by the MSU, bits[22:17] can be considered "don't care".

This register is read-only. It will be cleared on reset. Thereafter, the lower 16 bits will only be loaded on data-side aborts. The upper 16 bits will only update on prefetch aborts. Writes to this register will be ignored but not aborted.

Fault Status Fault Status Register **Addr $2C8C_0010**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
										Prefetch_Abort_Status[22:16]						
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
				Data_Abort_Status[12:0]												
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 62-7. Fault Status Register Description

Name	Description	Settings
Pretech Abort Status Bits 22-16	**Pretech Abort Status —** These bits correspond to the last prefetch abort. These bits have no relationship to the Fault Address Register. The MSU will need an ARM7TDMI-S pipeline follower to insure only aborts generated by the msu_prot_violation signal set Prefetch_Abort_Status[22].	
Data Abort Status Bits 12-0	**Data Abort Status —** These bits are the address attributes corresponding to the last data-side aborted cycle caused by the MSU. These bits correspond with the address in the Fault Address Register.	

The Fault Status Register bit definitions are given in Table 62-8

Table 62-8. Fault Status Bit Encoding

Fault Status Bit	Description
Data_Abort_Status[0]	MSU Data Abort Status Bit 0 - MSU did NOT generate last data abort 1 - MSU did generate the last data abort
Data_Abort_Status[1]	Write=1, Read=0
Data_Abort_Status[2]	Mode: Priveledged=1, User=0
Data_Abort_Status[4:3]	SIZE: 00 - Byte 01 - Halfword 10 - Word 11 - Reserved
Data_Abort_Status[6:5]	TRANS CODE: 00 - Internal Cycle 01 - Co-processor Cycle 10 - Non-sequential Cycle 11 - Sequential Cycle
Data_Abort_Status[7]	Null Area Data Abort 0 - Data Abort within defined Area(s) 1 - Data Abort outside of defined Area(s)
Data_Abort_Status[12:8]	MSU AREA ABORTED (DATA) 5'h00 - Area 0 5'h01 - Area 1 5'h02 - Area 2 . . 5'h1F - Area 31
[15:13]	RESERVED
Prefetch_Abort_Status[16]	MSU Prefetch Abort Status Bit 0 - MSU did NOT generate last prefetch abort 1 - MSU did generate the last prefetch abort
Prefetch_Abort_Status[17]	Null Area Pre-fetch Abort 0 - Prefetch Abort within defined Area(s) 1 - Prefetch Abort outside of defined Area(s)
Prefetch_Abort_Status[22:18]	MSU AREA ABORTED (PRE-FETCH) 5'h00 - Area 0 5'h01 - Area 1 5'h02 - Area 2 . . 5'h1F - Area 31
[31:23]	RESERVED

62.3.7 Area Enable Register

This read/write register contains the Area Enables for the MSU. If none of the Area Enables are set, the MSU is considered disabled and will not abort any bus transactions. This register will be cleared on system reset.

Area Enable Area Enable Register **Addr $2C8C_0030**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									Area Enable[7:0]							
TYPE	r	r	r	r	r	r	r	r	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 62-9. Area Enable Register Description

Name	Description	Settings
Area Enable Bits 7-0	**Area Enable —** The Area Enable Register bits can be set with accesses to the Set Enable Register or cleared with accesses to the Clear Enable Register. The Set Enable Register and Clear Enable Register are provided for fast context switching. That is, it is not necessary to perform read-modify-writes to the Area Enable Register to change only one bit in the register.	

62.3.8 Area Control Registers

The following table describes the individual Area control registers. All Area Control Registers are cleared to 0x0000_0000 on system reset. These registers are 32-bit read/write.

Note: Unpredictable behavior will result if any of the enabled Areas overlap.

Note: You must align an Area's base address (Area "X" Segment) to that Area's size boundary (Area X Size). Unpredictable behavior will result if this is not done.

Area Control n[1] Area Control Registers **Addr $2C8C_0038-$2C8C_0050**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
						Area n Segment[18:8]										
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Area n Segment[7:0]								Area n Size[4:0]					Execute	Read	Write
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. For n = 0 to 7

Table 62-10. Area Control Register Description

Name	Description	Settings
Area n Segment Bits 18:0	**Area n Segment —** This field sets the base address for a specific memory Area. Segment[18:0] corresponds to HADDR[28:10]. HADDR[31:29] will be compared to 3'b000.	
Area n Size Bits 15-8	**Area n Size –-** This field sets the size for a specific memory Area. The encoding is shown inTable 62-11 and is compatible with the ARM940T from 0x09 upward.	
Execute Bit 2	**Execute –-**	1 = this memory area allows user-mode instruction fetches 0 = this memory area denies user-mode instruction fetches
Read Bit 1	**Read –-**	1 = this memory area allows user-mode read access 0 = this memory area denies user-mode read access
Write Bit 0	**Write** —	1 = this memory area allows user-mode write access 0 = this memory area denies user-mode write access

Table 62-11. Area Size Encoding

AreaX_Size[4:0]	Area Size	Supported
0x00 - 0x05	Reserved	No
0x06	128 bytes	No
0x07	256 bytes	No
0x08	512 bytes	No
0x09	1 KB	Yes
0x0A	2 KB	Yes
0x0B	4 KB	Yes
0x0C	8 KB	Yes
0x0D	16 KB	Yes
0x0E	32 KB	Yes
0x0F	64 KB	Yes
0x10	128 KB	Yes
0x11	256 KB	Yes
0x12	512 KB	Yes
0x13	1 MB	Yes
0x14	2 MB	No
0x15	4 MB	No
0x16	8 MB	No
0x17	16 MB	No
0x18	32 MB	No
0x19	64 MB	No
0x1A	128 MB	No
0x1B	256 MB	No
0x1C	512 MB	No
0x1D	1 GB	No
0x1E	2 GB	No
0x1F	4 GB	No

Note: Only values 0x09 -0x13 (1KB - 1 MB) are supported on the initial design.

62.3.8.1 Discussion of MSU Address Compare Logic

As mentioned previously, HADDR[31:29] must be 3'b0 for the **msu_prot_violation** signal to assert. The Area_Segment and Area_Size registers then determine the compare logic performed on HADDR[28:10].

Table 62-12. "HADDR - Address Segment & Size Alignment" will help visualize the use of the Segment and Size Fields. Since the smallest area granularity supported by the MSU is 1KB, HADDR[9:0] are never compared. Since the largest area granularity is 1MB, HADDR[28:20] are always compared against the programmed value of the Area_Segment[18:10] (i.e. the base address). The comparison of HADDR[19:10] are dependent upon the programmed value in the Area_Size register.

Table 62-12. HADDR - Address Segment & Size Alignment

HADDR	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9:0
AREA SEGMENT[18:0]				18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
MASK[9:0]													9	8	7	6	5	4	3	2	1	0	

The Mask[9:0] signals shown in the above Table are internal to the design and are generated from the Area_Size register. When a Mask bit is set, no comparison will be done on the corresponding HADDR bit. Table 62-13. "Mask[9:0] Encoding" shows the Mask[9:0] encoding as a function of the Area_Size register values supported in the arm7_platform.

Table 62-13. Mask[9:0] Encoding

AreaX_Size[4:0]	Area Size	Mask[9:0]
0x09	1 KB	10'h00_0000_0000
0x0A	2 KB	10'h00_0000_0001
0x0B	4 KB	10'h00_0000_0011
0x0C	8 KB	10'h00_0000_0111
0x0D	16 KB	10'h00_0000_1111
0x0E	32 KB	10'h00_0001_1111
0x0F	64 KB	10'h00_0011_1111
0x10	128 KB	10'h00_0111_1111
0x11	256 KB	10'h00_1111_1111
0x12	512 KB	10'h01_1111_1111
0x13	1 MB	10'h11_1111_1111

62.4 MSU Caveats

NOTE:

Due to the pipelined nature of the R-AHB bus, writes which set an Area Enable bit must be followed by at least one instruction prior to an attempted access into the MSU protected space. Reason: since the address for the cycle following the write cycle comes out prior to the actual completion of the data phase of the write cycle, improper operation can result.

NOTE:

At the present time, Neptune's DMA devices, the GEM and DMAC, can be configured to be privileged or user-mode bus masters. It is suggested that DMA devices be set up to be privileged bus masters.

NOTE:

The multiply mapped regions of RAM and ROM must only be accessed in Supervisor Mode. That is, the MSU does NOT decode the aliased memory regions.

NOTE:

The address decoding of the ROM and CS[0] space is dependent on whether or not the device boots internally or externally - the state of the **boot_int** signal. It is software's responsibility to program the Area_Segment[28] bit correctly for the two cases.

62.5 Memory Map Example

This example illustrates a possible memory configuration for a small application.

Stack Area:0x0002_0000 - 0x0002_0FFF (4K)Attributes: Read Write

Data Area:0x0001_0000 - 0x0001_FFFF (64K)Attributes: Read Write

Program Area:0x0000_1000 - 0x0000_1FFF (4K)Attributes: Read Execute (note)

System Area:0x0000_0000 - 0x0000_0FFF (4K)Attributes: No Access

Note: Code that contains PC-relative read-only data must have the Read attribute enabled.

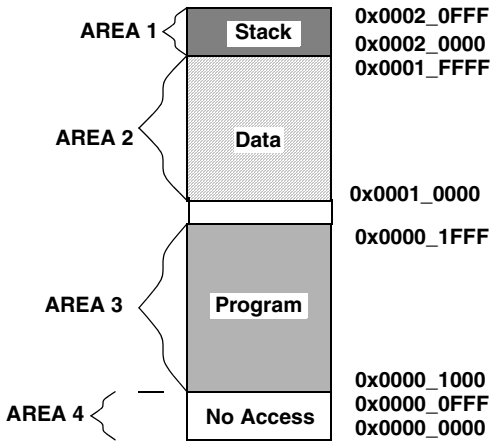


Figure 62-3. Example Memory Configuration

The following table shows the contents of the Area control registers associated with this memory configuration.

Table 62-14. Example Area Control Register Programming

Area Control Register #	Area Segment[18:0]	Permission	Area Size	Area Control Register[31:0]
0 (data)	0x0_0040	Read Write	0x0F (64 KB)	0x0000_407B
1 (stack)	0x0_0080	Read Write	0x0B (4 KB)	0x0000_805B
2 (program)	0x0_0004	Read Execute	0x0B (4 KB)	0x0000_045E
3 (no access)	0x0_0000	No Access	0x0B (4KB)	0x0000_0058
4		Disabled		0x0000_0000
5		Disabled		0x0000_0000
6		Disabled		0x0000_0000
7		Disabled		0x0000_0000

Chapter 63
Security Controller (SCC)

Revision History

Version	Date	Author	Notes
1.0mlb	October 10, 2001	Tom Tkacik	Combine Secure RAM and Security Monitor documents, and use MLB bus signals
1.1mlb	October 17, 2001	Greg Schmidt	Added clarification for Security Monitor
1.2mlb	October 31, 2001	Tom Tkacik	Add LaserID test, and security table
1.3mlb	November 21, 2001	Tom Tkacik	Corrected address map Include Patriot INDY external pin names
1.4ahb	December 12, 2001	Mike Oliver	Changed for use of AHB bus signals
1.5	May 17, 2002	Scott King, Shannon Osgood	Formatted registers per spec review 05/13/02.
1.6	June 13, 2002	Scott King, Shannon Osgood	Updated base address variable from BA_SCC to BA_SCC_MEM and BA_SCC_MON.
1.7	05/07/03		Updated for LTE specification release.

63.1 Introduction

The Security Controller is conposed of two blocks, the Secure RAM module and the Security Monitor. The Secure RAM provides a way of securely storing sensitive information both in on-chip RAM, and in off-chip non-volatile memory. On-chip, the data is stored in RAM that can be cleared, if necessary, to prevent un-authorized access. Off-chip, the data is stored in encrypted form, using an encryption key that is unique to each device and accessible only to the Secure RAM module.

The Secure RAM module will perform the encryption, and then the host processor can move the encrypted data off-chip.

The Secure RAM module and the Security Monitor (with Debug Detector), provide the initial seed for a secure platform. These added blocks cannot provide the entire security solution. There is a software component which is equally important. This document will concentrate on a hardware implementation of the Security Controller, meeting the requirements given in the *Rainbow/Patriot/Neptune Secure Memory, Mini-Monitor, and Debug Detector Requirements* document.

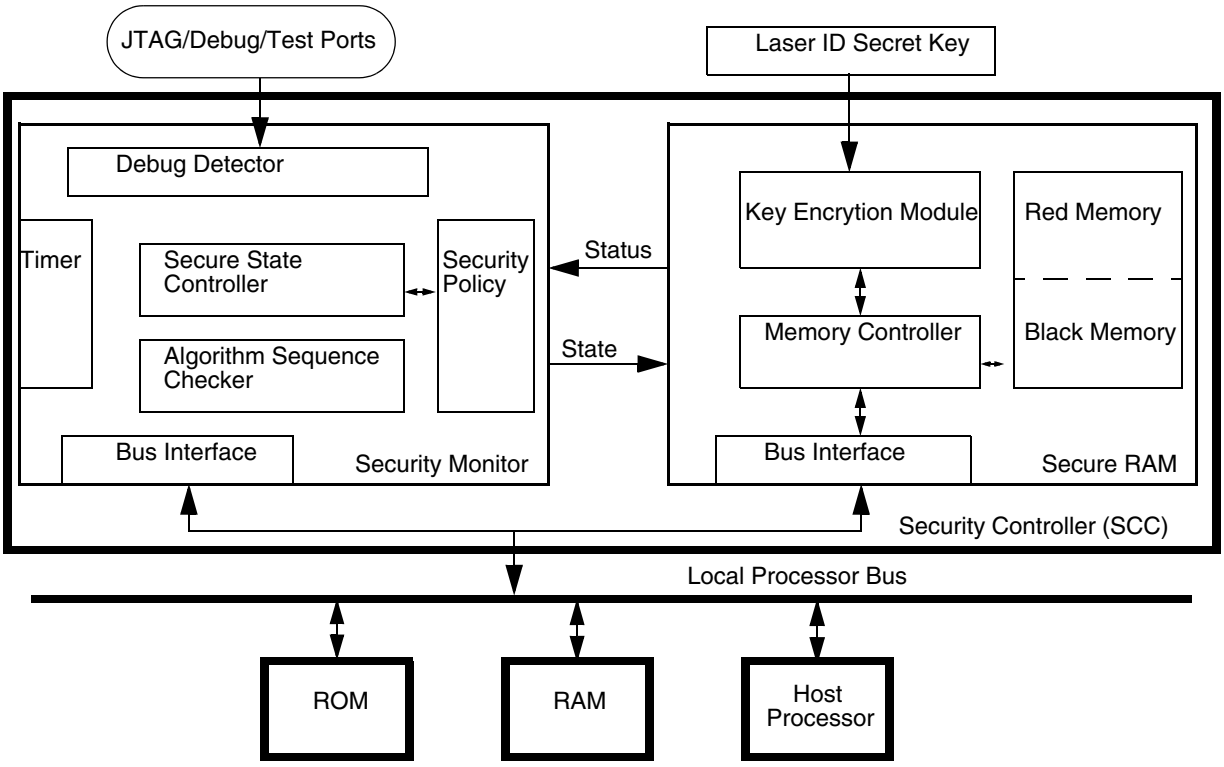


Figure 63-1.

63.2 External Pins

The following are the inputs and outputs to the Security Controller. These are internal signal names, as the external names will be dependent on which local bus the Secure RAM is connected to.

Table 63-1. SCC Pin List

Signal	Direction	Connects to	Signal Description
h_clk	input	Bus Interface	Master clock
hreset_b	input	Bus Interface	Master reset
scan_reset_b	input	Scan Wrapper	Reset when scan mode is first entered or with master reset
hwdata[31:0]	input	Bus Interface	Data in bus
scc_hrdata[31:0]	output	Bus Interface	Data out bus
haddr[11:0]	input	Bus Interface	Address bus
hsel_scm	input	Bus Interface	Module select for Secure RAM
hsel_smn	input	Bus Interface	Module select for Security Monitor

Table 63-1. SCC Pin List (Continued)

Signal	Direction	Connects to	Signal Description
hwrite	input	Bus Interface	Read/write from/to memory
hprot[1:0]	input	Bus Interface	Processor Protection Control Indicators
hsize[1:0]	input	Bus Interface	Transfer Size
htrans[1:0]	input	Bus Interface	Transfer Type
hready_in	inupt	Bus Interface	AHB hready (from EIM)
scm_hready	output	Secure RAM	Transfer Acknowledge from Secure RAM
scm_hresp0	output	Secure RAM	Transfer Error Acknowledge from Secure RAM
smn_hready	output	Security Monitor	Transfer Acknowledge from Security Monitor
smn_hresp0	output	Security Monitor	Transfer Error Acknowledge from Security Monitor
scc_scm_irq_b	output	Secure RAM	Interrupt from Secure RAM
scc_smn_irq_b	output	Security Monitor	Interrupt from Security Monitor
scan_mode	input	Test Controller	Module is in scan mode
scan_exit	input	Test Controller	Scan mode has been exited
ext_laser_id[167:0]	input	Laser Fuse	168 bit secret key (unique for each device)
int_boot	input	Platform Input	Boot from internal ROM
iim_scc_en	input	IIM	Enable SCC
debug1	input	Platform Input	A monitored debug/test signal
debug2	input	Platform Input	A monitored debug/test signal
debug3	input	Platform Input	A monitored debug/test signal
debug4	input	Platform Input	A monitored debug/test signal
debug5	input	Platform Input	A monitored debug/test signal
ipt_test_membist_sel	input	Bist Controller	Select memory bist mode
ipt_test_membist_invoke	input	Bist Controller	Start selected bist test
ipt_test_membist_reset	input	Bist Controller	Reset the bist logic
ipt_test_membist_fail	output	Bist Controller	A bist failure was detected
ipt_test_membist_done	output	Bist Controller	Bist test finished
ipt_test_membist_bitmap_out	output	Bist Controller	Memory bist data

H_CLK

This is the clock input. All registers are clocked off of this clock signal.

HRESET_B

This input is the master hardware reset. All internal registers will be reset, and the memory contents will be cleared. After reset is released, the memory will be cleared, and then the IRQ signal will be asserted. If an access is attempted while memory is being cleared, a bus error will result. The host should perform a software memory and hardware test after reset.

SCAN_RESET_B

This input must be asserted along with hreset_b. It also must assert for at least one clock cycle when scan test mode is first entered. This is used to clear all flops that might contain sensitive information, so that it cannot be scanned out.

HWDATA[31:0]

The data in bus connects to both the memory and the encryption module. If data is to be decrypted before writing to the memory, it will pass through the encryption module. Otherwise it will go directly to the memory. All data is written as 32 bit words.

SCC_HRDATA[31:0]

The data out bus is muxed from the memory data out and the encryption module data out. When data is not being read (and this output is not blocked), this output will be driven to all 0's. All data is read as 32 bit words.

HADDR[11:0]

This is the memory address bus. All Secure RAM addresses must be word aligned. Otherwise, and error will be generated.

HSEL_SCM

When this signal is asserted, then the Secure RAM module is selected, and will respond to the bus transaction.

HSEL_SMN

When this signal is asserted, then the Security Monitor is selected, and will respond to the bus transaction.

HWRITE

When hwrite is high, data will be written to the addressed memory location. When hwrite is low, data will be read from the addressed memory location.

HPROT[1:0]

This is a bus transaction signal, specifying the current bus transaction protection level. The protection levels include supervisor and user priveleges for instruction and data fetches.

HSIZE[1:0]

This hsize signals specify the size of the word being operated on during the current bus transaction.

HTRANS[1:0]

This htrans signals specify the type of the current bus transaction.

HREADY_IN

The hready_in specifies that the current bus transaction has completed.

SCM_HREADY

This indicates that the Secure RAM's bus transaction is complete.

SCM_HRESP0

This indicates that the Secure RAM's bus transaction had an error.

SMN_HREADY

This indicates that the Security Monitor's bus transaction is complete.

SMN_HRESP0

This indicates that the Security Monitor's bus transaction had an error.

SCC_SCM_IRQ_B

This is the interrupt request from the Secure RAM. It is a low priority interrupt.

SCC_SMN_IRQ_B

This is the interrupt request from the Security Monitor. It is a high priority interrupt.

SCAN_MODE

The module is in scan mode.This signal turns on some test circuits to bypass the RAM during scan testing, to improve scan testing.

SCAN_EXIT

The module has left scan mode. The Security Monitor will enter its Fail state.

EXT_LASER_ID

This is the 168 bit secret key. It should be unique for each device. It is programmed via one-time programmable laser fuses.

NOTE:

These signals *must* go directly from the source to the SCC inputs, and not through a scan wrapper, where they can be modified during scan testing. The SCC includes circuitry to improve test coverage of the internal circuits fed by these signals.

INT_BOOT

This signal indicates that the platform booted from internal ROM.

IIM_SCC_EN

This signal can be used to enable or disable the SCC. When disabled, the SCC will not respond to any bus request, and will not generate any interrupts. It will not exist on the bus. Iim_scc_en should be connected to a laser fuse.

DEBUG1 - DEBUG5

These signals should be connected to the appropriate signals on the device, which indicate that a test or debug mode is being entered. These are used by the Security Monitor to determine its current security state.

IPT_TEST_MEMBIST_SEL[2:0]

This input selects which mode the bist controller is in. The following table lists the various bist mode.

Table 63-2. Bist Modes

IPT_TEST_MEMBIST_SEL	Description
3'b000	Bist Disabled
3'b100	Production Mode
3'b101	Debug Mode
3'b110	Bitmap Mode
3'b111	Reserved

IPT_TEST_MEMBIST_INVOKE

The bist test will start when the ipt_test_membist_invoke signal is asserted. Before the bist test can be invoked, the Secure RAM must be allowed to complete its initialization of the memory. This takes about 1050 clock cycles, so after a reset, wait at least 1050 clock cycles before asserting ipt_test_membist_invoke.

RAM bist tests will not run properly if smn_block_access is asserted, or if the Secure RAM is initializing memory, encrypting data or has detected an internal hardware error. This includes a memory error, so the debug and bitmap modes will not work if the Secure RAM has detected a memory error.

IPT_TEST_MEMBIST_RESET

This signal resets the bist controller

IPT_TEST_MEMBIST_FAIL

This output indicates that a failure has been detected during the bist test.

IPT_TEST_MEMBIST_DONE

This output indicates that the bist test has finished.

IPT_TEST_MEMBIST_BITMAP_OUT[7:0]

This data output is used during bitmap mode, to read the contents of the memory, one byte at a time. This can be used to pinpoint the location of an error. Data cannot be read from the ipt_test_membist_bitmap_out port if smn_inhibit_key is not asserted (indicating the secure state), if smn_block_access is asserted, or if the Secure RAM is initializing memory, encrypting data or has detected an internal hardware error.

The memory BIST controller is only guarenteed to work when scc_enable is not asserted. This would be during production testing, before the laser fuses are configured. After scc_enable is asserted, the memory BIST may be blocked depending on the internal state of the Secure Memory. In particular, if the internal zeroize on reset test detects a memory failure, the production, debug and bitmap BIST modes will not function. Production testing of the internal memroy in the Secure Memory, must therefore take place before enabling the Secure Memory.

63.3 Secure RAM

The Secure RAM is one of the two blocks in the Security Controller (SCC) module. The other is the Security Monitor. The Secure RAM provides a way of securely storing sensitive information both in on-chip RAM, and in off-chip non-volatile memory. On-chip, the data is stored in RAM that can be cleared, if necessary, to prevent un-authorized access. Off-chip, the data is stored in encrypted form, using an encryption key that is unique to each device and accessible only to the Secure RAM module.

The Secure RAM module will perform the encryption, and then the host processor can move the encrypted data off-chip.

The Secure RAM module that is being defined in this document, together with the Security Monitor (with Debug Detector), provide the initial seed for a secure platform. These added blocks cannot provide the entire security solution. There is a software component which is equally important. This document will concentrate on a hardware implementation of the Secure Memory meeting the requirements given in the *Rainbow/Patriot/Neptune Secure Memory, Mini-Monitor, and Debug Detector Requirements* document.

The specified blocks to be added to a core processing device provide for the protection of sensitive data that should not exist outside of the host LSI. This sensitive data might include a user's social security number, credit card number, a Digital Rights Management (DRM) ID, or system data like an IMEI value. The following is a brief summary of both of the Secure RAM and the Security Monitor.

63.3.1 Secure RAM Overview

The Secure RAM is comprised of four sub-blocks to provide a safe environment for sensitive data when it is not being manipulated by the host. The three sub-blocks consist of a Laser ID, Key Encryption Module (KEM), a Zeroizable Memory, and a Memory Controller.

Laser ID - The Laser ID block consists of a number of fuse-like structures that can be configured during part fabrication. The number of Laser ID bits is based on the encryption algorithm implemented in the KEM, which for 3DES is 168 bits. The Laser ID value will be generated from a random number generator program and will be used as the master key for the KEM. After the Laser ID is programmed into each device, and verified, the random number must be discarded and there must be no correlation between the random numbers and any particular devices. The Laser ID block also contains a default key, which can be used to prevent unauthorized access to the encrypted data, or for test purposes.

Key Encryption Module - The KEM is used to protect sensitive data for storage off-chip. Sensitive data is entered into the host processor chip via user input or via various download modes. The data is encrypted by the KEM and then stored off chip. The KEM uses the unique Laser ID as a master key. Thus, any data encrypted using this key is of no value to any other host or cellular device since they will be unable to decrypt it. When the host processing device needs the data it will be read in from the external memory and decrypted by the KEM. Once decrypted, it will reside in the Secure RAM.

Zeroizable Memory - There are two types of data that will be stored in the Secure RAM module, sensitive information that needs to be secured, and non-sensitive (encrypted) information. The memory used to store the sensitive infomation is given the name Red memory. This is the data which we are trying to prevent unathorized access to. The memory holding the non-sensitive information is known as Black memory. This is encrypted information, and we do not have to restrict access to it. The zeroizable memory is currently defined to be 1k bytes of Red memory and 1k bytes of Black memory. The Black memory is used as a holding buffer for data that is to be decrypted, or data that has just been encrypted. All of the memory is implemented as a standard static RAM with a method to clear its contents in an automatic process. When the Zeroize signal becomes active, the memory address lines will be sequenced and an initial value will be loaded at each address, thus clearing the contents of both the Red and Black memory.

Memory Controller - The memory controller implements a DMA controller, memory clear function, and all of the user accessible control and status registers.

63.3.1.1 Use of the Secure RAM

There are two uses for the Secure RAM. The first is to store sensitive information on-chip. For this function, the Secure RAM provides a way of clearing that information if the Security Monitor recoginizes a security problem. When this occurs, the Security Monitor will cause the Secure RAM to clear all of its contents, and also to inform the host that this has occured. Also during power on reset, the secure memory will be cleared to make sure that there is not old sensitive data lying around ready to be read out.

The second use of the Secure RAM module is to prepare that sensitive information for external storage. To do this, the information is first loaded into the red (or sensitive) memory. Then a DMA controller within the Secure RAM is programmed to copy the sensitive information to the black (or non-sensitive) memory. During this copying, the data is first encrypted using a secret key. The host can then read the data from the black memory and store it safely off-chip. To recover the data, it is read from off-chip and loaded into the black memory. The DMA controller is then programmed to copy from black to red memory, and the data is decrypted on the way. The sensitive information now resides in the secure memory. The host can use it as needed.

63.3.1.2 Use of the Secure RAM in a non-secure application

The Seucre RAM module has been designed to be as non-intrusive as possible, while still maintaining its ability to provide security when needed. One design goal was that if security is not required, and the Secure RAM module are not accessed, then it will not interfere with the operation of the platform. This goal was mostly met.

The Secure RAM contains a 2K byte memory for storing sensitive data. If security is not required in a particular application, then that memory may be used for any other purpose. Although the memory may be used for storing other data, there are restrictions on that use.

After coming out of reset, the Secure RAM will initialize all of its memory, to prevent any sensitive data that was stored before the reset to remain. It takes about 1000 clock cycles to initialize and verify the 2k bytes of memory. During this initialization time, the memory will be unavailable. Any attempt to access the memory will result in an aborted bus cycle.

Several methods are available to determine when the initialization has completed. The first is simply to have the boot code do something else during that time, such that by the time that task is completed the Secure RAM is available. The second method is to poll a Secure RAM status register. The status register will be readable during this initialization time, and will indicate whether the memory initialization has completed. The third method is to enable the Secure RAM interrupt (at the platform interrupt controller). When the Secure RAM initialization is completed, it will generate an interrupt (which may be ignored by the interrupt controller).

After the initialization has completed, there are restrictions on accessing the memory, which also apply when storing sensitive information. Only a 32-bit, supervisor mode, aligned access is allowed. Any other type of access will result in an abort. Alternate bus masters may also access the memory, as long as they can perform a 32-bit, supervisor mode, aligned access.

Given these provisos, the data store in the Secure RAM memory will not be cleared by any security related event.

63.3.1.3 Security Monitor

The Security Monitor block is used to determine when and how Secure RAM resources will be used in the system. This block ensures that the system is running in such a manner as to provide protection for the sensitive data that is resident in the system. The specified Security Monitor is a simplified version of the

monitor used in Motorola's High Assurance Security Architecture products. As the need for assurance increases in the future, this monitor's functionality can be replaced with a monitor that checks for additional failure mechanisms.

The Security Monitor contains a state machine that controls the Secure RAM access. The starting state begins with the system reset or Power On Reset (POR). The Security Monitor will transition into one of three states: the Failure State if the Secure Memory fails to zeroize, the Non-Secure State if the processor boots off chip, or into Health Check 1 State if the processor boots from internal ROM and if the Secure RAM successfully zeroizes.

For more information on the Security Monitor, see 63.4.

63.3.2 Block Diagram

Figure 63-3 shows a functional block diagram of the Secure RAM module. To encrypt data for external storage, it will first be written to the Red memory, encrypted and then read back from the Black memory. To decrypt data from external storage, it will be written to the Black memory, decrypted and then read back from the Red Memory.

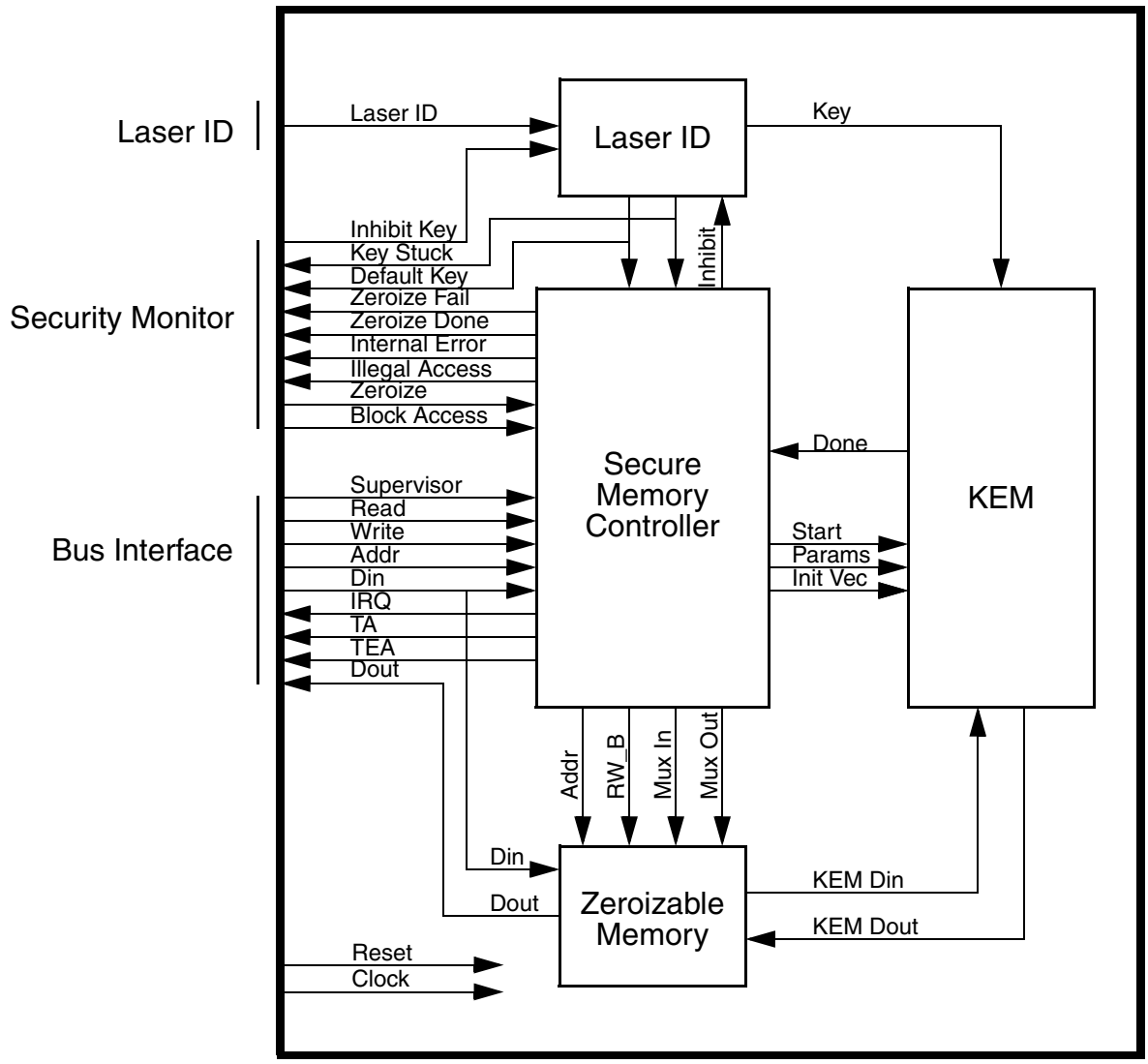


Figure 63-3. Secure RAM Block Diagram

63.3.2.1 Memory Controller

The external data bus is 32 bits, but the encryption algorithm requires 64 bits. There will be one physical 2K byte RAM configured as 512x32 bits. This will be partitioned into two logical 1K byte RAMs with possibly non-contiguous address spaces. The KEM will perform two reads or writes to access each block of data.

To clear the memory, a 9 bit counter is used to sequentially writes to all memory locations. This counter must NOT be part of any scan chain. It will need to be tested through functional testing. All memory will then be read to verify that the data has been properly cleared. If non-initialized data is found, an error signal will be sent to the Security Monitor.

The memory is 512 32-bit words, numbered from 0 to 511. During memory initialization each memory location is written with its address (0-511). The first Red memory location will get the value 0, the last Red memory location will get the value 255, the first Block memory location will get the value 256, and the last Black memory location will get the value 511. After all of the memory locations have been written, each

memory location will then be read to verify that it contains the proper initialization value. If any of the memory locations is not properly initialized, the SCM_ZERO_FAIL signal will be asserted, and the status register will reflect this error condition. Access to the memory will be denied until reset.

A BIST controller will also be included in the memory controller. It will be on the scan chain, and be accessible through the JTAG TAP controller.

63.3.2.2 Key Encryption Module

The data encryption module will use 3DES for the first implementations, and use a 168 bit key. The key will be laser scribed during manufacture, and will be accessible only to the encryption module. It will not be accessible on any bus outside of the secure memory module.

The 3DES module will take 48 clock cycles to encrypt/decrypt one 64 bit block of data (not counting the time required to read and write the data).

The Key Encryption Module will contain accessible registers for holding the Start Addresses, Length, Status and Control words. All accessible registers will be 32 bits.

There are two independent start addresses to index into the Red and Black memory. The status register will include a done signal which mimics the interrupt. The control register will include clear interrupt, mask interrupt, direction (either encrypt from secure memory to buffer memory, or decrypt from buffer memory to secure memory), and go (start processing) bits.

63.3.2.3 Bus Interface

The bus interface will be platform specific, and connects the internal bus to the platform system bus.

63.3.3 Internal Signal Pins

The following are the signals which go between the Secure RAM module and the Security Monitor.

Table 63-3. Internal Signal Pins

Signal	Direction	Connects to	Signal Description
smn_zeroize	input	Security Monitor	Reset encryption module and clear all memory
smn_inhibit_key	input	Security Monitor	Use the default key, not secret key
smn_block_access	input	Security Monitor	Block access to the secret key, KEM and memory
scm_illegal	output	Security Monitor	User mode access attempted.
scm_zero_done	output	Security Monitor	The memory has been cleared and verified
scm_zero_fail	output	Security Monitor	The memory did not verify as cleared.
scm_bad_key	output	Security Monitor	The secret key is stuck at the default key, inverse default key, or one of the defined DES weak keys.
scm_default_key	output	Security Monitor	The default key is being used (not secret key)
scm_internal_err	output	Security Monitor	The built-in assurance tests have detected a failure

SMN_ZEROIZE

This input from the Security Monitor will cause the secure memory to be cleared. This will be performed by sequentially writing an initial value to each of the memory's addresses.

The SMN_ZERIOIZE signal need be high for only a single cycle. If it is high while the memory is being cleared, it will be ignored. If it is still high when the memory has finished clearing, it will start to clear again.

SMN_INHIBIT_KEY

When asserted, this input from the Security Monitor causes the default key (currenly defined as 0x55555555555555_AAAAAAAAAAAAAA_33333333333333) to be used as the encryption key rather than the secret key. This allows for functional testing of the Secure RAM module using known answer tests, but does not allow access to information encrypted wiht the secret key.

The above default key is equivalent to the 192 bit 3DES key 0xFE01FE01FE01FE01_01FE01FE01FE01FE_FEFE0101FEFE0101. This key can be used for creating known answer tests using the default key.

SMN_BLOCK_ACCESS

When asserted, this input from the Security Monitor causes all writes to the zeroizable memory to be ignored and all reads to return 0, along with assertion of the TEA signal (with the exception of the STATUS and ERROR status registers, which can be read). The STATUS register can be read during a block_access, and will have a bit set indicating that SMN_BLOCK_ACCESS is currently asserted, and that access to the memory is denied.

SCM_ZERO_DONE

This output signals that the contents of the memory have been cleared, and the contents verified to contain the intial value. It will be deasserted when the zeroize operation begins, and will be asserted at the end of the operation. At the end of the operation the IRQ signal will also be asserted.

To provide assurance that memory was cleared properly, the initial value for each memory location will be different (the 9-bit address of each memory location).

The SCM_ZERO_DONE signal will be asserted whenever the memory is cleared, even if the Block_access signal is also asserted. It will not be latched into the status register if SMN_BLOCK_ACCESS prevents access from occuring when the zeroize completes.

SCM_ZERO_FAIL

This output signals the the verification step of zeroize operation has found a non-initial value in one of the memory locations. The memory contents could contain the sensitive data that was stored before the zeroize operation was executed. This should only happen if an internal error was encountered. The IRQ signal will not be asserted, and the STATUS register will continue to indicate that memory is not currently accessible, as well as that the zeroize operation failed.

This signal will be cleared at reset, or when another zeroize command is received.

SCM_BAD_KEY

There are two different keys used by the Secure RAM, a secret key and a default key. The default key is known to all and any data encrypted with it can be decrypted by all and is not secure.

The secret key is checked to make sure it is a valid key. There are nine code bits added to the key at manufacture, and these are checked to see that the key (along with the code bits) forms a proper code word. A Hamming code, capable of detecting all single, double and triple bit errros is used. There is also a set of DES keys which are considered weak keys. If the secret key contains any of these weak keys, is the default key or forms an invalid code word, it has likely been not programmed correctly, and the secret key cannot be used.

If the secret key is not valid or is weak, it is likely stuck, and bad_key will be asserted.

The 56-bit DES weak keys are 0x00000000000000, 0xFFFFFFFFFFFFFF, 0x0000000FFFFFFF and 0xFFFFFFF0000000.

SCM_DEFAULT_KEY

This output indicates that the key being used is the default key. If this signal does not indicate what the Security Monitor is requesting, it could be an indication of an internal error.

SCM_INTERNAL_ERROR

When asserted, this output indicates that an internal hardware error was detected. This checking includes verifying that the proper number of rounds were completed between an internal encryption start and done signal, memory initialization completed properly, and that the internal state machine never enters an undefined state. (This would be a check of the encryption state machine.) The key used by the encryption engine is compared against both the default key and the secret key. If the key is not the correct key, it is likely stuck. Under any of these circumstances, scm_internal_error will be asserted.

SCM_ILLEGAL

When asserted, this output indicates to the Security Monitor that a User mode access, byte or halfword access, unaligned access, or an access to a reserved address was attempted. The bus cycle will also be terminated with an bus error signal.

63.3.4 Register Summary

The following table shows the address map for the secure RAM module. All addresses are given in hex, starting from address 0 (the actual start address for the secure memory has not been specified). There is a total of 4K bytes of address space dedicated to the Secure RAM.

All accesses are (32 bit) word accesses. The “When Busy” column represents access when the Busy bit in the status register is asserted (block access, encrypting data, zeroizing, etc.)

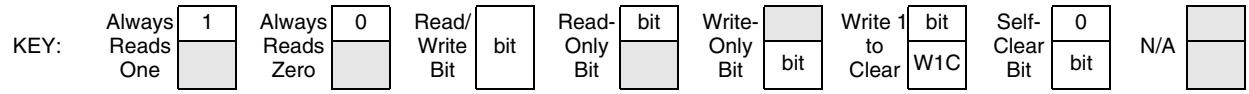


Table 63-4. SCC Secure RAM Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RED_START $2E8E_0000	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	RED_START[6:0]						
	W																
BLACK_START $2E8E_0004	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	BLACK_START[6:0]						
	W																
LENGTH $2E8E_0008	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	LENGTH6:0]						
	W																
CONTROL $2E8E_000C	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0						E/C	E/D
	W														START		
STATUS $2E8E_0010	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	ENCR FAIL	ACCESS ALLOWED	ENCRC OMP	CIR COMP	INT STATUS	SECRET KEY	INTERN ERROR	BAD KEY	CIR FAIL	BLOCK ACCESS	ENCR	CIR MEM	BUSY
	W																
ERROR $2E8E_0014	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	UNALIGN	BYTE	ILLEGAL ADDR	USER	DEAULT KEY	INTERN ERROR	BAD KEY	CIR FAIL	BLOCK ACCESS	ENCR	CIR MEM	BUSY
	W																
INT_MASK $2E8E_0018	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0			
	W														CLR MEM	CLR INT	IMASK
CONFIG $2E8E_001C	R	VERSION[4:0]					BLACK SIZE[9:0]										RED SIZE[9]
	W																
	R	RED SIZE[8:0]									BLOCK SIZE[6:0]						
	W																

Table 63-4. SCC Secure RAM Register Summary (Continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INIT_VECTOR_0 $2E8E_0020	R	INIT_VECTOR_0[31:16]															
	W																
	R	INIT_VECTOR_0[15:0]															
	W																
INIT_VECTOR_1 $2E8E_0024	R	INIT_VECTOR_1[31:16]															
	W																
	R	INIT_VECTOR_1[15:0]															
	W																

63.3.5 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the SCC registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

63.3.5.1 RED_START

This 7 bit address is the starting address in the Red memory where data should be read or written by the memory controller.

RED_START Red Memory Start Address **Addr $2E8E_0000**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
										RED_START[6:0]						
TYPE	r	r	r	r	r	r	r	r	r	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-5. RED_START Description

Name	Description	Settings
RED_START Bits 6 - 0	**RED_START —** The starting address is given in blocks, where a block is 64 bits (for 3DES) and can be read from the Configuration register. If the start address plus the length is greater than the size of red memory, a bit will be set in the STATUS register, and IRQ will be asserted immediately.	

63.3.5.2 BLACK_START

This 7 bit address is the starting address in the Black memory where data should be read or written by the memory controller.

BLACK_START Black Memory Start Address **Addr $2E8E_0004**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
										BLACK_START[6:0]						
TYPE	r	r	r	r	r	r	r	r	r	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-6. BLACK_START Description

Name	Description	Settings
BLACK_START Bits 6 - 0	**BLACK_START —** The starting address is given in blocks, where a block is 64 bits (for 3DES) and can be read from the Configuration register, and will. If the start address plus the length is greater than the size of black memory, a bit will be set in the STATUS register, and IRQ will be asserted immediately.	

63.3.5.3 Length

This 7 bit length is the number of blocks to be encrypted (or decrypted).

LENGTH Length Register **Addr $2E8E_0008**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
										LENGTH[6:0]						
TYPE	r	r	r	r	r	r	r	r	r	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-7. LENGTH Description

Name	Description	Settings
LENGTH Bits 31-20	**LENGTH —** A block is 64 bits (for 3DES) and can be read from the Configuration register. If the length is big enough to cause either the red or black memory address to wrap, no encryption will be performed, the interrupt signal will be asserted immediately, and the status register will indicate that the encryption failed. Note that the length is actually 1 less than the total number of blocks to encrypt (or decrypt). So, to encrypt a single block, the length register should contain 0, while to encrypt 128 blocks (512 bytes), the length register should contain 7f.	

63.3.5.4 Control

The control register contains the start encryption bit. Writing to this register will cause the encryption (or decryption) to begin. It also contains the direction bit, and the encryption mode bit (ECB or CBC).

CONTROL Control Register **Addr $2E8E_000C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
														START	E/C	E/D
TYPE	r	r	r	r	r	r	r	r	r	r/w	r/w	r/w	r/w	slfclr	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-8. Control Description

Name	Description	Settings
START Bit 2	**START** — Start Encryption Processing - a self-resetting bit	0 = no operation 1 = Start processing according to Start, Length, Direction and Mode fields
E/C Bit 1	**E/C** — Encryption Mode	0 = ECB (Electronic Code Book) 1 = CBC (Cipher Block Chaining)
E/D Bit 0	**E/D** — Encrypt/Decrypt	0 =encrypt data from Red memory to Black memory 1 = decrypt data from Black memory to Red memory

63.3.5.5 Status

The status register contains information reflecting the internal operation of the Secure RAM. When the Secure RAM is performing a long operation, such as encrypting data or clearing memory, and the memory is not accessible to the host processor, the status register will reflect the reason memory is not accessible. The status register will always be readable, so that the internal state may be polled at any time. The status register will not be reset by the Clear Memory command.

The ClrComp, EncrComp, AccessAllowed and EncrFail bits are set when the corresponding function (seen in the ClrMem, Encr, BlockAccess, ClrFail and InternalError bits) completes. The InterruptStatus bit will be set when any of these functions completes. These interrupt status bits will clear when the ClrInt bit in the Interrupt control register is set. They will also clear when some internal function starts (as seen in the Busy bit).

Note: There may be a one or two clock cycle synchronization period in the setting and clearing of the busy and interrupt status bits.

STATUS Status Register **Addr $2E8E_0010**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
				ENCR FAIL	ACCESS ALLOWED	ENCR COMP	CIR COMP	INT STATUS	SECRET KEY	INTERN ERROR	BAD KEY	CIR FAIL	BLOCK ACCESS	ENCR	CIR MEM	BUSY
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-9. Status Description

Name	Description	Settings
ENCR FAIL Bit 12	**Encr Fail** — Interrupt caused by encryption processing failed (start + length too large).	0 = Interrupt is not a result of encrypt failed 1 = Interrupt is a result of encrypt failed
ACCESS ALLOWED Bit 11	**Access Allowed** — Interrupt caused by block_access being deasserted	0 = Interrupt is not a result of access becoming allowed 1= Interrupt is a result of access becoming allowed
ENCR COMP Bit 10	**Encr Comp** — Interrupt caused by encryption processing completed	0 = Interrupt is not a result of encryption processing completing 1 = Interrupt is a result of encryption processing completing
CIR COMP Bit 9	**Clr Comp** — Interrupt caused by zeroize complete	0 = Interrupt is not a result of zeroizing memory completing 1= Interrupt is a result of zeroizing memory completing

Table 63-9. Status Description (Continued)

Name	Description	Settings
INT STATUS Bit 8	**Interrupt Status** — Indicates that there is a currently pending interrupt. This bit is reflected in the interrupt output signal, but while the output signal may be masked, the bit in the status register will never be masked.	0 = No interrupt pending 1 = Interrupt pending
SECRET KEY Bit 7	**Secret Key** — using the secret key, as specified by the Security Monitor. If a bad key is detected, this bit will show the default key being used.	0 = Default key will be used during encryption processing 1 = Secret key will be used during encryption processing
INTERN ERROR Bit 6	**Internal Error**— same as the external internal_error signal, and access will be denied until reset.	0 = No internal faults have been detected 1 = An internal fault has been detected
BAD KEY Bit 5	**Bad Key** — the secret key is not a valid key. The secret key will not be used for any encryption or decryption operations. (Busy is not set because of Bad Key).	0 = The secret key is a valid key 1 = The secret key is not valid, either due to not being programmed or perhaps a programming error
CIR FAIL Bit 4	**Clr Fail** — clearing memory failed, and access will be denied until reset.	0 = Built-in memory initialization hardware is working 1 = Built-in memory initialization hardware failed
BLOCK ACCESS Bit 3	**Block Access** — the Security Monitor is specifying that memory access should be denied.	0 = The Security Monitor is allowing access to internal memory 1 = The Security Monitor is denying access to internal memory
ENCR Bit 2	**Encr** — currently performing encryption processing.	0 = No encryption processing is being performed 1 = Encryption processing is currently being performed
CIR MEM Bit 1	**Clr Mem** — memory is currently being cleared.	0 = Memory is not currently being initialized 1 = Memory is currently being initialized
BUSY Bit 0	**Busy** — access to memory is denied. This is the logical OR of bits 1, 2, 3, 4, and 6.	0 = Access to memory is allowed 1 = Access to memory is denied (attempted access will result in a bus error)

63.3.5.6 Error Status

The error register latches information about the cause of an aborted bus transaction. If the TEA signal is asserted by the secure memory, then the reason for the assertion will be registered. The error register can be cleared, by writing to it.

The first 8 bits of the Error Status register are copied from the Status register at the time of the aborted access. The remaining bits give the reason that the access was aborted. There may be more than one error bit set (eg., for a User mode, byte access to an unaligned illegal address, all four error bits would be set).

ERROR Error Register **Addr $2E8E_0014**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
					UNALIGN	BYTE	ILLEGAL ADDRESS	USER	DEFAULT KEY	INTERN ERROR	BAD KEY	CIR FAIL	BLOCK ACCESS	ENCR	CLR MEM	BUSY
TYPE	r	r	r	r	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-10. ERROR Description

Name	Description	Settings
UNALIGN Bit 11	**Unalign** — Access to an unaligned address. Neither ARM nor MCore will generate a misaligned word access. If a word access to an odd address occurs, then the lower two bits are to be ignored, (and treated as 00). This Unalign bit will always read as 0 for ARM and MCore platforms, and is here for future platforms that may generate misaligned accesses.	0 = Legal word aligned access 1 = Illegal non-word aligned address
BYTE Bit 10	**Byte** — Byte or half-word access	0 = Legal word access 1 = Illegal byte or half-word access
ILLEGAL ADDRESS Bit 9	**Illegal Address** — Access attempted to an illegal or reserved address	0 = Legal register or memory access 1 = Illegal or reserved memory access
USER Bit 8	**User** — User mode access	0 = Supervisor mode access 1 = Illegal User mode access
DEFAULT KEY Bit 7	**Secret Key** — using the secret key, as specified by the Security Monitor. If a bad key is detected, this bit will specify the default key.	0 = Default key will be used during encryption processing 1 = Secret key will be used during encryption processing

Table 63-10. ERROR Description

Name	Description	Settings
INTERN ERROR Bit 6	**Internal Error** — same as the external internal_error signal, and access will be denied until reset.	0 = No internal faults have been detected 1 = An internal fault has been detected
BAD KEY Bit 5	**Bad Key** — the secret key is not a valid key.	0 = The secret key is a valid key 1 = The secret key is not valid, either due to not being programmed or perhaps a programming error
CLR FAIL Bit 4	**Clr Fail** — clearing memory failed, and access will be denied until reset.	0 = Built-in memory initialization hardware is working 1 = Built-in memory initialization hardware failed
BLOCK ACCESS Bit 3	**Block Access** — the Security Monitor is specifying that memory access should be disabled.	0 = The Security Monitor is allowing access to internal memory 1 = The Security Monitor is denying access to internal memory
ENCR Bit 2	**Encr** — currently performing encryption processing.	0 = No encryption processing is being performed 1 = Encryption processing is currently being performed
CLR MEM Bit 1	**Clr Mem** — memory is currently being cleared.	0 = Memory is not currently being initialized 1 = Memory is currently being initialized
BUSY Bit 0	**Busy** — access to memory is denied. This is the logical OR of bits 1, 2, 3, 4, and 6.	0 = Access to memory is allowed 1 = Access to memory is denied (attempted access will result in a bus error)

63.3.5.7 Interrupt Control

The Interrupt Control register contains three bits. When set, interrupts are disabled. This is cleared on reset, and thus when the memory is first zeroized, an interrupt will be generated. If this first interrupt needs to be masked, it must be done at the interrupt controller.

This register is writable even when access to other registers and memory is blocked. This is to allow clearing memory, even if encryption processing is currently occurring.

The interrupt mask bit is not cleared when the CLR_MEM bit is set, nor when the Zeroize input is asserted. All other registers, except Status, are reset. The Status register will continuously reflect the current state of the Secure RAM.

INT_MASK Interrupt Control Register **Addr $2E8E_0018**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
														CLR MEM	CLR INT	IMASK
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	slfclr	slfclr	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-11. INT_MASK Description

Name	Description	Settings
CLR MEM Bit 2	**CLR MEM** - Clear Memory - a self-resetting bit	0 = no operation 1 = Re-initialize both Red and Black memory, and reset Key Encryption Module and memory controller.
CLR INT Bit 1	**CLR INT** - Clear Interrupt - a self-resetting bit	0 = no operation 1 = Clear the interrupt bit
IMASK Bit 0	**IMASK** -Interrupt Mask	0 = Interrupt is enabled 1 = Interrupt is disabled

63.3.5.8 Configuration

The Configuration register contains static information which can be read at reset to allow software to identify the actual Secure Memory configuration. This will include, (at a minimum), the number of bytes in a data block. 3DES uses 8 byte data blocks. This register will also include the number of blocks in the red memory and black memory. Other defined configuration information is TBD.

CONFIGURATION Configuration Register **Addr $2E8E_001C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	VERSION[4:0]					BLACK SIZE[9:0]										RED SIZE[9]
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	RED SIZE[8:0]									BLOCK SIZE[6:0]						
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-12. Configuration Description

Name	Description	Settings
VERSION Bits 31-27	**Version** - 5 bit field returning the version of the Secure Memory. The initial version will return 1.	Always read as 00001 = 1
BLACK SIZE Bits 26-17	**Black Size** - 10 bit field returning the number of blocks in Black memory The current implementation will return 128 (1K bytes).	Always read as 00_1000_0000 = 128
RED SIZE Bits 16-7	**Red Size** - 10 bit field returning the number of blocks in Red memory The current implementation will return 128 (1K bytes).	Always read as 00_1000_0000 = 128
BLOCK SIZE Bits 6-0	**Block Size** - 7 bit field returning the number of bytes in a block. 3DES will return the value 8 (64 bit block) Other values may be returned by future versions of the Secure Memory.	Always read as 000_1000 = 8

63.3.5.9 Initialization Vector

Init_vector_0 and Init_vector_1 are the left most and right most halves of the 64 bit initialization vector for 3DES in CBC (Cipher Block Chaining) mode. When using ECB (Electronic Code Book) mode, the initialization vector is ignored. When data is encrypted (or decrypted) in CBC mode, the initialization vector will be updated after each block is processed.

If a large block of data is to be encrypted using CBC mode, then the block must be broken up into 1k byte chunks. Each chunk will be encrypted with the initialization vector being updated as the processing continues. If this must be stopped (due to a task switch, or other reason), the initialization vector should be saved and then restored when the processing continues.

INIT_VECTOR_0 Initialization Vector Register **Addr $2E8E_0020**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	Init_Vector[63:48]															
TYPE	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Init_Vector[47:32]															
TYPE	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INIT_VECTOR_1 Initialization Vector Register **Addr $2E8E_1024**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	Init_Vector[31:16]															
TYPE	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Init_Vector[15:0]															
TYPE	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-13. INIT_VECTOR Description

Name	Description	Settings
Init_Vector Bits 31-0	**Init_Vector** - The initialization vector is a 64 bit value which is used to initialize 3DES when using CBC mode.	

63.3.5.10 RED_MEM

The Red memory contains the decrypted sensitive information. It contains 1K bytes (256 words, or 128 8-byte blocks) of word addressable memory.

63.3.5.11 BLACK_MEM

The Black memory contains the encrypted data. It contains 1K bytes (256 words, or 128 8-byte blocks) of word addressable memory.

63.3.5.12 Initialization Vector

Init_vector_0 and Init_vector_1 are the leftmost and rightmost halves of the 64 bit initialization vector for 3DES in CBC (Cipher Block Chaining) mode. When using ECB (Electronic Code Book) mode, the initialization vector is ignored. When data is encrypted (or decrypted) in CBC mode, the initialization vector will be updated after each block is processed.

If a large block of data is to be encrypted using CBC mode, then the block must be broken up into 1k byte chunks. Each chunk will be encrypted with the initialization vector being updated as the processing continues. If this must be stopped (due to a task switch, or other reason), the initialization vector should be saved and then restored when the processing continues.

63.3.6 Functional Details

63.3.6.1 DMA Controller

There will be a DMA controller which controls the flow of data through the encryption module and between the red and black memories. It uses the red start address, the black start address and the length registers. When a command is written to the Control register, the DMA controller starts the encryption process.

The DMA controller will perform two reads from memory, and present the encryption module with a 64-bit block of data. It will then assert the start signal, and wait for the done signal from the encryption module. When the done signal is asserted, the DMA controller will write the data to the proper location in memory. The length field will then be decremented. When the length field reaches zero, the DMA controller will assert IRQ.

It is considered an error if either the red_start + length > red_size or the black_start + length > black_size. These conditions are checked when the Start Encryption bit is set in the control register. If either of these conditions is detected, the encryption will not occur, the interrupt will be generated immediately, and the Encr_Fail bit of the status register will be set.

If a host access to the memory occurs during the encryption process, TEA will be asserted. The host should read the Status register to verify that the memory is accessible before reading or writing data.

63.3.6.2 Zeroization

When the Security Monitor sends a Zeroize signal to the Secure RAM, it will cause the contents of memory to be cleared. The zeroize_done signal will go low, indicating that the memory has started to zeroize. The zeroize counter will start at memory location 0, and write 0 to the first memory address. The zeroize counter will increment, and the address value will be written to the next address. This will continue until all memory, both red and black, has been initialized; with the last black memory location getting the value 511.

At the end of the memory clear operation, the zeroize counter will run through all of the addresses once again. This time, the memory will be read, and the value will be checked to see that it really did get initialized. If it did not, then the zeroize_failed signal will be asserted. If all of the memory was checked and cleared properly, then the zeroize_done signal will go high, indicating proper completion.

The zeroize signal will also be used to reset the encryption module and the DMA controller. While the zeroize_complete signal is low, any attempted access by the host will result in a transfer error acknowledge.

When the Zeroize signal is asserted, or when the CLR MEM bit in the Interrupt Control register is set, not only will the red and black memory be initialized, but most of the registers will also be reset. The Red_start, Black_start, Length, Control, Error Status, Init_vector_0 and Init_vector_1 registers will be cleared. The encryption engine will also be reset.

During reset, the memory will be zeroized before access to it is permitted. When the zeroize function is completed, an interrupt will be generated. During this zeroize function, no access to the secure memory or the registers will be allowed. Since interrupts are enabled by default, an interrupt will occur after this first zeroize completes. If this needs to be masked, it must be masked at the interrupt controller.

63.3.6.3 Blocking Access to Memory

Access to the Red and Black memories and to the control registers will occur when the Busy bit in the status register is set. When access is blocked, any bus access to these memory or register addresses will result in a bus error. No data can be read or written. The only registers which can be read during block access are: Status, Error Status, Interrupt Control and Configuration. Only the Interrupt Control register can be written. There are five conditions which cause access to memory to be denied.

1. When memory is currently being initialized.
2. During encryption processing.
3. When the Security Monitor is specifying that memory access should be denied.
4. If memory initialization fails.
5. As a result of an internal hardware error.

If the SCM_ENABLE input is not asserted, the SCC module will not respond to bus accesses at all. It will specifically not respond with a bus error. The bus cycle will terminate when the system watch-dog timer (not the Security Monitor's) times out.

63.3.6.4 Laser ID

The Laser ID module will have as its input the 168-bit secret key. The physical structures for the secret key will reside elsewhere. It will contain a key mux, to select between the secret key and the default key and test logic to determine the validity of the secret key.

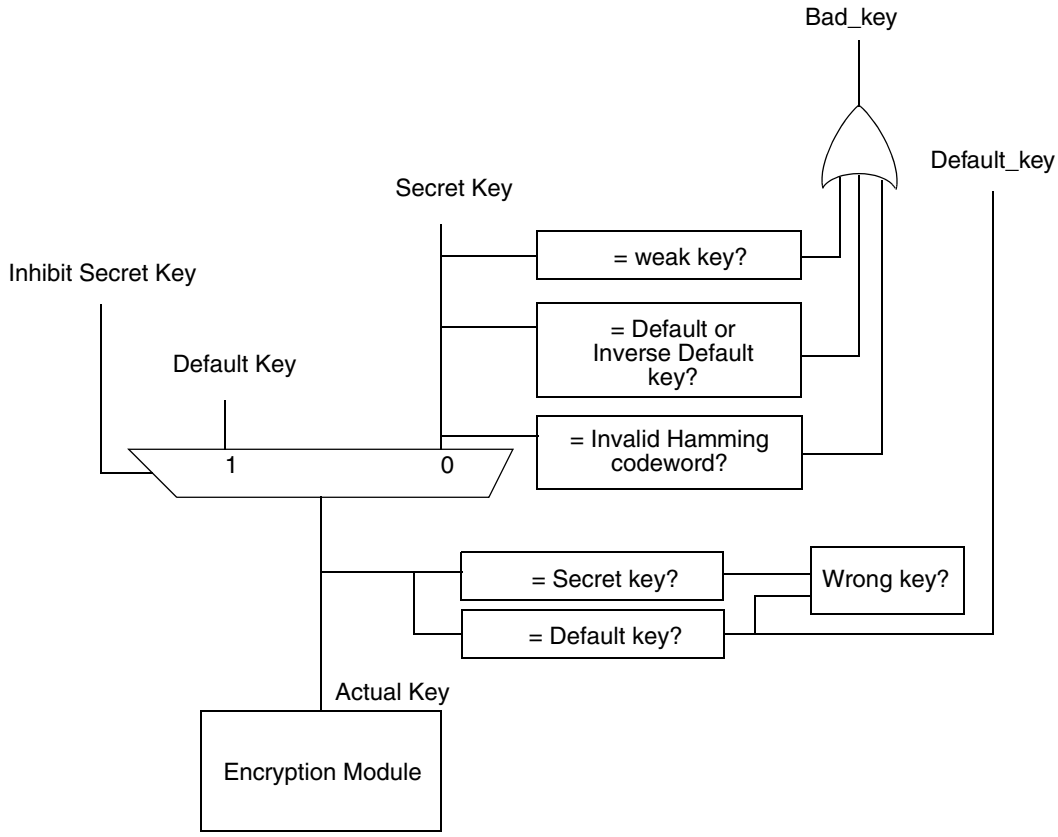


Figure 63-4. Laser ID Block Diagram

The 168-bit secret key is actually three 56-bit keys. The bits of the secret key are numbered from from left to right, 167 to 0. The first key is composed of bits 167 to 112; the second, bits 111 to 56; and the third, bits 55 to 0. Each of these three keys correspond to a 64-bit DES key with the parity bits removed, and processed through the key permutation.

A 64-bit DES key is numbered from left to right, 1 to 64. Bit 57 of the first 64-bit DES key is bit 167 of the secret key, and bit 49 is bit 166 of the secrect key. Bit 12 of the third 64-bit DES key is bit 1 of the secret key, and bit 4 is bit 0 of the secret key.

The secret key will be checked, to assure that the laser ID fuses were actually programmed the with the intended value. The key will be compared to the DES weak keys (there are four DES weak keys), the default key and the inverse of the default key. The secret key must never contain a weak key or be the inverse of the default key. The secret key will also include nine code bits, which forms a Hamming code capable of detecting all single, double and triple bit errors in programming the laser fuses, and the secret key will be checked to see that it forms a valid codeword. If the secret key is found to contain a weak key, default key, inverse default key, or be an invalid codeword, then this would indicate that the secret key was either not programmed or was misprogrammed, and the secret key should not be used. The actual key used by the encryption module is also compared to the default key and secret key. If it is found to be the wrong key, (not the key indicated by inhibit_secret_key), this indicates a hardware failure, and the Secure RAM module should not be used.

The following is a list of the patterns which are specifically checked as representing bad keys. (x means don't care). No keys that match any of these patterns should be programmed into the Laser ID fuses.

- 00000000000000_xxxxxxxxxxxxxx_xxxxxxxxxxxxxx
- xxxxxxxxxxxxxx_00000000000000_xxxxxxxxxxxxxx
- xxxxxxxxxxxxxx_xxxxxxxxxxxxxx_00000000000000
- 11111111111111_xxxxxxxxxxxxxx_xxxxxxxxxxxxxx
- xxxxxxxxxxxxxx_11111111111111_xxxxxxxxxxxxxx
- xxxxxxxxxxxxxx_xxxxxxxxxxxxxx_11111111111111
- 0000000FFFFFFF_xxxxxxxxxxxxxx_xxxxxxxxxxxxxx
- xxxxxxxxxxxxxx_0000000FFFFFFF_xxxxxxxxxxxxxx
- xxxxxxxxxxxxxx_xxxxxxxxxxxxxx_0000000FFFFFFF
- FFFFFFF0000000_xxxxxxxxxxxxxx_xxxxxxxxxxxxxx
- xxxxxxxxxxxxxx_FFFFFFF0000000_xxxxxxxxxxxxxx
- xxxxxxxxxxxxxx_xxxxxxxxxxxxxx_FFFFFFF0000000
- 55555555555555_AAAAAAAAAAAAAA_33333333333333
- AAAAAAAAAAAAAA_55555555555555_CCCCCCCCCCCCCC

NOTE:

If the secret key is the same as the default key, this will appear as a hardware failure in the checking logic, and the Secure RAM will not be usable. This should never happen in practice.

The Inhibit Secret Key signal will be asserted under the following conditions:

- Inhibit_key signal from Security Monitor is asserted
- The Secure RAM state machine is in its Reset state
- The Secure RAM state machine is in its Fail state

63.3.6.5 Error Code for the Laser ID

The Hamming code used for the 168-bit Laser ID uses nine code bits. It will detect all 1, 2 and 3 bit errors in the codeword. It will also detect most (though not all) multiple bit errors. Therefore, the secret key will actually contain 159 random bits.

Numbering the bits in the Laser ID from 0 to 167, the code bits will be bits 0, 1, 2, 4, 8, 16, 32, 64 and 128. All of the remaining bits will be data bits. Each of these code bits is the modulo 2 sum (i.e., XOR or parity) of a subset of the bits in in the entire word. Note that key1 will contain one code bit (bit 128), key2 will contain one code bit (bit 64), and key 3 will contain 7 code bits (0, 1, 2, 4, 8, 16, 32).

To determine which bits are used to form each code bit, first look at the binary representation of the bit position of each code bit (ignoring bit 0 for the time being).

Table 63-14. Code Bit Position

Code Bit	Binary Represientation
0	00000000
1	00000001
2	00000010
4	00000100
8	00001000
16	00010000
32	00100000
64	01000000
128	10000000

For code bit number 1, there is a single ‘1’ bit in its binary representation. This code bit will be the modulo 2 sum (XOR) of all bit positions which also have a ‘1’ in this same point in their binary representation (i.e., all odd bit positions). Alternatively, if we bitwise AND code bit 1’s bit number with a data bit’s bit number, and the result is not zero, then that bit gets XOR’ed into code bit 1. The same is true for the other code bits, (excepting code bit 0). For example, code bit 2 will be the XOR of bits 3, 6, 7, 10, 11, 14, 15, ..., 166, 167, while code bit 128 will be the XOR of all bits from 129 through 167 inclusive. After all of the other code bits have been calculated, code bit 0 is simply the XOR of all of the other bits (including the code bits).

Another way of looking at this is to ask which code bits does a data bit affect. If we look at the binary representation of the bit position of a data bit, the 1’s represent the code bits that this data bit affects. For example, data bit 99 (01100011), will be XOR’ed into code bits 1, 2, 32 and 64.

63.3.6.6 Generating the Code Bits

To generate a code word to be programmed into the laser fuses for the secret key, it is simplest to start with an array of 168 random bits, (the array numbered 0 through 167). The following psuedo-code will then replace the nine code bits with their proper values. The starting random bits at those locations are discarded.

Example 63-1.

```
// Generate the Hamming code bits for the 168 bits
// stored in Number. The values at the locations of the code bits
// are ignored and will be overwritten with the generated values.
Generate_code_bits(bool Number[168])
{
      int i, j;

      // Calculate each code bit in turn
      for(i = 1; i <= 128; i = (i << 1)) {
            // Examine each data bit
            // Only bits greater than i need to be checked as no
            // bit less than i will ever be XORed into i
            // J starts at i so that Number[i] is initialized to 0
            for(j = i; j <= 167; j = j + 1) {
                        if ( (i & j) != 0) {
                              Number[i] = Number[i] ^ Number[j];
                        }
            }
      }
      // Calculate the overall parity
      // J starts at 0 so that Number[0] is initialized to 0
      // Number[0] contains the even parity of all of the bits
      for(j = 0; j <= 167; j = j + 1) {
            Number[0] = Number[0] ^ Number[j];
      }
```

63.3.6.7 Checking the Code Bits

The logic within the Secure RAM will use an equivalent of the following psuedo-code to check that the 168-bit secret key value is a legal codeword, or if any of the bits were not programmed properly.

Example 63-2.

```
// Check the Hamming code bits for the 168 bits stored in Number.
// If an error is detected, return BAD.
// If no errors are detected, return GOOD.
bool Check_code_bits(bool Number[168])
{
      int i, j, n;
      bool Check[9];

      // Calculate each code bit in turn
      // i steps through Number
      // n steps through Check; n = log(i)
      for(i = 1, n=0; i <= 128; i = (i << 1), n = n + 1) {
            // Examine each data bit
            // Only bits greater than i need to be checked as no
            // bit less than i will ever be XORed into Check[n]
            Check[n]= 0;
            for(j = i; j <= 167; j = j + 1) {
                        if ( (i & j) != 0) {
                             Check[n] = Check[n] ^ Number[j];
                        }
            }
      }
      // Calculate the overall parity
      // Check[8] is the even parity of all of the bits
      for(j = 0; j <= 167; j = j + 1) {
            Check[8] = Check[8] ^ Number[j];
      }
      // Any non-zero Check bit means an error has been detected
      for(n = 0; n <=8; n = n + 1) {
            if(Check[n] != 0) {
                        return BAD;
            }
      }
      return GOOD;
}
```

63.4 Security Monitor

The Security Monitor is one of the two blocks in the Security Controller (SCC) module. The other is the Secure RAM. The Security Monitor is an element used to provide some security assurance for the platform. Specifically, it determines when and how Secure RAM resources will be used in the system as well as providing mechanisms for verifying software algorithm integrity. This block ensures that the system is running in such a manner as to provide protection for the sensitive data that is resident in the system. This is a scaled back version of the Monitor used in Motorola's High Grade Security Architecture products. As the need for assurance increases in the future, this monitor's functionality can be replaced with a monitor that checks for additional failure mechanisms.

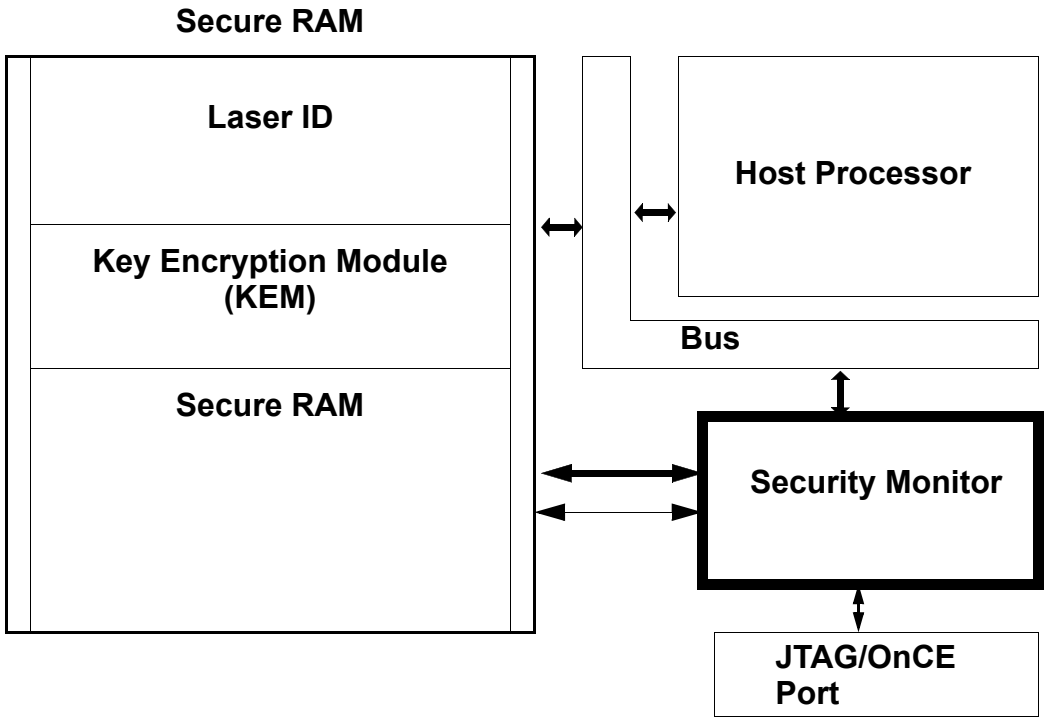


Figure 63-5. High Level Block Diagram of Security Monitor with Host Processor

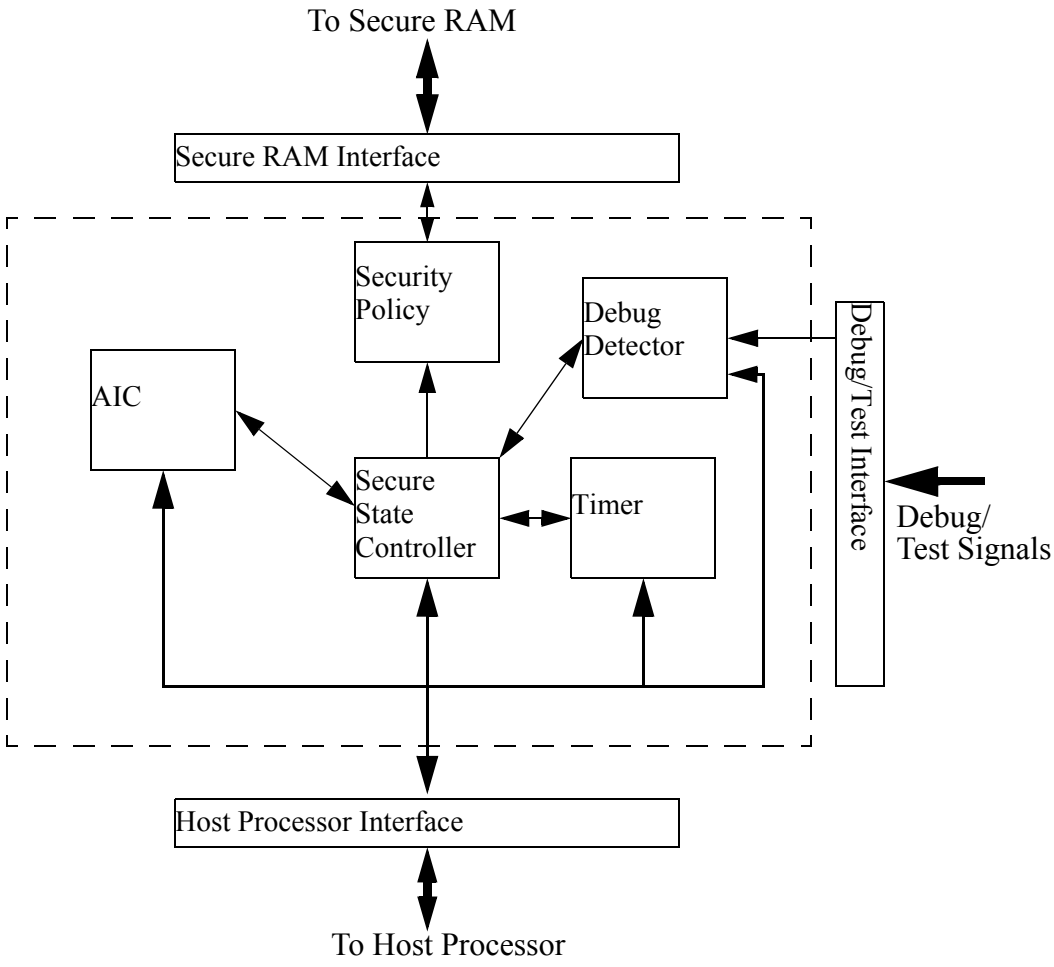


Figure 63-6. Security Monitor Block Diagram

63.4.1 Monitor Components

The Security Monitor consists of five main sub-blocks: The Secure State Controller, the Security Policy, the Algorithm Integrity Checker (AIC), the Timer, and the Debug Detector. See Figure 63-6. When any of these monitor blocks encounter an error condition, the monitor will take the appropriate action, based on the Security Policy, to limit access to the Secure RAM and alert the host processor of the error. Figure shows a more detailed diagram of the Security Monitor.

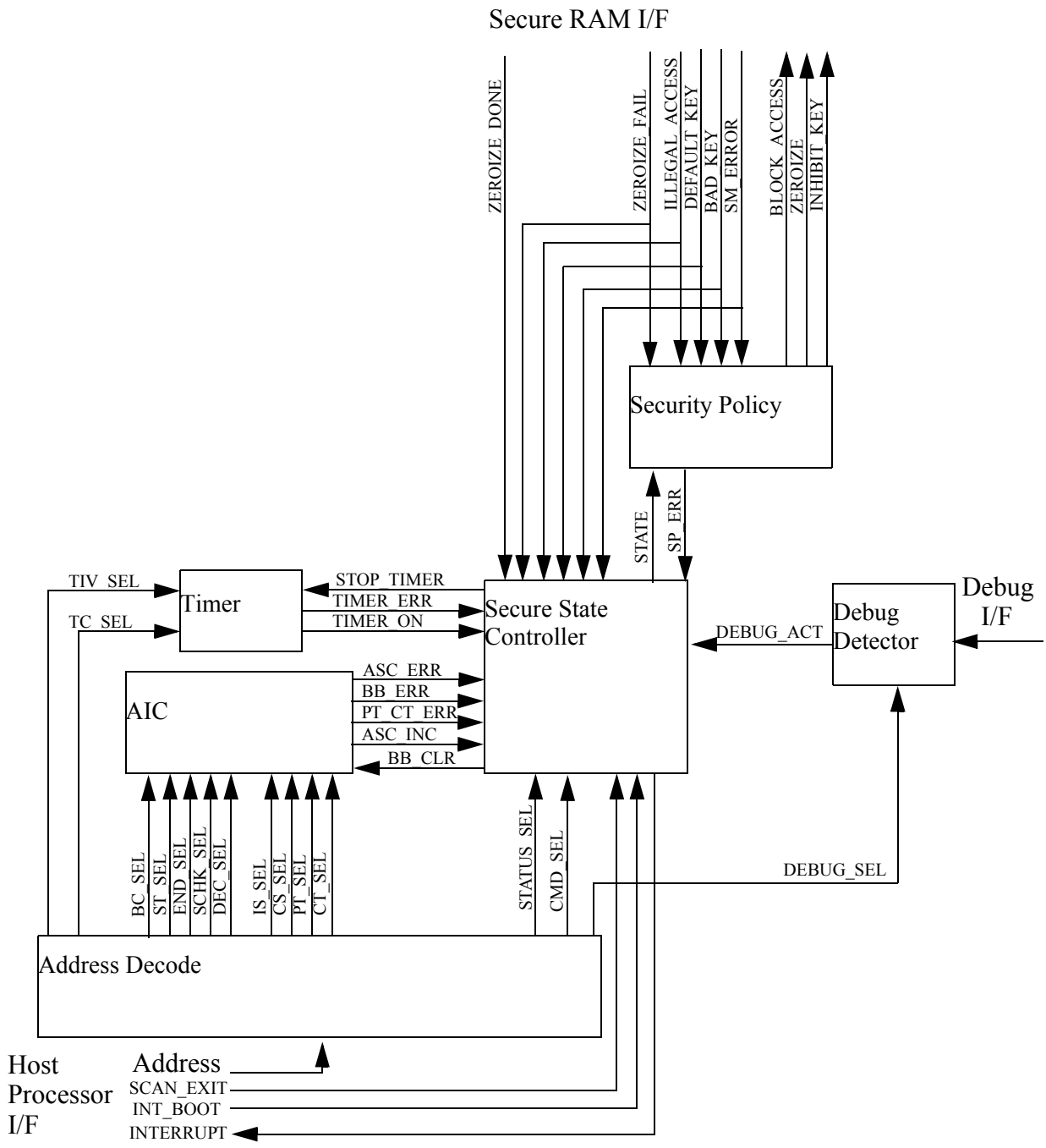


Figure 63-7. Security Monitor Detailed Diagram

63.4.1.1 Secure State Controller

The Secure State Controller is a state machine that controls the security modes of the chip. The Secure State Controller starts from a system reset. It must meet predefined operating conditions and successfully pass a series of tests to ensure that the host processor and all participating hardware are in a secure condition. If a Secure Initial State has not been achieved, the Monitor will transition to a Non-Secure or

Failure state. The timer in the Security Monitor will be used to ensure that the transition to the Secure state occurs within an acceptable amount of time, otherwise the monitor will transition to the Non-Secure state. If a Secure State has been achieved and an error is detected, it will transition to a Failure state.

Figure shows the block diagram for the Secure State Controller. Inputs to the state machine are the Command Register (Command), a signal indicating that the chip has booted from internal ROM (INT_BOOT), signals indicating whether the Secure RAM zeroize was successful (ZEROIZE_DONE and ZEROIZE_FAIL), a signal indicating that a debug or test mode is active (DEBUG_ACT), a signal indicating that the Algorithm Sequence Checker (ASC) has completed (ASC_INC), timer signals indicating whether the timer is running (TIMER_ON), and error signals from the Security Policy, Algorithm Sequence Checker, Bit Bank, Plaintext/Ciphertext Comparator, and Timer blocks (SP_ERR, ASC_ERR, BB_ERR, PT_CT_ERR, TIMER_ERR). The State Machine outputs its state (STATE) to the Security Policy block. The Status register is used to relay operational status to the host processor. The Command Register is used to by the host processor to give commands to the Monitor. The Command Register provides the means to enable the interrupt, clear the interrupt, clear the bit bank, or generate a Software Alarm.

The Interrupt Controller determines if an interrupt condition occurs and if the external Interrupt signal should be asserted. An interrupt condition is either a Software Alarm, a Security Policy error, an ASC error, a Bit Bank Error, a Plaintext/Ciphertext Comparator Error, a Timer Error, or Debug signal activity. In Secure State, the external Interrupt signal is asserted whenever an interrupt condition occurs. In the other states, the external Interrupt signal will be asserted only when the Interrupt Enable bit is set in the command register and an interrupt condition occurs.

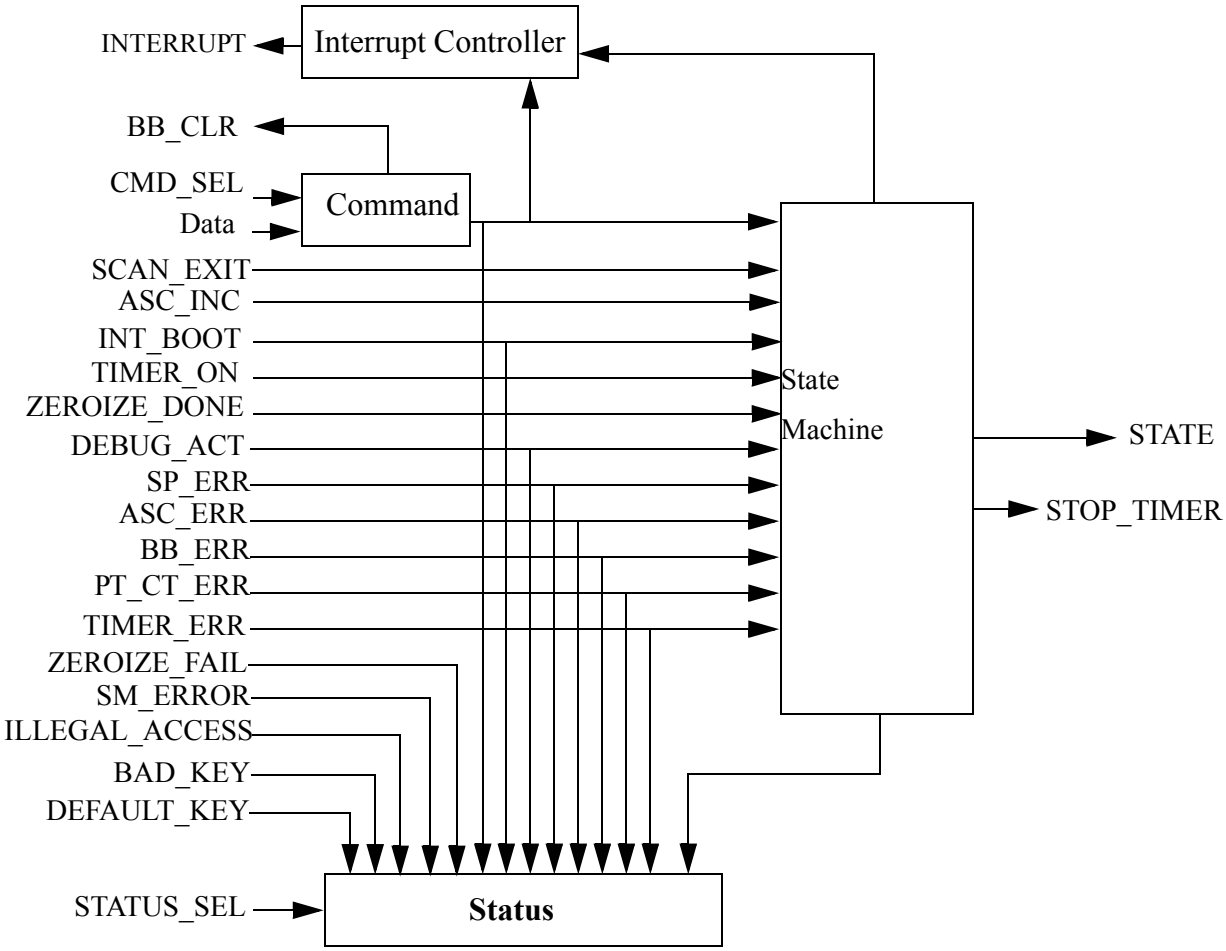


Figure 63-8. Secure State Controller Block Diagram

Figure 63-9 is the state diagram for the Secure State Controller State Machine.

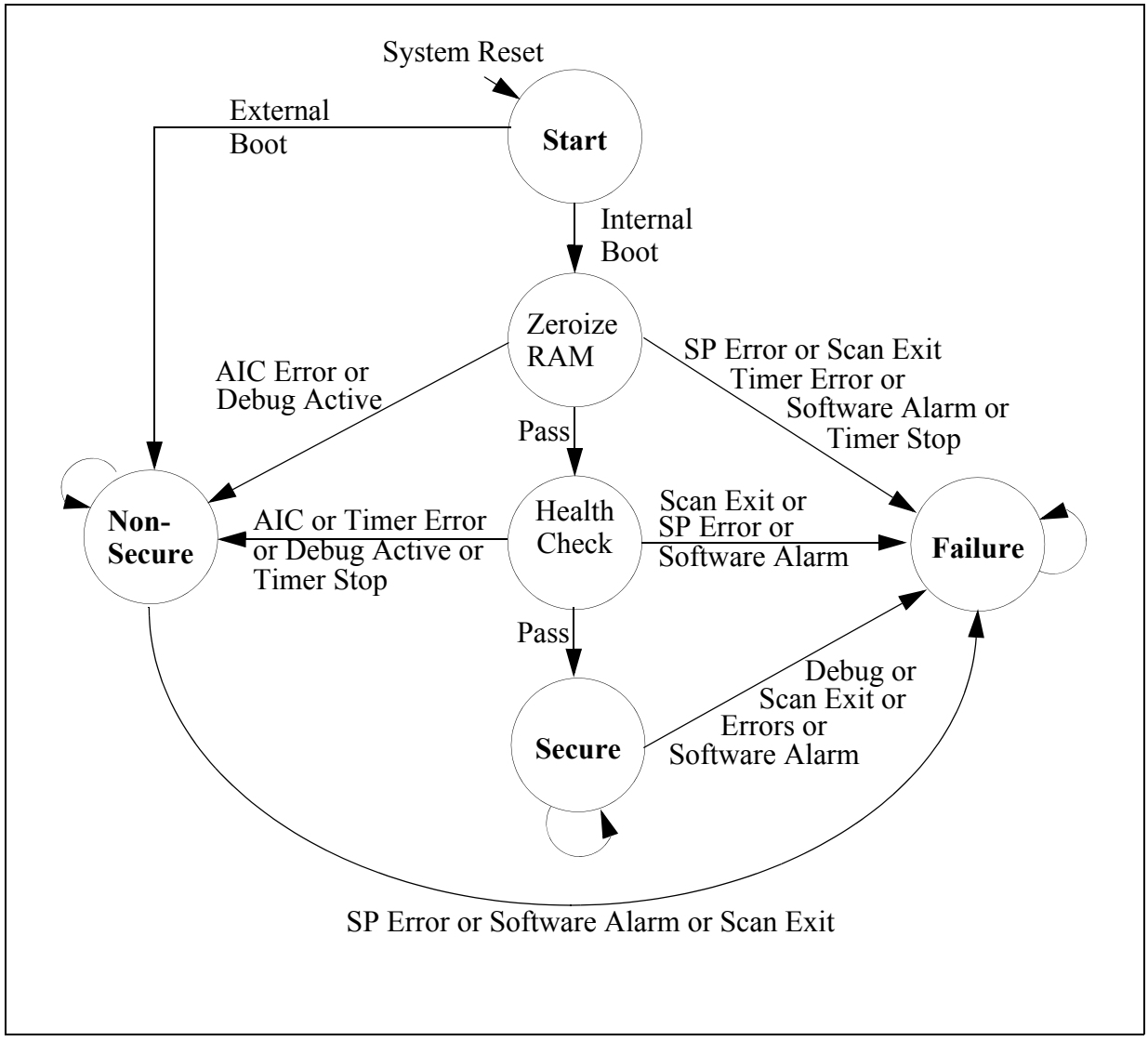


Figure 63-9. Secure State Controller State Diagram

Start State - The Secure State Controller state machine begins with the System reset. The controller will transition into one of two states: the Non-Secure State if the processor boots off chip, or into the Zeroize RAM State if the processor boots from internal ROM.

Zeroize RAM State - The Secure RAM shall perform a zeroization process (RAM set to a known value). If the Zeroization process fails the Monitor shall transition to the Failure State. If the Zeroization process succeeds the Monitor will transition to the Health Check State.

Health Check State - The processor will validate the contents of the external Flash, perform a RAM test, and perform any other tests that are necessary to reach a Secure state. The Algorithm Sequence Checker (ASC) is used when performing these tests to verify that all of the tests are completed, and that they are completed in the correct sequence. First, the SeqStart and SeqEnd registers must be written with the appropriate sequence start and end values. As each test is completed, the corresponding sequence check value must be written to the SeqChk register. See Section 63.4.1.3.1 for more information on use of the ASC. If the tests are successful and all of the correct sequence values are written to the ASC, the Monitor will transition to Secure and the timer will be stopped. If the tests are not successful and thus the correct sequence values are not written, the monitor will transition into the Non-Secure state.

Non-Secure - This state is entered when the processor fails to perform its start sequence correctly. The only way to transition out of the Non-Secure state is through a reset.

Secure State - This state is entered when the processor has booted from internal ROM and all of the processor and Monitor tests have completed successfully. In this state, if debug modes or internal failures are detected, the Secure RAM will zeroize and the Monitor will transition into the Failure State.

Failure State - The Failure State can only be reached by one of three sequences: either a Security Policy error occurs, scan_exit is asserted, or the Monitor has reached its Secure State and a debug mode is entered or an error is detected by the Monitor. The Monitor will remain in this state until the processor is reset. Only Status, Command, and Debug Detector Status (DDStatus) registers may be accessed while in Failure State.

63.4.1.2 Security Policy

The Security Policy block uses State information from the Secure State Controller (STATE) along with inputs from the Secure RAM (ILLEGAL_ACCESS, BAD_KEY, DEFAULT_KEY) to determine what access to the Secure RAM is allowed based on the policy table (see Figure). It also determines if a Security Policy error (SP_ERR) has occurred. Table shows when a Security Policy error is generated, based on the current state and the active error signal.

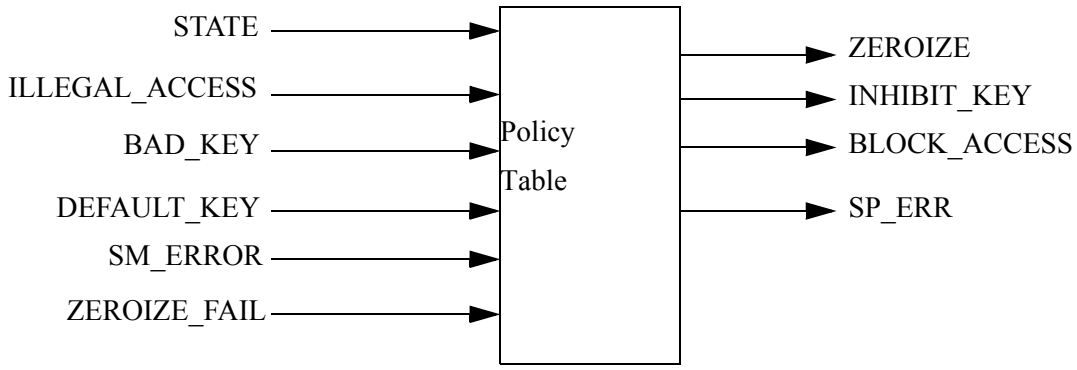


Figure 63-10. Security Policy Block Diagram

Table 63-15. SP_ERR Assertion

Signal	SP_ERR Action	
	Not Secure State	Secure State
ILLEGAL_ACCESS=1	No Action	No Action
BAD_KEY=1	SP_ERR Asserted	SP_ERR Asserted
DEFAULT_KEY=1	No Action	SP_ERR Asserted
DEFAULT_KEY=0	SP_ERR Asserted	No Action
SM_ERROR=1	SP_ERR Asserted	SP_ERR Asserted
ZEROIZE_FAIL=1	SP_ERR Asserted	SP_ERR Asserted

Table 63-16 shows the Security Policy implemented for the chip in the Policy Table. This block uses state information from the Secure State Controller to provide the access control signals for the Secure RAM.

Table 63-16. Security Policy

	Start / Zeroize RAM	Health Check	Non-Secure	Secure	Failure
Laser ID	No Access	No Access	No Access	Access Enabled	No Access
KEM	No Access	Access Enabled	Access Enabled	Access Enabled	No Access
RAM	No Access	Access Enabled	Access Enabled	Access Enabled	No Access
Monitor	Perform Secure Initial State tests	Perform Secure Initial State tests	Report Error Status	Monitor for Errors	Send Interrupt to host (if interrupts not masked) Zeroize other blocks

The Security Policy has five different levels of access control: Start/Zeroize RAM, Health Check, Non-Secure, Secure, and Failure.

The Power-Up level includes the Start, and Zeroize RAM states. During these states, no accesses will be allowed to the Secure RAM.

The Health Check level includes the Health Check state. During this state, the Secure RAM may be accessed, but the Default Key will be used to perform encryption/decryption.

The Non-Secure level includes the Non-Secure state. In this state, access is allowed to the Key Encryption Module (KEM) and the RAM, but the default key will be used in the KEM. Access will not be granted for the Laser ID.

The Secure level includes the Secure state. In this state, full access to the Secure RAM is allowed.

The Failure level includes the Failure state. In this state, no access is allowed to any of the Secure RAM and the memory and KEM are zeroized.

Table 63-17. Security Policy Outputs

	Start / Zeroize RAM	Health Check	Non-Secure	Secure	Failure
SMN_BLOCK_ACCESS	Asserted	Negated	Negated	Negated	Asserted
SMN_INHIBIT_KEY	Asserted	Asserted	Asserted	Negated	Asserted
SMN_ZEROIZE	Negated	Negated	Negated	Negated	Asserted

63.4.1.3 Algorithm Integrity Checker (AIC)

The Algorithm Integrity Checker (AIC), shown in Figure , is used in conjunction with software to provide assurance that critical software (such as an encryption algorithm) operates correctly. It is also an integral part of the power-up procedure since it must be used to achieve a secure state. It is broken down into three components: an Algorithm Sequence Checker (ASC), a Bit Bank (BB), and a Plaintext/Ciphertext Comparator (PT/CT). Some software will be able to take advantage of all of these blocks, but some of these blocks will not be appropriate for particular software and need not be used.

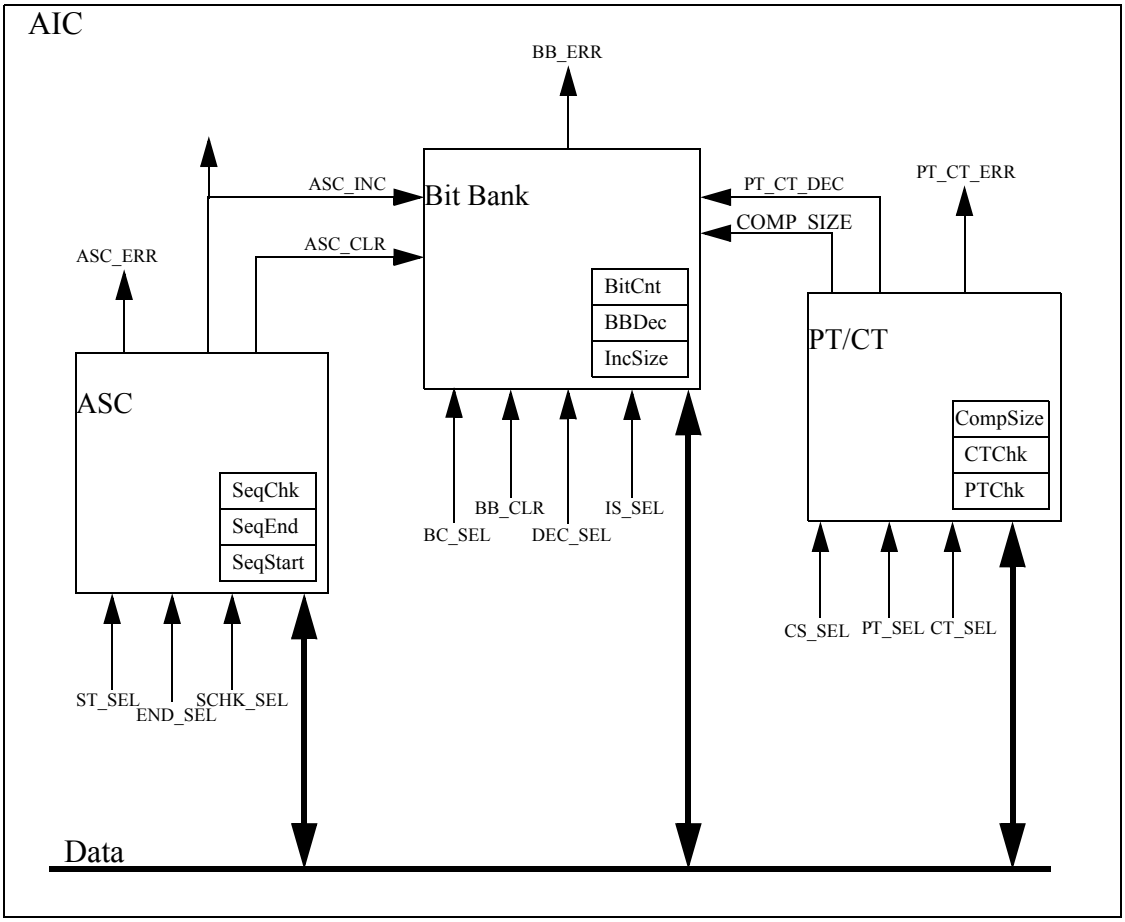


Figure 63-11. Algorithm Integrity Checker Block Diagram

To verify that specific critical software algorithm routines are operating in the proper sequence, each routine can communicate pre-defined steps to the Algorithm Integrity Checker (AIC). The Algorithm Sequence Checker (ASC) of the AIC will ensure that these steps occur in the right order and that the right number of steps are taken. The Bit Bank (BB) will keep a count of the number of encryption bits that have been generated and used, so the software does not send out more traffic bits than are produced. It may also be used as a real time process check. The Plaintext/Ciphertext Comparator provides a hardware mechanism to verify that the encryption appears valid by checking that the ciphertext is not the plaintext exclusive-ored with all 1's, all 0's, or alternating 1's and 0's. Depending on the individual algorithm or its implementation, one or more of these blocks within the AIC may not be used.

63.4.1.3.1 Algorithm Sequence Checker (ASC)

The ASC is responsible for testing that a given software routine follows a given set of steps. This is done by having the software check in with the hardware using predefined values that are hard coded into the critical software routine. The ASC will use a simple Linear Feedback Shift Register (LFSR) to calculate independently from the software the succeeding values coded into the software algorithm. If the software checks in with the hardware with the wrong value, the hardware concludes that the software is not following the proper sequence. There is no time requirement for writing successive values.

The predefined values are determined by the maximal length irreducible polynomial $F(x) = x^{16} + x^{15} + x^{13} + x^4 + 1$. A starting value is determined by the software and is used to initialize the LFSR implementing this polynomial. When the software reaches a certain point in its execution, it writes the next sequential value to the ASC. When this is written, the value is compared with the value calculated by the LFSR. Upon a successful compare, the LFSR cycles to the next value (according to F(x)). Sequential values are written until the software algorithm is finished. At this point, the LFSR should contain the same ending value that was predetermined by the software.

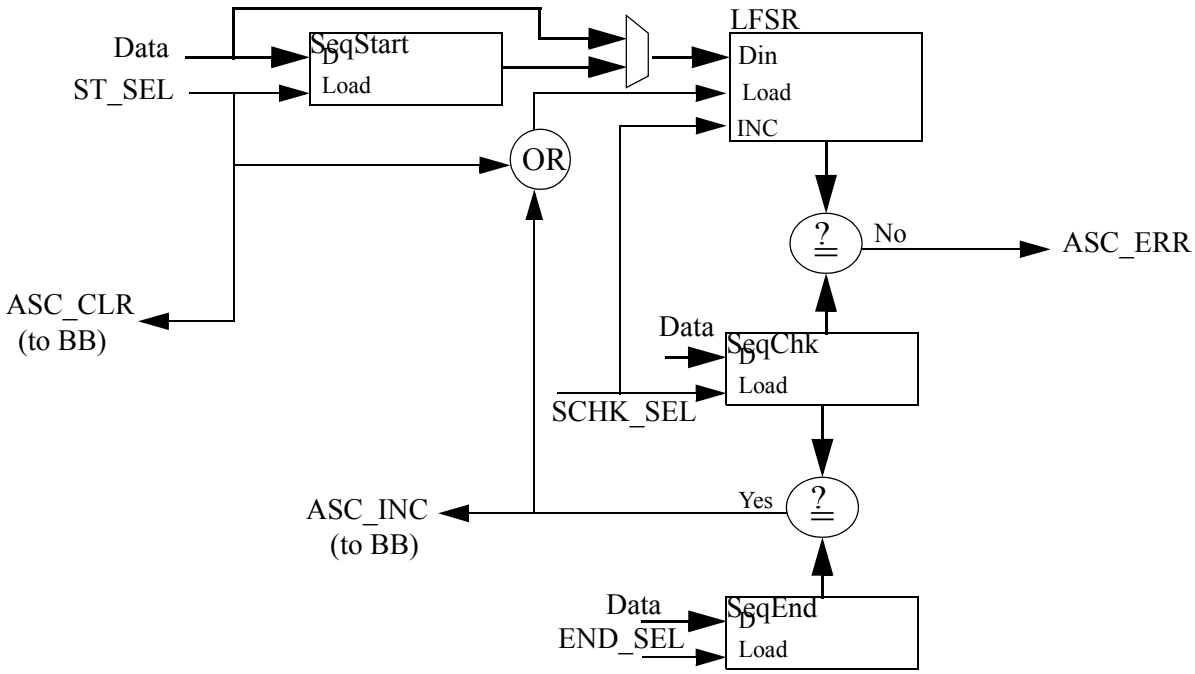


Figure 63-12. ASC Block Diagram

Figure shows the ASC block diagram. Before using the ASC, it must be initialized. To initialize the ASC, the Sequence Start (SeqStart) and Sequence End (SeqEnd) registers must be written. Writing the first 16-bit sequence value to the SeqStart register loads the LFSR with the beginning of the sequence and clears the Bit Bank (ASC_CLR signal). The LFSR will then be advanced to the first sequence step value. The final 16-bit sequence value must be written to the SeqEnd register. This value determines when the algorithm has finished.

As the software proceeds through its algorithm, it checks in with the hardware by writing the correct step value to the Sequence Check (SeqChk) register. Each time the software writes a step value, it is compared with the internally calculated step value that uniquely identifies that step. If the step value written by the software matches the expected step value, the software is seen as running in the correct sequence. The LFSR will then advance to the next step value. If any step values are skipped, inserted, or provided in the wrong order, the ASC will see a different step value written from that which was expected and will trigger an ASC error (ASC_ERR). In a proper sequence, when the software writes the correct value for the last step at the right time (step value written matches the number in End), the Bit Bank is incremented (ASC_INC signal) and the LFSR is reintialized with the Start value so that another identical sequence may be verified without manually reinitializing the ASC. The Start and End values do not need to be rewritten each time the ASC is used, but may be rewritten if a different routine is to be checked. There must be at least one clock period between the final write of a sequence and the first write of the next sequence so the LFSR can reinitialize (load the start value from the start register).

Table 63-18 shows a small sample of a valid sequence determined by F(x). Any value may be used as the start value written to the Start register except 0x0000. Values immediately following the start value would then be the data written successively into the SeqChk register. Additional step values can be calculated from F(x).

Table 63-18. Algorithm Sequence Checker LFSR Step Values

Index	Value	Index	Value	Index	Value	Index	Value
1	aaaa	11	972a	21	f0e5	31	34fc
2	5555	12	cb95	22	f872	32	1a7e
3	2aaa	13	e5ca	23	fc39	33	8d3f
4	9555	14	72e5	24	7e1c	34	c69f
5	caaa	15	3972	25	3f0e	35	e34f
6	e555	16	1cb9	26	9f87	36	f1a7
7	72aa	17	0e5c	27	4fc3	37	f8d3
8	b955	18	872e	28	a7e1	38	7c69
9	5caa	19	c397	29	d3f0	39	3e34
10	2e55	20	e1cb	30	69f8	40	1f1a

63.4.1.3.2 Bit Bank

The AIC incorporates a Bit Bank that works along with the Algorithm Sequence Checker. The Bit Bank counts transmit traffic bits and key bits generated by the software in an ASC sequence and checks to see that no more traffic bits are sent than were produced by the routines checked by the AIC.

Figure 63-13 shows the block diagram for the Bit Bank. The bit repository is the 11-bit Bit Count (BitCnt) register, allowing storage of 0-2047 bits. When a new start value is written to the ASC (ASC_CLR) or a Bit Bank Clear (BB_CLR) command is given, the BitCnt register will reset to 0. When an ASC sequence successfully finishes (ASC_INC), the BitCnt register will be incremented by the value in the Increment Size (IncSize) register. When Plaintext and Ciphertext are verified to be different in the Plaintext/Ciphertext Comparator (PT_CT_DEC), the BitCnt register will be decremented by the value in the CompSize register. The BitCnt register may also be decremented by the value of the Bit Bank Decrement (BBDec) register when written (DEC_SEL). An error occurs if the BitCnt is incremented past its maximum size (overflow error) or if it is decremented below zero (underflow error). There must be one clock period between the final sequence write and a BBDec write to give the Bit Bank time to increment from the completion of a successful sequence.

The Bit Bank may be disabled by writing 0 to the IncSize and CompSize registers (CompSize part of PT/CT described in 63.4.1.3.3).

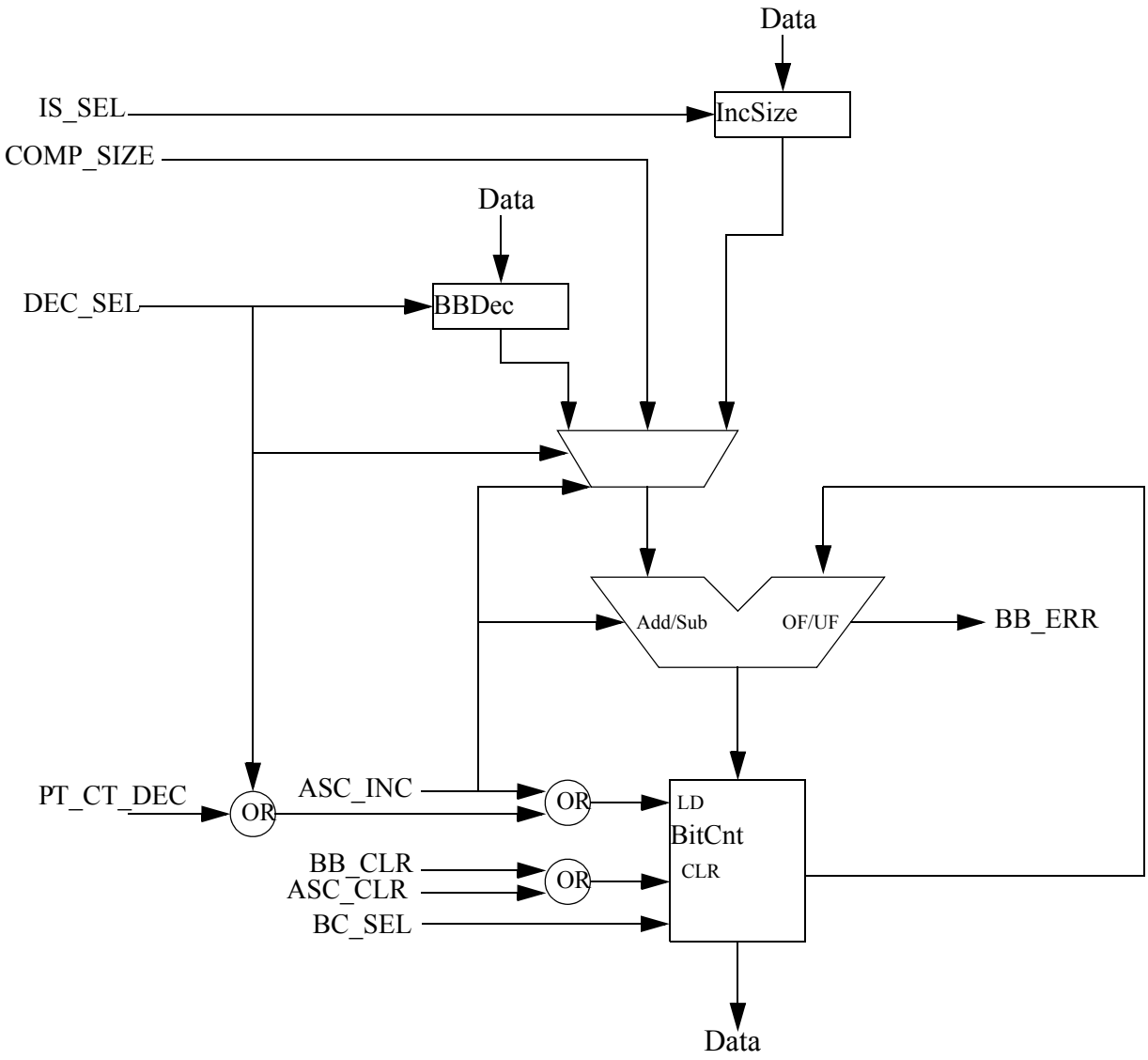


Figure 63-13. Bit Bank Block Diagram

63.4.1.3.3 Plaintext/Ciphertext Comparator

The Plaintext/Ciphertext Comparator (PT/CT) performs a comparison of the plaintext (PTChk register) and the ciphertext (CTChk register) to verify that the processor is sending out encrypted data.

Figure shows the block diagram for the Plaintext/Ciphertext Comparator. Before use, the PT/CT must be initialized by writing the Compare Size (CompSize) register with the word size to be compared. This value must be 0,8,16,24, or 32. A value of 0 indicates that no bits will be compared, a value of 16 indicates that the 16 least significant bits will be compared, a value of 32 indicates that the entire word will be compared, etc. After that initialization, the PT/CT can be used to compare plaintext and ciphertext. The plaintext to be verified must be written to the PTChk register first. Upon writing the ciphertext to be verified to the CTChk register, the two registers will be compared. If the comparators find that the plaintext and ciphertext are the same, the exact inverse of each other, or a toggling value of each other, the appropriate Match Counter will be incremented by the number of bits in the CompSize register (in the Bit Bank). If they are not, the appropriate Match Counter will be cleared. There are four counters in the “Match Counters” block, one each to check for encryption by 0x00000000, 0xFFFFFFFF, 0x55555555, and 0xAAAAAAAA.

Due to the way the PT/CT is implemented, it guarantees that it will find an error if 95 successive bits are "bad". However, 64 successive "bad" bits may also generate an error. This is because the logic will compare the entire contents of the PT and CT registers with eachother. For example, with CompSize set to 32:

PT: 0x 12345678 12345678

CT: 0x 12345678 12345678

will produce an error, because 64 bits have matched.

Here's another example with CompSize set to 32:

PT: 0x 12345678 12345678 12345678

CT: 0x 92345678 12345678 12345678

In this case, the error will not be generated until the third set of words is written, since the first word wasn't an exact match. In this case, it took 95 bits matching to generate the error.

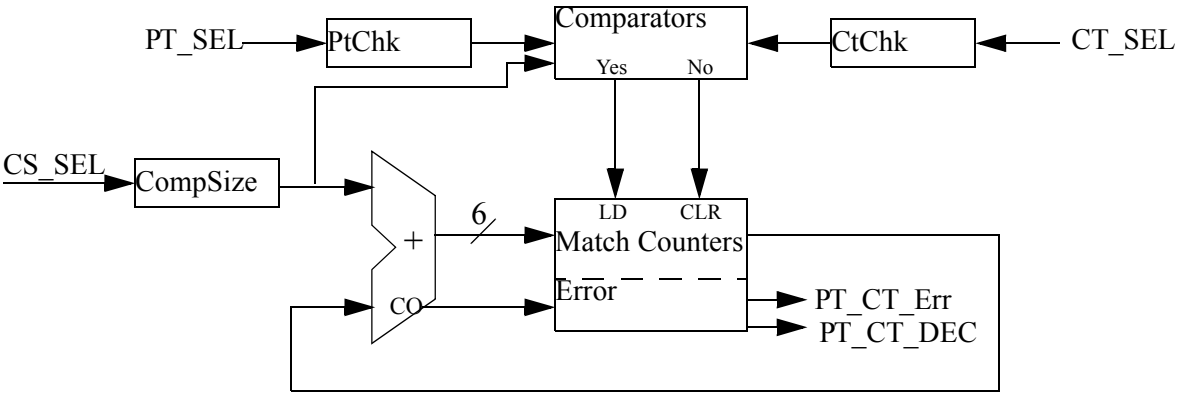


Figure 63-14. PT/CT Comparator Block Diagram

63.4.1.3.4 AIC Example

The following is an example cycle of the AIC with no errors. The software routine being checked by the AIC can produce 16 bits of ciphertext with each algorithm execution. In this example the start value is 0x5CAA, the end value is 0xC397, providing for a 10 step sequence. The Plaintext/Ciphertext Comparator will compare 16 bits of plaintext to ciphertext.

1. The SCP initializes the ASC by writing the appropriate start step value in the Start register (0x5CAA in this example) and the desired final step value in the End register (0xC397).
2. A value of 0x10 (16 decimal) is written to the IncSize register to indicate that 16 bits of ciphertext will be produced,
3. A value of 0x10 (16 decimal) is written to the CompSize register to indicate that 16 bits of plaintext and ciphertext will be compared.
4. The Plaintext data is written into the least significant bits of the PTChk register.
5. The software then writes the first step into the SeqChk register (0x2E55), and continues writing to this register until it writes the final step (0x972A, 0xCB95, 0xE5CA, 0x72E5, 0x3972, 0x1CB9, 0x0E5C, 0x872E, 0xC397). When the software routine finishes an ASC by writing the final step value to the SeqChk register, the BitCnt register is incremented by 16.
6. The ciphertext is written into the CTChk register for plaintext/ciphertext comparison.
7. If the plaintext and ciphertext match, the match counter is incremented by the value in the CompSize register (16), and the other match counters are cleared. The same kind of incrementing or clearing is done to the appropriate match counters if the compared data is inverted, or is toggled (alternating 1’s and 0’s). As long as the match counters have not counted to 64, the Bit Bank will be decremented by the value in CompSize (16).

Steps 3-5 may be repeated as necessary to generate more ciphertext.

63.4.1.4 Timer

The Timer is a 32-bit programmable timer. It is used in conjunction with the Secure State Controller during power-up to ensure that the transition to the Secure state happens in the appropriate amount of time. At power-up, the timer will be loaded with the value 0x06000000 (about 2 seconds with a 50 MHz clock) and will immediately begin counting down. If necessary, the host processor may reset the timer to the desired value for a different power-up timeout value, but the counter must not be stopped. To do this, you would write a new timer value to the TimerIV register, and then write 0x00000003 to the TimerCntl register to load the timer with the new value and keep the timer running. If the Timer is stopped in the Zeroize RAM state, the Secure State Controller will transition to the Failure state. If the Timer is stopped in the Health Check state, the Secure State Controller will transition to the Nonsecure state. Upon transition to the Secure state, the timer will automatically be stopped. After power-up it may be used as a watchdog timer for any time-critical routines or algorithms. If the timer expires, an error is generated. Figure shows the block diagram for the Timer block. The TimerIV register is used to load the initial value for the counter. The TimerCtl register is used to set, start, or stop the timer. When the timer is set and started, the counter will be loaded with the value in TimerIV and will count down. If the counter reaches 0, an Error is generated.

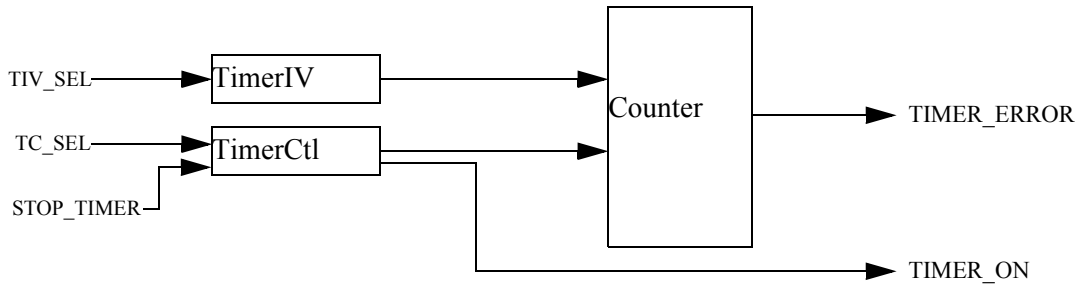


Figure 63-15. Timer Block Diagram

63.4.1.5 Debug Detector

The Debug Detector constantly monitors the various debug and test signals within the platform and alerts the Secure State Controller when any of the test or debug modes becomes active. See Figure . The Secure State Controller will transition to the appropriate state based on the current state, and the Security Policy block will take the appropriate action to alert the rest of the chip and clear up any sensitive data still resident in the chip if necessary.

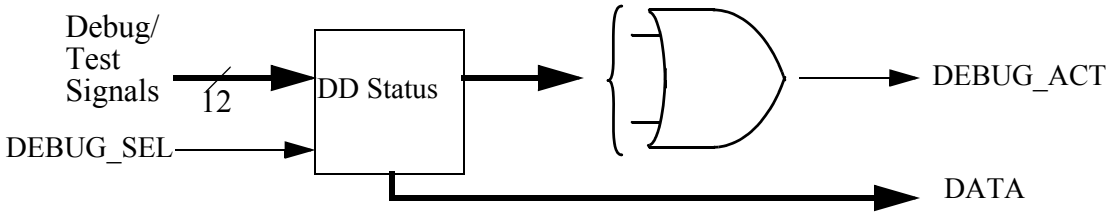


Figure 63-16. Debug Detector Block Diagram

The Debug/Test Signals are six active-high and six active-low inputs to the monitor. They may be connected to relevant debug or test signals. Some possible events requiring detection are JTAG/OnCE, EIM/Show Cycles, and Scan. The DD Status register may be read by the host processor to determine which debug signal is currently active.

63.4.2 Use in a Non-Secure Application

The Security Monitor has been designed to be as non-intrusive as possible, while still maintaining its ability to provide security when needed. One design goal was that if security is not required, and the Security Monitor is not accessed, then it will not interfere with the operation of the platform. This goal was mostly met.

The Security Monitor contains a state machine defining the various security states of the platform. These states are the boot-up states (Start, Zeroize RAM, Health Check), the Non-Secure state, the Secure state and the Failure state. When security is not a requirement, the security state machine will go through the boot-up states and enter the Non-Secure state. Once Secure RAM memory has been initialized, the Security Monitor will allow unrestricted access to it.

Only while in the Secure state, will the Security Monitor monitor for test and debug events, thus clearing the Secure RAM memory and generating an interrupt to the host processor.

In the Non-Secure state, such action will occur only in the event of a recognized hardware failure within the Secure RAM module. But this same hardware failure would cause the Secure RAM memory to no longer be accessible anyway.

The Security Monitor will enter the Secure state only if there is direct action taken by the host processor within 2 seconds of reset. Otherwise, the Security Monitor will enter its Non-Secure state.

63.4.3 Internal Signals Pins

The following are the signals which go between the Secure RAM module and the Security Monitor.

Table 63-19.

Signal	Direction	Connects to	Signal Description
smn_zeroize	input	Security Monitor	Reset encryption module and clear all memory
smn_inhibit_key	input	Security Monitor	Use the default key, not secret key
smn_block_access	input	Security Monitor	Block access to the secret key, KEM and memory
scm_illegal	output	Security Monitor	User mode access attempted.
scm_zero_done	output	Security Monitor	The memory has been cleared and verified
scm_zero_fail	output	Security Monitor	The memory did not verify as cleared.
scm_bad_key	output	Security Monitor	The secret key is stuck at the default key, inverse default key, or one of the defined DES weak keys.
scm_default_key	output	Security Monitor	The default key is being used (not secret key)
scm_internal_err	output	Security Monitor	The built-in assurance tests have detected a failure

SMN_ZEROIZE - This output instructs the Secure RAM to be clear its RAM contents.

SMN_INHIBIT_KEY - This output instructs the Secure RAM to replace the secret key with the default key. This allows the Secure RAM module to be used with a known key, but does not compromise the the Laser ID or other sensitive information.

SMN_BLOCK_ACCESS - This output instructs the Secure RAM module to disables all reads and writes. This condition represents a full lock-down of the Secure RAM, and will be removed only by a hardware reset of the Security Monitor (and the rest of the system).

SCM_ILLEGAL_ACCESS - This input indicates that the Secure RAM was accessed in User mode.

SCM_ZERO_DONE - This input signals that the Secure RAM has been zeroized, and its contents verified to contain the reset value.

SCM_ZERO_FAIL - This input indicates that the Secure RAM was unable to zeroize.

SCM_BAD_KEY - This input indicates that the laser ID being used in the Secure RAM is all 0's, all 1's, the default key, the inverse of the default key, a weak key or an invalid key (a key with invalid code bits). Any of these could indicate that the key has not been programmed, or programmed improperly. The secret key should not be used. This will prevent the chip from entering the secure state, but is not considered fatal and will therefore allow the Secure RAM to remain operational.

SCM_DEFAULT_KEY - This input indicates that the key being used in the Secure RAM is the default key.

SCM_INTERNAL_ERR - This input indicates that an internal hardware error was detected in the Secure RAM. A test is made that the actual key used by the encryption module is the one that should be used. If the wrong key is used for encryption, this is considered an internal error. This signal is connected to SM_ERROR internally.

63.4.4 Register Definitions

KEY: Always Reads One (1) | Always Reads Zero (0) | Read/Write Bit (bit) | Read-Only Bit (bit) | Write-Only Bit (bit) | Write 1 to Clear (bit / W1C) | Self-Clear Bit (0 / bit) | N/A

Table 63-20. SCC Security Module Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STATUS $2D8D_0000	R	VERSION[7:0]								SC	UA	BY	AD	UM	DK	BK	IA
	W																
	R	SM	I	SA	TE	PE	BE	AE	SE	DA	ZF	IB	STATE[4:0]				
	W																
COMMAND $2D8D_0004	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	CI	CB	EI	SA
	W																
SEQSTART $2D8D_0008	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	START VALUE[15:0]															
	W																
SEQEND $2D8D_000C	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	END VALUE[15:0]															
	W																
SEQCHK $2D8D_0010	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CHECK VALUE[15:0]															
	W																
BITCNT $2D8D_0014	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	BIT COUNT[10:0]										
	W																
INCSIZE $2D8D_0018	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	INC SIZE[10:0]										
	W																
BBDEC $2D8D_001C	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	DEC AMT[10:0]										
	W																
COMPSIZE $2D8D_0020	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	SIZE[5:0]					
	W																
PTCHK $2D8D_0024	R	PLAINTEXT[31:16]															
	W																
	R	PLAIN TEXT[15:0]															
	W																
CTCHK $2D8D_0028	R	CIPHER TEXT[31:16]															
	W																

Table 63-20. SCC Security Module Register Summary (Continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	R	CIPHERTEXT[15:0]															
	W																
TIMERIV $2D8D_002C	R	IV[31:16]															
	W																
	R	IV[15:0]															
	W																
TIMERCLT $2D8D_0030	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LD	ST
	W																
DD STATUS $2D8D_0034	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	DC	DB	DA	D[9:1]								
	W																
TIMER $2D8D_0038	R	TIMER[31:16]															
	W																
	R	TIMER[15:0]															
	W																

Table 63-21. Address Map

Address	Name	Read/Write	Protection Mode
00	Status	R/W	Supervisor
04	Command	R/W	Supervisor
08	SeqStart	R/W	Supervisor
0C	SeqEnd	R/W	Supervisor
10	SeqChk	R/W	Supervisor/User
14	BitCnt	R	Supervisor
18	IncSize	R/W	Supervisor
1C	BBDec	W	Supervisor/User
20	CompSize	R/W	Supervisor
24	PtChk	R/W	Supervisor/User
28	CtChk	R/W	Supervisor/User
2C	TimerIV	R/W	Supervisor
30	TimerCtl	R/W	Supervisor
34	DD Status	R/W	Supervisor
38	Timer	R	Supervisor

63.4.4.1 Secure State Controller Registers

63.4.4.1.1 Status (R/W, Supervisor)

The Status Register relays Monitor operational information to the host processor.

STATUS Status Register **Addr $2D8D_0000**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	Version[7:0]								SC	UA	BY	AD	UM	DF	BK	IA
SM	r	r	r	r	r	r	r	r	r	w1c	w1c	w1c	w1c	r	w1c	w1c
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	SM	I	SA	TE	PE	BE	AE	SE	DA	ZF	IB	Strate[4:0]				
TYPE	w1c	r	w1c	w1c	w1cr	w1c	w1c	w1c	r	w1c	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-22. Status Description

Field	Description	Settings
Version Bits 31–24	**Version** — Current version of the Security Monitor.	0x01 = First Version
SC Bit 23	**Scan Exit** — Reflects the value of the Scan Exit input pin.	1 = Scan mode used and exited since last reset 0 = Scan mode has not been used since reset
UA Bit 22	**Unaligned** — This bit may be cleared by writing it to a 1. Neither ARM nor MCore will generate a misaligned word access. If a word access to an odd address occurs, then the lower two bits are to be ignored, (and treated as 00). This UA bit will always read as 0 for ARM and MCore platforms, and is here for future platforms that may generate misaligned accesses.	1 = An access to an unaligned address was attempted 0 = An access to a word-aligned address was attempted
BY Bit 21	**Byte** — This bit may be cleared by writing it to a 1.	1 = A byte or half-word access has been attempted 0 = A word access has been attempted
AD Bit 20	**Invalid Address** — This bit may be cleared by writing it to a 1.	1 = Invalid Address access attempted 0 = Valid address access
UM Bit 19	**User Mode** — This bit may be cleared by writing it to a 1.	1 = User Mode access attempted 0 = Supervisor Mode access
DF Bit 18	**Default Key —**	1 = Secure RAM is using Default Key 0 = Secure RAM is using secret key

Table 63-22. Status Description

Field	Description	Settings
BK Bit 17	**Bad Key** — This bit may be cleared by writing it to a 1.	1 = Secure RAM key is a weak or bad key 0 = No error has been detected
IA Bit 16	**Illegal Access** — This bit may be cleared by writing it to a 1.	1 = Illegal Access has been attempted to Secure RAM 0 = No illegal access has been detected
SM Bit 15	**Secure RAM Error** — This bit may be cleared by writing it to a 1.	1 = Internal hardware error in Secure RAM 0 = No error has been detected
I Bit 14	**Interrupt**	1 = Interrupt Asserted 0 = Interrupt Deasserted
SA Bit 13	**Software Alarm** — This bit may be cleared by writing it to a 1.	1 = The host processor notified the Monitor of a security error 0 = No error has been detected
TE Bit 12	**Timer Error** — This bit may be cleared by writing it to a 1.	1 = The Timer has expired 0 = No error has been detected.
PE Bit 11	**PT/CT Error** — This bit may be cleared by writing it to a 1.	1 = The Plaintext/Ciphertext Comparator has detected 64 bits of related plaintext and ciphertext 0 = No error has been detected.
BE Bit 10	**Bit Bank Error** — This bit may be cleared by writing it to a 1.	1 = The Bit Bank detected an overflow or underflow error 0 = No error has been detected.
AE Bit 9	**ASC Error** — This bit may be cleared by writing it to a 1.	1 = The Algorithm Sequence Checker detected a sequence error 0 = No error has been detected.
SE Bit 8	**Security Policy Error** — This bit may be cleared by writing it to a 1.	1 = The Security Policy block detected an error 0 = No error has been detected.
DA Bit 7	**Debug Active** — This bit may be cleared by clearing all bits in the DDStatus register	1 = One or more debug signals is active 0 = No debug signals active
ZF Bit 6	**Zeroize Failed** — This bit may be cleared by writing it to a 1.	1 = Zeroize Failed 0 = Zeroize has not failed
IB Bit 5	**Internal Boot** — Boot Mode of processor	1 = Processor booted from internal ROM 0 = Processor booted from external flash
State Bits 4–0	**State** — Encoded value of the Secure State Controller State Machine	00000 = Start 00101 = Zeroize RAM 00110 = Health Check 01001 = Failure 01010 = Secure 01100 = Non-Secure

63.4.4.1.2 Command (R/W, Supervisor)

The Command Register is used by the host processor to give commands to the Monitor.

COMMAND Command Register **Addr $2D8D_0004**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
													CI	CB	EI	SA
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	slfclr	slfclr	r/w	slfclr
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-23. Command Description

Field	Description	Settings
CI Bit 3	**Clear Interrupt** — Self-Clearing Bit:	1 = Clear Interrupt 0 = Don't clear Interrupt
CB Bit 2	**Clear Bit Bank** — Self-Clearing Bit:	1 = Clear BitCnt Register 0 = Don't clear BitCnt Register
EI Bit 1	**Enable Interrupt** — Interrupt is always enabled in Secure State, but may be masked in any other state.	1 = Interrupt Enabled 0 = Interrupt Disabled (Masked)
SA Bit 0	**Software Alarm** — Self-Clearing Bit:	1 = The host processor notifies the Monitor of a security error, causing the monitor to transition to the Failure state 0 = No error has been detected

63.4.4.2 Algorithm Sequence Checker Registers

63.4.4.2.1 SeqStart (R/W, Supervisor)

The Sequence Start register initializes the LFSR in the ASC and clears the Bit Bank.

SEQSTART Sequence Start Register **Addr $2D8D_0008**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	STARTVALUE[15:0]															
TYPE	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-24. SEQSTART Description

Field	Description	Settings
Start Value Bits 15–0	**Start Value** — ASC start value	

63.4.4.2.2 SeqEnd (R/W, Supervisor)

The Sequence End register stores the final step value for an algorithm being checked by the ASC.

SEQEND Sequence End Register **Addr $2D8D_000C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	ENDVALUE[15:0]															
TYPE	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 63-25. SEQEND Description

Field	Description	Settings
End Value Bits 15–0	**End Value — Final ASC step value**	

63.4.4.2.3 SeqChk (R/W, Supervisor/User)

The Sequence Check register is used by the software to compare its intermediate step values with those calculated by the ASC LFSR.

SEQCHK Sequence Check Register **Addr $2D8D_0010**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	CHECKVALUE[15:0]															
TYPE	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-26. SEQCHK Description

Field	Description	Settings
Check Value Bits 15–0	**Check Value** — ASC intermediate step value	

63.4.4.3 Bit Bank Registers

63.4.4.3.1 BitCnt (R, Supervisor)

The Bit Count register is the Bit Bank repository. When an ASC sequence finishes successfully, this register is incremented. When Plaintext and Ciphertext are successfully compared in the PT/CT, this register is decremented. BitCnt may be read, decremented, or cleared by the host processor but may not be incremented or loaded with a non-zero value.

BITCNT Bit Blank Register **Addr $2D8D_0014**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
						BITCOUNT[10:0]										
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-27. BITCNT Description

Field	Description	Settings
Bit Count Bits 10–0	**Bit Bank** — Number of bits in the Bit Bank	

63.4.4.3.2 IncSize (R/W, Supervisor)

The Increment Size register contains the number of bits by which the Bit Bank should be incremented upon completion of an ASC sequence.

INCSIZE Increment Size Register **Addr $2D8D_0018**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
						INCSIZE[10:0]										
TYPE	r	r	r	r	r	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-28. INCSIZE Description

Field	Description	Settings
Inc Size Bits 10–0	**Increment Size** — Number of bits by which to increment the Bit Bank	

63.4.4.3.3 BBDec (W, Supervisor/User)

The Bit Bank Decrement register provides a means to manually subtract bits from the Bit Bank.

BBDEC Bit Bank Decrement Register **Addr $2D8D_001C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
						DECAMT[10:0]										
TYPE	r	r	r	r	r	w	w	w	w	w	w	w	w	w	w	w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-29. BBDEC Description

Field	Description	Settings
Dec Amt Bits 10—0	**Bit Bank Decrement** — Number of bits by which to manually decrement the Bit Bank	

63.4.4.4 PT/CT Registers

63.4.4.4.1 CompSize (R/W, Supervisor)

The Compare Size register contains the number of bits that should be compared by the PT/CT as well as the number of bits by which to decrement the Bit Bank upon a valid PT/CT compare.

COMPSIZE Compare Size Register **Addr $2D8D_0020**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
											SIZE[5:0]					
TYPE	r	r	r	r	r	r	r	r	r	r	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-30. COMPSIZE Description

Field	Description	Settings
Size Bits 5–0	**Compare Size** — Number of bits to compare/decrement	

63.4.4.4.2 PTChk (R/W, Supervisor/User)

The Plaintext Check register stores Plaintext that will be used in the Plaintext/Ciphertext Comparator.

PTCHK Plaintext Check Register **Addr $2D8D_0024**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	PLANTEXT[31:16]															
TYPE	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	PLANTEXT[15:0]															
TYPE	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-31. PTCHK Description

Field	Description	Settings
Plaintext Bits 31–0	**Plaintext Check** — Plaintext to be compared	

63.4.4.4.3 CTChk (R/W, Supervisor/User)

The Ciphertext Check register stores Ciphertext that will be used in the Plaintext/Ciphertext Comparator. Writing to the CTChk register executes the compare between the PTChk and CTchk registers.

CTCHK Ciphertext Check Register **Addr $2D8D_0028**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	CIPHERTEXT[31:16]															
TYPE	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	CIPHERTEXT[15:0]															
TYPE	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-32. CTCHK Description

Field	Description	Settings
Ciphertext Bits 31–0	**Ciphertext Check** — Ciphertext to be compared.	

63.4.4.5 Timer Registers

63.4.4.5.1 TimerIV (R/W, Supervisor)

The Timer Initial Value register is used to set the timer length. The counter will be loaded with this value when the LD value is written to the TimerCtl register and count down to zero.

TIMERIV Timer Initial Value Register **Addr $2D8D_002C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	IV[31:16]															
TYPE	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	IV[15:0]															
TYPE	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-33. TIMERIV Description

Field	Description	Settings
IV Bits 31–0	**Timer Initial Value** — Initial Value for the Timer.	

63.4.4.5.2 TimerCtl (R/W, Supervisor)

The Timer Control register is used to set, start, or stop the timer. One command may be given to load and start the timer, i.e. 0x00000003.

TIMERCTL Timer Control Register **Addr $2D8D_0030**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
															LD	ST
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	slfclr	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-34. TIMERCTL Description

Field	Description	Settings
ST Bit 1	**Start/Stop** — Start/Stop Bit	1 = Start 0 = Stop
LD Bit 0	**Load** — Load the value of TimerIV into the counter, self-clearing bit	1 = Load the value of TimerIV into the counter

63.4.4.5.3 Timer (R, Supervisor)

The Timer register is used to read the value of the timer. The Timer is reset to a value of approximately 2 seconds (0x06000000).

TIMER Timer Register **Addr $2D8D_0038**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	TIMER[31:16]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	TIMER[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-35. TIMER Description

Field	Description	Settings
Timer Bits 31–0	**Timer** — Value of 32-bit down counter.	

63.4.4.6 Debug Detector Register

63.4.4.6.1 DD Status (R/W, Supervisor)

The Debug Detector Status register shows which debug/test conditions are currently active.

DDSTATUS Debug Detector Status Register **Addr $2D8D_0034**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
					DC	DB	DA	D9	D8	D7	D6	D5	D4	D3	D2	D1
TYPE	r	r	r	r	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 63-36. DDSTATUS Descriptions

Field	Description	Settings
DC Bit 11	**DebugC** — This bit may be cleared by writing it to a 1	1 = Debug 12 Active (i.e. debug12_b=0) 0 = Debug 12 Inactive
DB Bit 10	**DebugB** — This bit may be cleared by writing it to a 1	1 = Debug 11 Active (i.e. debug11_b=0) 0 = Debug 11 Inactive
DA Bit 9	**DebugA** — This bit may be cleared by writing it to a 1	1 = Debug 10 Active (i.e. debug10_b=0) 0 = Debug 10 Inactive
D9 Bit 8	**Debug9** — This bit may be cleared by writing it to a 1	1 = Debug 9 Active (i.e. debug9_b=0) 0 = Debug 9 Inactive
D8 Bit 7	**Debug8** — This bit may be cleared by writing it to a 1	1 = Debug 8 Active (i.e. debug8_b=0) 0 = Debug 8 Inactive
D7 Bit 6	**Debug7** — This bit may be cleared by writing it to a 1	1 = Debug 7 Active (i.e. debug7_b=0) 0 = Debug 7 Inactive
D6 Bit 5	**Debug6** — This bit may be cleared by writing it to a 1	1 = Debug 6 Active (i.e. debug6=1) 0 = Debug 6 Inactive
D5 Bit 4	**Debug5** — This bit may be cleared by writing it to a 1	1 = Debug 5 Active (i.e. debug5=1) 0 = Debug 5 Inactive
D4 Bit 3	**Debug4** — This bit may be cleared by writing it to a 1	1 = Debug 4 Active (i.e. debug4=1) 0 = Debug 4 Inactive

Table 63-36. DDSTATUS Descriptions

Field	Description	Settings
D3 Bit 2	**Debug3** — This bit may be cleared by writing it to a 1	1 = Debug 3 Active (i.e. debug3=1) 0 = Debug 3 Inactive
D2 Bit 1	**Debug2** — This bit may be cleared by writing it to a 1	1 = Debug 2 Active (i.e. debug2=1) 0 = Debug 2 Inactive
D1 Bit 0	**Debug1** — This bit may be cleared by writing it to a 1	1 = Debug 1 Active (i.e. debug1=1) 0 = Debug 1 Inactive

63.5 DFT and Scan Testing

The Secure RAM may contain sensitive information which should not be accessible in any mode of operation, normal mode or test mode. This need for security of data places a limit on the openness of the design, and hence ease of observing some internal state information. One of the design goals is to have a module which is capable of securely storing sensitive information, while at the same time, being testable. This section will describe the information which cannot be allowed to be observed, and a proposal for a test strategy.

Although the Security Monitor does not contain sensitive information whose access must be denied during scan testing, it does contain state information which must not be allowed to be set in normal operation mode, by scanning in new state.

63.5.1 Sensitive Information

At times, the Secure RAM may hold what is considered to be sensitive information. At boot time, the RAM goes through an initialization proceedure to clear out any old information. After initialization, the RAM may be read and written freely, to allow testing to procede. There also exist some internal registers which at times hold the memory contents, and which therefore need to be reset before testing may begin.

The other information which cannot be allowed to leave the chip, is the 168-bit secret key. This value is programmed using LaserID bits, and is input only to the Secure RAM module. Internally, this secret key is used only as the key input to the encryption block. There is a blocking mux on this secret key, so that at times a default key is used. During all testing, the secret key is hidden, and the default key is used. Because the default key has a known value, all scan and functional tests will have a known result. The down side is that all internal logic for testing the secret key becomes untestable. After the secret key has been programmed and the chip packaged, there is no way for the secret key to be observed. This makes it difficult to prove that the secret keys are indeed being programmed correctly.

The Security Monitor does not hold sensitive information, but must not be allowed to have security state machine placed into a given state and then put back into normal operation. The testing problem is slightly different, but the same solution will hopefully be used for both modules.

63.5.2 Scan Testing

During scan testing, the RAM is disabled (along with the secret key), so reading from the RAM is not a problem. However, it is possible for the RAM data to have been previously read out, and stored in internal registers. Therefore, the Secure RAM module will cause a reset to occur when the scan_mode signal is

asserted. After this reset occurs, full scan testing can begin. The only logic which is expected to not be testable is the logic put in to check the validity of the secret key. This is because the secret key needs to be protected, but also because the secret key inputs cannot be controlled by scan testing.

63.5.3 Scan Test Controller

In order to allow full scan testing of the Security Controller module, (i.e., put all flops on scan chains), we have come up with a small scan test controller. This circuit is not scan testable itself, but should allow all of the Security Controller flops to be placed on scan chains. The scan test controller currently uses two flops to determine that scan testing has occured and prevent the Security Controller from re-entering a normal operation mode. The threat we are trying to prevent is putting the Security Controller into a state such that the data in memory can be read, or allowing access to the secret key.

This scan test controller will be instantiated in the Platform scan wrapper.

The scan test controller performs an edge detection of the scan_mode signal. When the rising edge of scan_mode is detected, scan_reset_b is asserted from the rising edge of scan_mode until the second rising edge of the clock. scan_reset_b is used to reset flops which can contain sensitive information (i.e., the encryption module working register, and the register holding data read from the memory). On the falling edge of scan_mode, scan_exit is asserted until either scan_mode is again asserted, or until system reset.

The two flops in the scan test controller are not on scan chains, and will only be reset when not in scan mode.

After scan testing is completed, the scan test controller will prevent the Security Monitor from operating normally. The only way to get back to normal mode will be to reset the module first.

63.5.4 Testing the secret key test logic

Logic is included in the Security Controller to test the secret key. This is done for two purposes. The first is to simply check that the key has been programmed. It is also checked to verify that the data that has been programmed is stable (i.e., does not have different values each time the chip is powered up).

The logic testing the secret key generates a single signal, bad_key. Because the secret key is by necessity protected from being read or modified, this test logic has 168 inputs which cannot be controlled. To allow the secret key test logic to be tested during scan testing, the 168 inputs are put through scan muxes. 56 registers are added (which each register connected to 3 scan muxes), which can be modified during scan mode to test the secret key test logic, but without changing the secret key as it goes to the encryption module. 56 bits were chosen as a compromise between too great an addition of area and not enough test coverage.

63.6 Security Features

The following table lists some of the debug and test modes which can be used for accessing sensitive data. It also gives some information about how the Security Controller will attempt to prevent sensitive data from being accessed.

Table 63-37.

Access to chip	Eventual threat regarding security	Implemented protection scheme
Show Cycles (EIM)	Visibility of AHB bus through EIM pads.	Whenever Show Cycles is activated, the signal weim_show_cycle_on is asserted by EIM and the Security Monitor will go into the non-secure or failure state.
External Boot	Read the contents of the red RAM, or try to use KEM to de/encrypt data.	The Security Monitor enters the non-secure state when detecting external boot via its input pin int_boot = platform signal indicating external boot.
Watchpoint	Visibility of AHB bus via the comparators.	The watchpoint can be enabled or modified only under software control. The software in the flash is certified using the HAC block. If the watchpoint would be modified by some access via the JTAG interface, the jtag_active signal gets asserted and puts the Security Monitor into non-secure or failure state.
JTAG IEEE1149.1	Not considered as a threat.	
JTAG long chain (scan divergence)	Implemented for the Guinness platform, but not supported by JTAG-TEST POG block. Security concern is TBD on ARM platform.	
JTAG - Once Debug mode.	Through the Once it is possible to put the MCU in debug mode and use it to run User Code that can be dangerous (dump contents of RAM, use KEM to de/encrypt data, ...)	When the signal jtag_active[1] is asserted, the Security Monitor will go to the non-secure or failure state. jtag_active needs to be defined for ARM platform.
Nexus interface	Read secure RAM contents via Nexus interface.	After reset, the Nexus interface is in the “no trace” mode, and no data can be traced. In order to trace data, the Nexus interface has to be configured via JTAG, which asserts the jtag_active signal, and puts the Security Monitor into non-secure or fail state.
Test: Scan the scc_top module[2]	1. Go from normal operation to scan test mode and shift out sensitive data RAM contents or the Laser ID). 2. Modify the Security Monitor Status.	1.As JTAG is used to activate the scan mode, the jtag_active signal will put the Security Monitor into non-secure or failure state. 2. When entering scan test mode (scan_mode signal from guinness_test) all sensitive data will be cleared from the security modules. There will be no path from the secret Laser ID to flops on the scan chain, and the RAM contents are inaccessible. 3. When exiting scan test mode (scan_exit signal from guinness_test), the Security Monitor transitions to the failure state.
Test: Scan the neighboring modules[2]	Pilot the SCC module in a state where the secured RAM can be read, or the KEM can be used to de/encrypt data	As JTAG is necessary to activate the scan mode, the jtag_active signal will put the Security Monitor into non-secure or failure state.
Test: Memory Bist[2]	RAM bist could be used to read sensitive RAM contents.	The RAM Bist always starts with a write before reading the RAM, so there is no threat. The “bit mapping” test is only possible while the SCC is still disabled via a fuse.[3]

Table 63-37.

Access to chip	Eventual threat regarding security	Implemented protection scheme
Alternate Bus Masters	Alternate Bus Masters (like IPCM) can dump contents of RAM, use KEM to de/encrypt data, ...	An access to the SCC is only possible in Supervisor mode. If ARM platform grants Supervisor access to an alternate bus master, it can access the secure RAM contents. However, if an alternate master is commanded from outside, a JTAG access is necessary which asserts the jtag_active signal and puts the Security Monitor into non-secure or failure state.

1. Due to potential noise on pads (EMI at board level, no schmitt trigger), POG JTAG generates the signal jtag_active which tells the security modules that there has been a JTAG access: this will cover all JTAG and special test modes. This signal is TBD in the ARM platform.
2. The signal jtag_active monitors all test modes, but it will not disturb the activity of this test mode. Only the default encryption key is available in the test modes.
3. For this reason, the BIST test should be run at wafer probe test, before the fuse is blown.

Chapter 64
Security Laser ID Module (SLID)

Revision History Table

Revision	Date	Author	Changes
0.1	25 Jun 2001	Glen Zoerner	initial release
0.2	25 Jun 2001	Glen Zoerner	remove 4 module enable fuses and ports remove reference to 16-bit CRC
0.3	25 Jul 2001	Glen Zoerner	add polarity of output signals
0.4	05/07/03		Updated for LTE specification release.

64.1 Description

The Security Laser ID module (SLID) provides a laser programmable security key for cryptographic purposes. It consists of 168 laser fuses and associated sensing and latch circuitry. This module is not scannable and has no connections to an MCU bus structure; the 168 wires route directly to the cryptographic block.

The value that is programmed into the fuses is a cryptographically strong random number. The algorithm used to generate the value is beyond the scope of this document. As there is no mechanism for checking the correctness of the programmed value, it is suggested that a check-sum value be embedded in the programmed value. Circuitry that uses the random number is responsible for determining the correctness of the value (in Neptune this is the Secure RAM module). Note that the actual value cannot be verified. The checksum indicates that all 168 bits have been programmed correctly but does not indicate the value itself.

The polarity of the signals on the slid_security_key bus is: 1 = fuse blown, 0 = fuse intact.

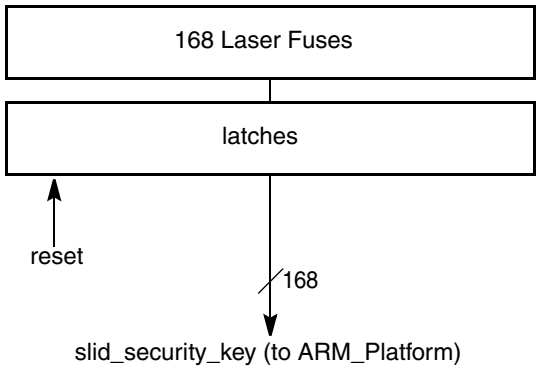


Figure 64-1. SLID Block Diagram

64.1.1 SLID Module Pin List

Table 46-1 is a list of the SLID module pins.

Table 64-1. IIM Module Pin List

Pin Name	Direction	Description		
slid_security_key[167:0]	output	Cryptography security key 1 = fuse blown 0 = fuse intact		
reset_b	input	active low reset		

Chapter 65
MCU to Dual-Port-Ram Peripheral Interface (MDPI)

Revision History Table

Revision	Date	Author	Changes
0.1	01/21/01	Scott Herzog	Initial Release
0.2	03/04/02	Scott Herzog	Major design change - removed state machine and replaced with 3 flops to control interface. Other minor logic changes...
0.3ahb	04/18/02	Randall Metzger	Additional logic added for ahb interface. Signal name changes. And, changes to waveform diagrams.
0.4	05/07/03		Updated for LTE specification release.

65.1 Overview

The MCU to Dual-Port_ram Interface (MDPI) module provides the translation of bus control activity between the MCU (Mcore) and a Dual Port RAM (DPR), which is intended as shared memory between the DSP and the MCU. The DPR is configured as two separate blocks of memory (X and Y RAM) and the MDPI must coherently manage the control of data between the MCU and the DPRs. In addition, the MDPI interface allows the MCU to read/write words(32-bit) to the 16-bit DPR memories. This allows one single 32 bit accesses with 1 wait state per access versus two separate 0 wait state, 16-bit accesses which would typically be less efficient. Note: byte accesses are not supported as they are not supported by the DPR.

Figure 1 shows the block diagram for the MDPI. It is comprised of the Bus Interface State Machine (BISM) block and the MDPI Multiplexor Block (MMB)

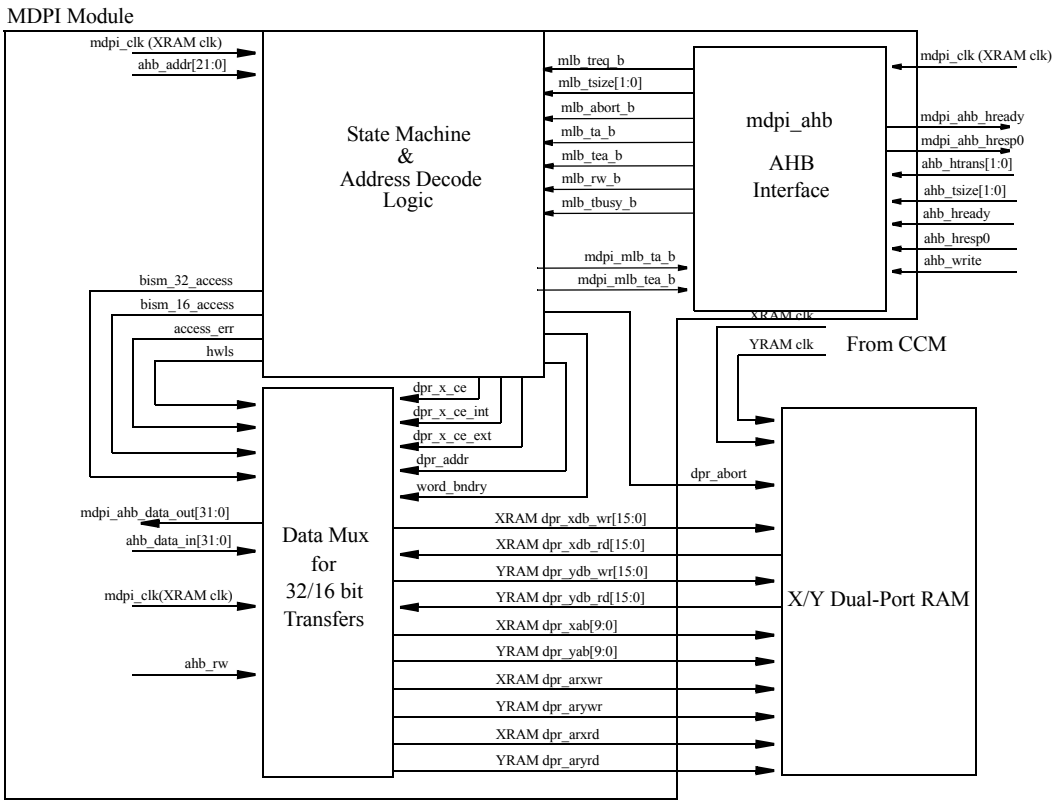


Figure 65-1. MDPI Module Block Diagram (exclude X/Y DPR)

65.2 Bus Interface State Machine Block

The Bus Interface State Machine (BISM) block (refer to figure1) of the MDPI is designed to be compatible with the Mcore architectures, but will easily be modified for ARM based applications (Neptune). The slave interface within the MDPI is designed to accommodate a 32-bit access to the 16-bit memory (adding 1-wait state during this mode of operation). The Bus Interface block is made up of the Bus Interface State Machine (BISM), address generation block, address decode block to determine X or Y RAM accesses, error generation logic and miscellaneous control logic.

Control Signal Description (between blocks, excluding bus interface control signals):

1. **hwls:** This signal (half word lane select) is essentially the output of the bism_32_access flop. It tells the Multiplexor block when to switch it's "data" muxes for 32-bit access mode only. It directs the most significant half word of data through the muxes during the first access of a 32-bit operation. In 16-bit access mode this signal is not asserted.
2. **dpr_x_ce**: This control signal is used in the Multiplexor block for latching read data back from the DPR in 32-bit mode, data muxing back to MCU from DPR on second half of access during 32-bit mode and on 16-bit accesses and read/write control signal muxing to the appropriate X-RAM or Y-RAM.
3. **dpr_x_ce_int:** This is the validated select for X-RAM accesses. It is generated by combinational logic from mdpi_sel_valid and decoded address off the ahb_addr input. It is only valid for the initial cycle of the current access after the previous address has been taken. It is used as the input to the dpr_x_ce flop and for mux control of the read/write control strobes to the X/Y RAMs.

4. **dpr_x_ce_ext:** This control signal is only valid during the second half of a 32-bit access. It is a flop that gets it's source from dpr_x_ce. It is used to provide data mux control from the DPR to MCU during the second half of accesses during 32-bit mode.
5. **bism_16_int:** This is the validated 16-bit access control signal. It is generated by combinational logic from mdpi_sel_valid and decoded ahb_tsiz during the initial cycle of the current access after the previous address has been taken (ahb_ta = 1'b1 or ahb_tea = 1'b1). It is used as the validated input to the BISM flop, bism_16_access.
6. **bism_16_access:** This flop gets its source from bism_16_int and is only asserted for one cycle provided a valid 16-bit access is in progress. This control signal is used to gate on the signals mdpi_ta_b or mdpi_tea_b during 16-bit mode and data mux control during 16-bit writes to the X/Y RAMs.
7. **bism_32_int:** This is the validated 32-bit access control signal. It is generated by combinational logic from mdpi_sel_valid and decoded mlb_tsiz during the initial cycle of the current access after the previous address has been taken (ahb_ta = 1'b1 or ahb_tea = 1'b1). It is used as the validated input to the BISM flop, bism_32_access.
8. **bism_32_access:** This flop gets its source from bism_32_int and is only asserted for one cycle provided a valid 16-bit access is in progress. This control signal is used to gate on the error response signal, mdpi_tea_b, during 32-bit mode and data mux control (via hwls) during 32-bit writes to the X/Y RAMs.
9. **bism_32_extend:** This flop gets its source from bism_32_access. This control signal is used to gate on the response signal, mdpi_ta_b, during 32-bit mode and used for data mux control during 32-bit reads/writes to the X/Y RAMs.
10. **dpr_addr**: This is the decoded address for the X/Y RAM. It is massaged from the registered version of the ahb_addr input to accommodate the addressing for the DPR in 32/16-bit addressing modes and is routed to the MCB to guide it to the appropriate DPR being accessed.
11. **oob_error:** This internal error signal is generated when an access exceeds the 1K x 16 boundary limit of the DPRs and is generated by sampling the incoming address against the address range for this particular ram size being used and is registered for generating a tea_b.
12. **byte_err:** This signal is created combinationally during the initial access of a valid access if the access is a byte operation.
13. **bism_err:** This error control signal is a flop that gets its source from oob_err or byte_err inputs. It is always generated on the second cycle of an access and is used to generated the bus error response, mdpi_mlb_tea_b.
14. **access_err:** This control signal is the combinational "OR" of byte_err, oob_err and bism_err. It is used to prevent the read/write strobes from asserting during these error conditions to prevent accesses to the DPR.
15. **access_taken:** This signal is generated through the combinational logic made up of ahb_tbusy_b (ahb_htrans[1:0]), ahb_treq_b, ahb_ta and ahb_tea. If an ahb master has started a transfer on the bus (ahb_treq_b ==1'b0) or it has the bus (ahb_treq_b ==1'b0 and ahb_tbusy_b==1'b0) and the previous access has been taken (ahb_ta (hready)==1'b1 or ahb_tea (hresp0)==1'b1) access taken will assert allowing the MDPI to continue handling the current access request.
16. **word_bndry:** This control signal is needed for the MMB to direct data from the MCU to the DPR on mcu writes through the muxes. It's primary use is on 16-bit writes to word boundary locations to guide the data from the upper halfword of the ahb data bus (ahb_data[31:16]) to the dpr data bus.
17. **mdi_sel_valid:** This mdpi-enable signal is generated from the combinational "and" of the input mdpi_sel (from the readmux) and access_taken. If both are true, mdpi_sel_valid asserts indicating a valid access for the current operation. This signal is only valid for the initial cycle of the current access (address phase). It is used for a variety of functions: a) To gate the address latch for 32-bit operation, b) To mux the non-latched address off the mcu address bus to the appropriate X/Y RAM, c) To generate a valid dpr_x_ce_int, d) To generate a valid oob_err, e) To generate a valid byte_err, f) To generate a valid bism_16_init, g) To generate a valid bism_32_init, h) To provide a valid read or write strobe to

the X/Y RAMs during the first cycle of an access, i) To provide a "valid" control to latch the read/write control from the mcu.

The heart of the bus interface state machine consists of 3 major flip flops(bism_16_access, bism_32_access and bism_32_extend(hwls)). The control logic is described as follows (reference Figure 3 and Figure 5, MDPI_BISM Timing diagram, for details)

BISM_16_access flop:

This flop latches valid 16-bit accesses selects. When the MDPI is selected, a valid access is generated (mdpi_sel_valid) and the access is a half word(bism_16_init), this flop will be set and active for the next cycle. If there is not a 16-bit access following the current access, this flop will negate on the next clock edge. The output of this flop is used to generate a variety of activity within the rest of the MDPI logic. The following is a list of control signals that it controls:

1. mdpi_tea_b: With an MCORE device, if an error occurred within the current access, the error acknowledge response is routed back to the MLB's tea_b, which will be asserted while this flop is set in 16-bit mode. It occurs on the second cycle of the 16-bit access. With an ARM7 device, this signal is internal to the MDPI and helps generate mdpi_hresp0 and mdpi_hready for the AHB.
2. mdpi_ta_b: With an MCORE device, this signal is routed back to the MLB's ta_b. On the second cycle of the current access, mdpi_ta_b is turned on when the BISM_16_access flop gets set. With an ARM7 device, this signal is internal to the MDPI and helps generate mdpi_hready for the AHB.
3. dpr_xdp_wr/dpr_ydb_wr: Provides the multiplexer control during 16-bit mode for the write data to the XRAMs.

BISM_32_access flop:

This flop latches valid 32-bit accesses selects for controlling internal MDPI logic in 32-bit mode. When the MDPI is selected (assertion of input mdpi_sel), a valid access is generated (mdpi_sel_valid) and the access is a word, this flop will be set in the second cycle of the 3 cycle access for 32-bit mode. If there is not a 32-bit access following the current operation, this flop will negate on the next clock edge. The output of this flop is used to generate a variety of activity within the rest of the MDPI logic. The following is a list of control signals that it generates/controls:

1. hwls: This control signal is just renamed from the flop for the BISM_32_access signal. It is used for control within the MMB. The signal, hwls controls the registering of the second half word of data off the ahb data bus for 32-bit mode writes, it controls the registering of the second half word of data from the DPRs to the ahb data bus, it controls the muxing of the first half word of the access through the muxes in the MMB and it provides an enable the read and write strobes to the DPRs in 32-bit mode.
2. The signal, bism_32_access controls logic within the bism module. It controls the assertion of the bism_32_extend flop and the assertion of the dpr_x_ce_ext flop. It also gates the mdpi_tea_b during an abort response.

BISM_32_extend flop:

This flop samples the output flop bism_32_access. When bism_32_access asserts, this flop will assert on the following clock cycle. It will remain asserted until it sees the MDPI inputs, ahb_ta (hready) or ahb_tea (hresp0) assert (indicating that the MDPI's access has been recognized and terminated) and on the next rising edge bism_32_extend will negate. The following is a list of control signals that it controls:

1. Asserts the mdpi_ta_b acknowledge output when in 32-bit mode.
2. Muxes the lower half word of data from the MCU to the DPRs during the second cycle of a 32-bit operation.
3. Muxes the latched DPR data (from the first cycle access of a 32-bit read operation) onto the MCU data bus and muxes the DPR data from the second cycle of a 32-bit read operation onto the MCU data bus.

Miscellaneous Control logic:

Address Generation Logic:

This logic will take the address off the ahb_addr bus, massage it and pass it onto the MMB where it will be directed to both X/Y RAMs. To coherently access the 16-bit addressable memory using the ahb_addr bus, the least significant bit of the ahb_addr bus must be dicarded and the remainder of the address sent to the RAMs. For 16-bit accesses and the first half of an access of a 32-bit access, this massaged address from the ahb_addr bus is passed right to the RAMs to setup the first address to be clocked into the RAMs before the first edge of cycle S1. For 32-bit operation mode only, this address will be latched on the first clock cycle(S1) (see Figure 5, MDPI_BISM Timing Diagram) to prepare the address for the second half of the access on the next clock edge of cycle S2. This second address for the RAMs will just have the lsb of the latched address "set" to access the next memory location (as the 2-lsb's of the ahb_addr bus should always be zero if accessing a 32-bit location on a word boundary).

Bus Error Generation Logic:

This logic will generate an error response (assertion of mdpi_tea_b) if it's access violated certain conditions. There are basically three conditions within the MDPI that can cause an error response to occur:

1) If an address is detected that exceeds the boundary limits of the X/Y RAMs (1K x 16 in the case for Patriot_indy), an "out-of bounds" (OOB) error will occur and cause mdpi_tea_b to assert (ending the cycle) and block the mdpi_ta_b from asserting.

2) If the input ahb_abort_b is asserted the cycle after the MDPI's access, this will cause mdpi_tea_b to assert (ending the cycle) and block the mdpi_ta_b from asserting. The ahb_abort_b, if asserted, will occur in the data phase of the current access. Therefore, on a 16-bit access, it will occur in the Term_access state and move to the BISM will move to the IDLE state while the tea logic will be asserted. On a 32-bit access, it will occur in the XFR_32 state. During this state the BISM looks for an abort and if it is detected, the BISM will move to the IDLE state and the mdpi_tea_b logic will be asserted.

3) If the input control signal, ahb_tsiz == 2'b00 (byte access), this will cause byte_err to assert which will cause mdpi_tea_b to assert (ending the cycle) and block the mdpi_ta_b from asserting.

AHB Logic (for the ARM7):

Gasket logic to interface MDPI (mlb design) to the ARM7's AHB. Signals are as follows:

1. mdpi_tea_b_reg: This is the inverted and registered (flop) version of mdpi_tea_b (refer to figure 2)
2. mdpi_ahb_hresp0: If an access error occurs, this signal gets set for two cyles. It is mdpi_tea_b_reg or gated with ~mdpi_tea_b (refer to figure 2).
3. mdpi_ahb_hready: If no access error occurs, this signal is just mdpi_ta_b inverted. If an access error occurs, this signal gets set for the second cylce of mdpi_hresp0's activation. mdpi_hready is just ~mdpi_ta_b muxed with mdpi_tea_b_reg, controlled with mdpi_ahb_hresp0 (refer to figure 2).
4. ahb_htrans[1:0]: ahb's transfer type - when a SEQ or NSEQ occurs, the mdpi's mlb_treq_b is pulled low.
5. ahb_hready and ahb_hresp0: Used to generate mlb_ta_b and mlb_tea_b
6. mlb_treq_b_idle_busy: ARM requires a 0 wait state when idle or busy. This signal is active when ahb_htrans[1:0] is equal to 2'b00 (idle) or 2'b01 (busy) and ahb_hready is set.

MDPI to AHB hresp0 and hready Signals

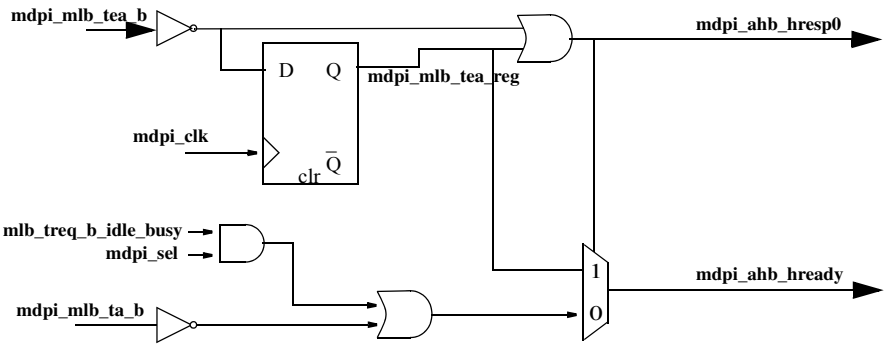


Figure 65-2. hresp0 and hready Block Diagram

65.3 MDPI Multiplexor Block (MMB)

The MDPI Multiplexor block (MMB), (refer to figure 3) is the main interface-switch for directing/controlling the MCU data/control signals to the X/Y RAMs. Most of the control for these muxes come from the BISM module and are described below:

1. **dpr_x_ce**: This signal originates from the BISM and is used for latching read data back from the DPR in 32-bit mode, data muxing back to MCU from DPR on second half of an access during 32-bit mode and on 16-bit accesses and read/write control signal muxing to the appropriate X-RAM or Y-RAM.
2. **dpr_x_ce_int:** This signal originates from the BISM and is used for mux control of the read/write control strobes to the X/Y RAMs.
3. **dpr_x_ce_ext:** This signal originates from the BISM and is used to provide data mux control from the DPR to MCU during the second half of accesses during 32-bit mode.
4. **mdpi_sel_valid:** This signals originates from the BISM and its primary use in the MMB is to provide a valid read or write strobe to the X/Y RAMs during the first cycle of an access and to provide a "valid" control to latch the read/write (ahb_rw) control from the mcu.
5. **bism_16_access:** This signal originates from the BISM and is used to control the data mux during 16-bit writes to the X/Y RAMs.
6. **bism_32_extend:** This signal originates from the BISM and is used for data mux control during 32-bit reads/writes to the X/Y RAMs.
7. **dpr_addr:** The massaged address bus for the appropriate DPR being accessed. The address is not muxed and is driven to both DPRs (XMEM and YMEM).
8. **word_bndry:** This control signal comes from the BISM and is used to direct data from the MCU to the DPR on writes through the muxes. It's primary use is on 16-bit writes to word boundary locations to guide the data from the upper halfword of the ahb data bus (ahb_data[31:16]) to the dpr data bus.
9. **access_err:** This signal originates from the BISM and is used to prevent the read/write strobes from-serting during these error conditions to prevent accesses to the DPR.

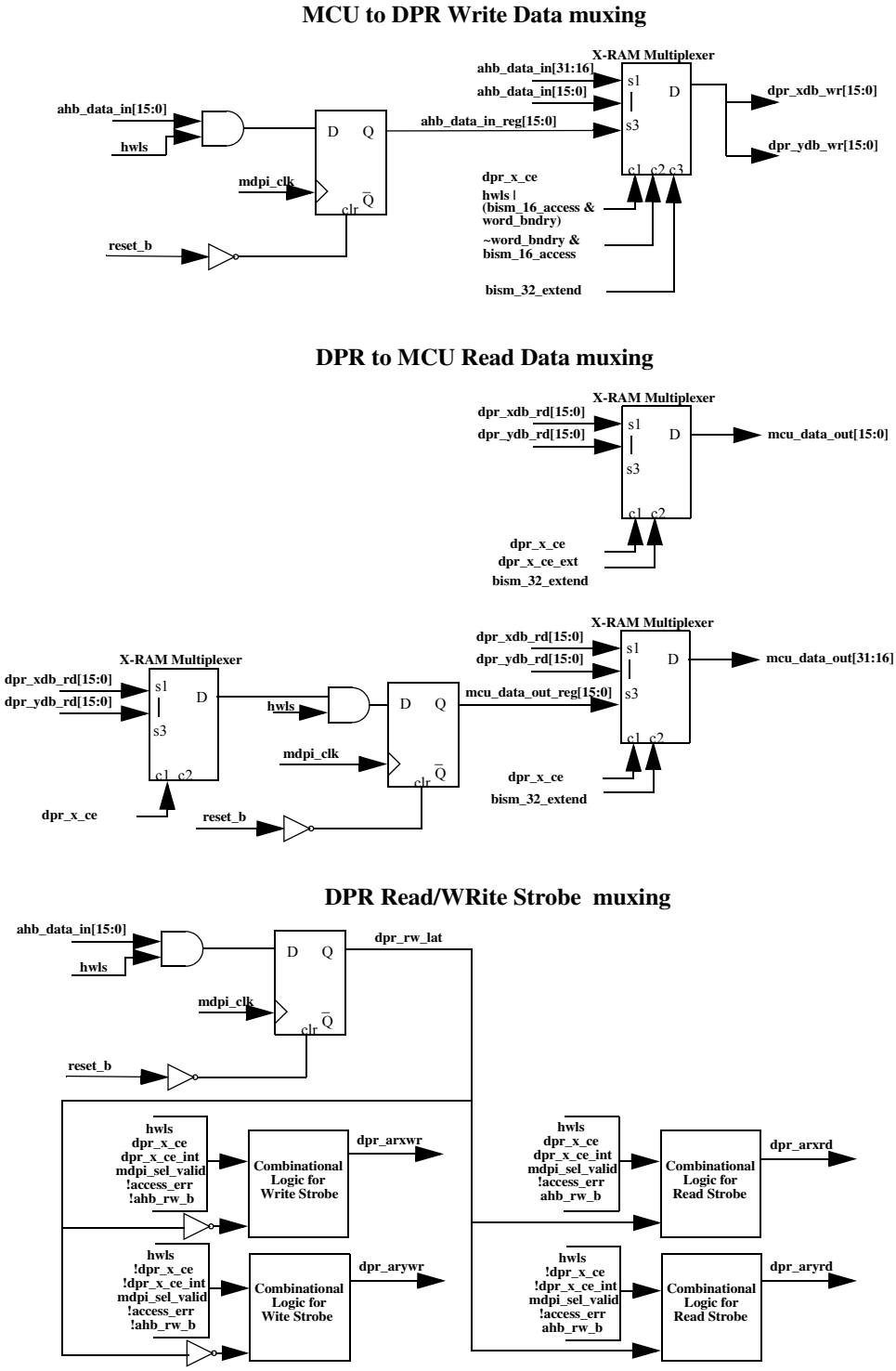
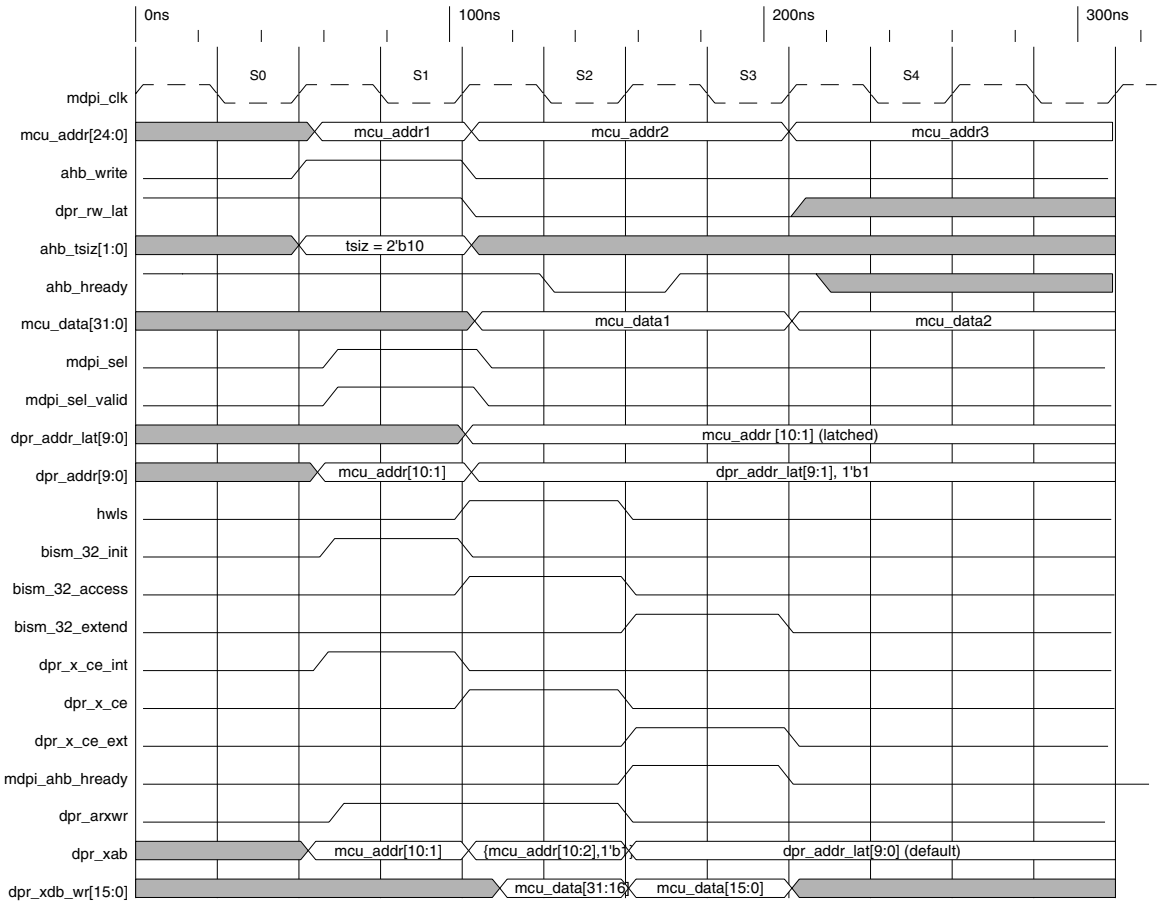
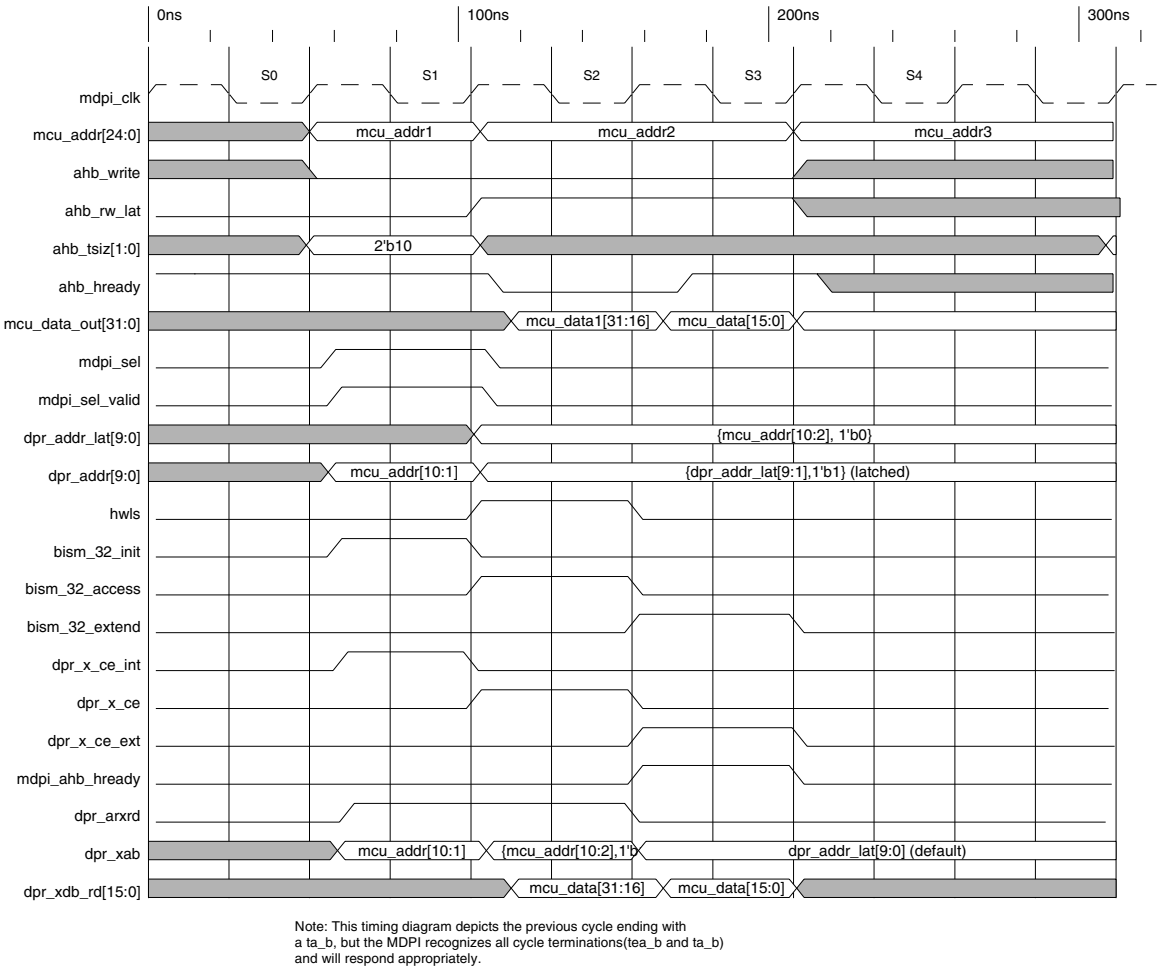


Figure 65-3. MDPI Multiplexer Block Diagram



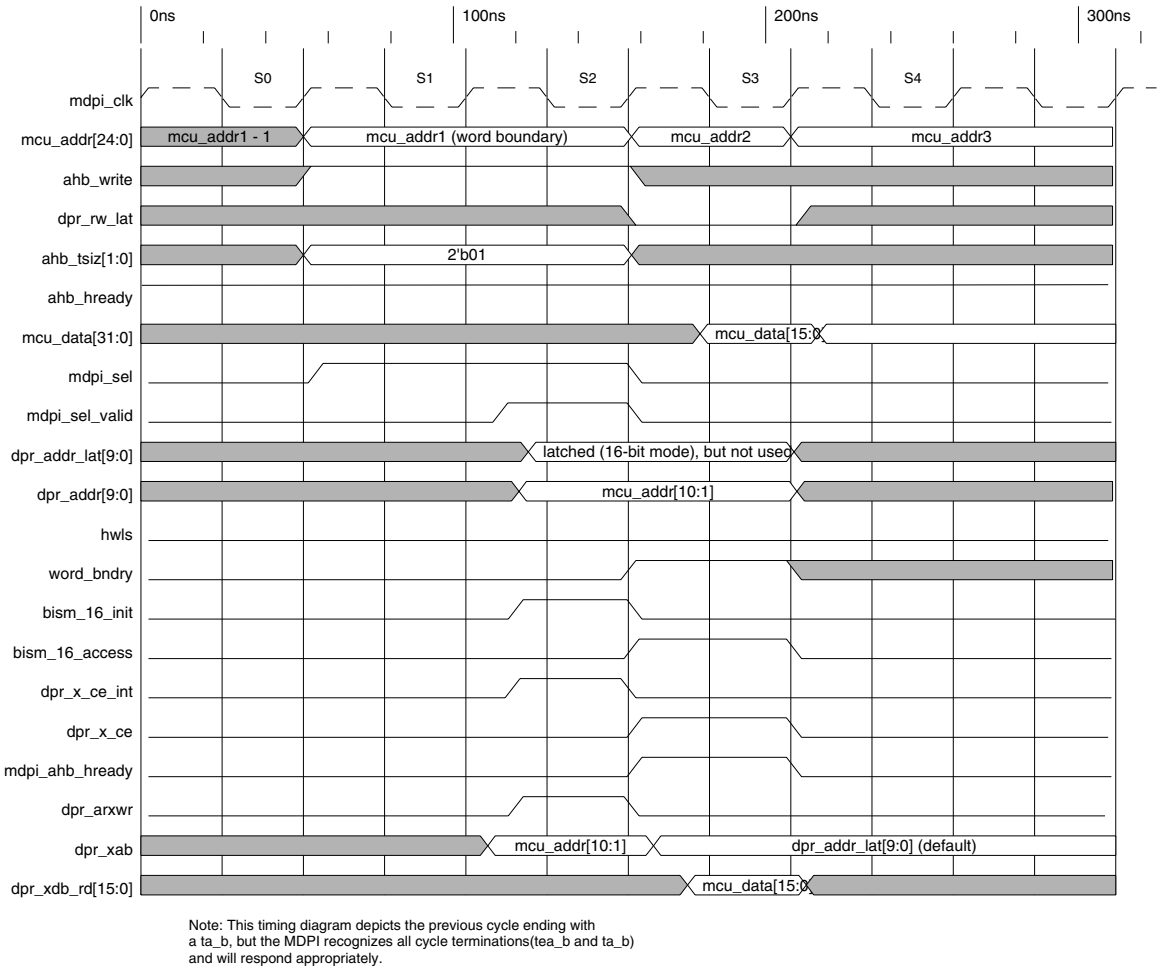
Note: This timing diagram depicts the previous cycle ending with
a ta_b, but the MDPI recognizes all cycle terminations(tea_b and ta_b)
and will respond appropriately.

Figure 65-4. MDPI_BISM Timing Diagram (32-bit write)



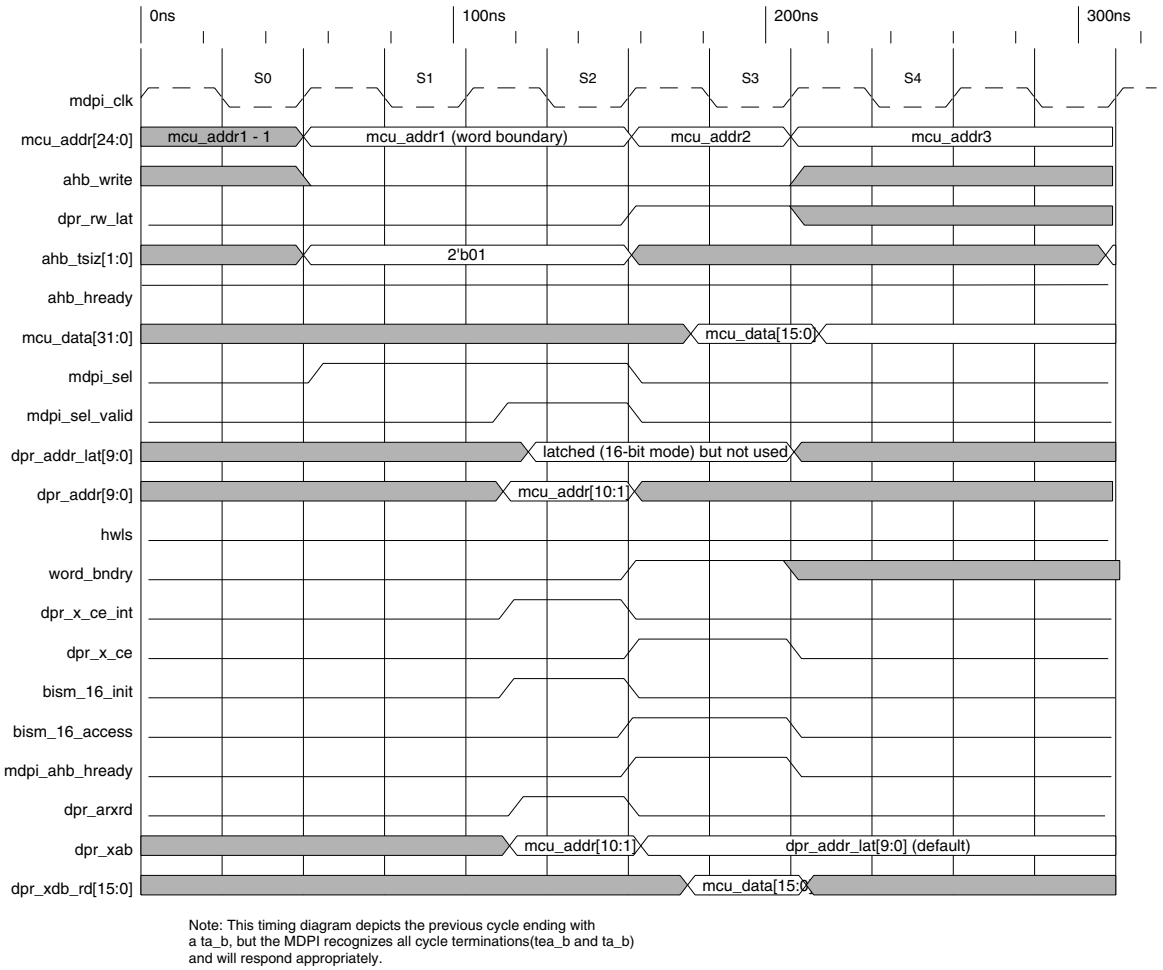
Note: This timing diagram depicts the previous cycle ending with
a ta_b, but the MDPI recognizes all cycle terminations(tea_b and ta_b)
and will respond appropriately.

Figure 65-5. MDPI_BISM Timing Diagram (32-bit read w/0 wait states)



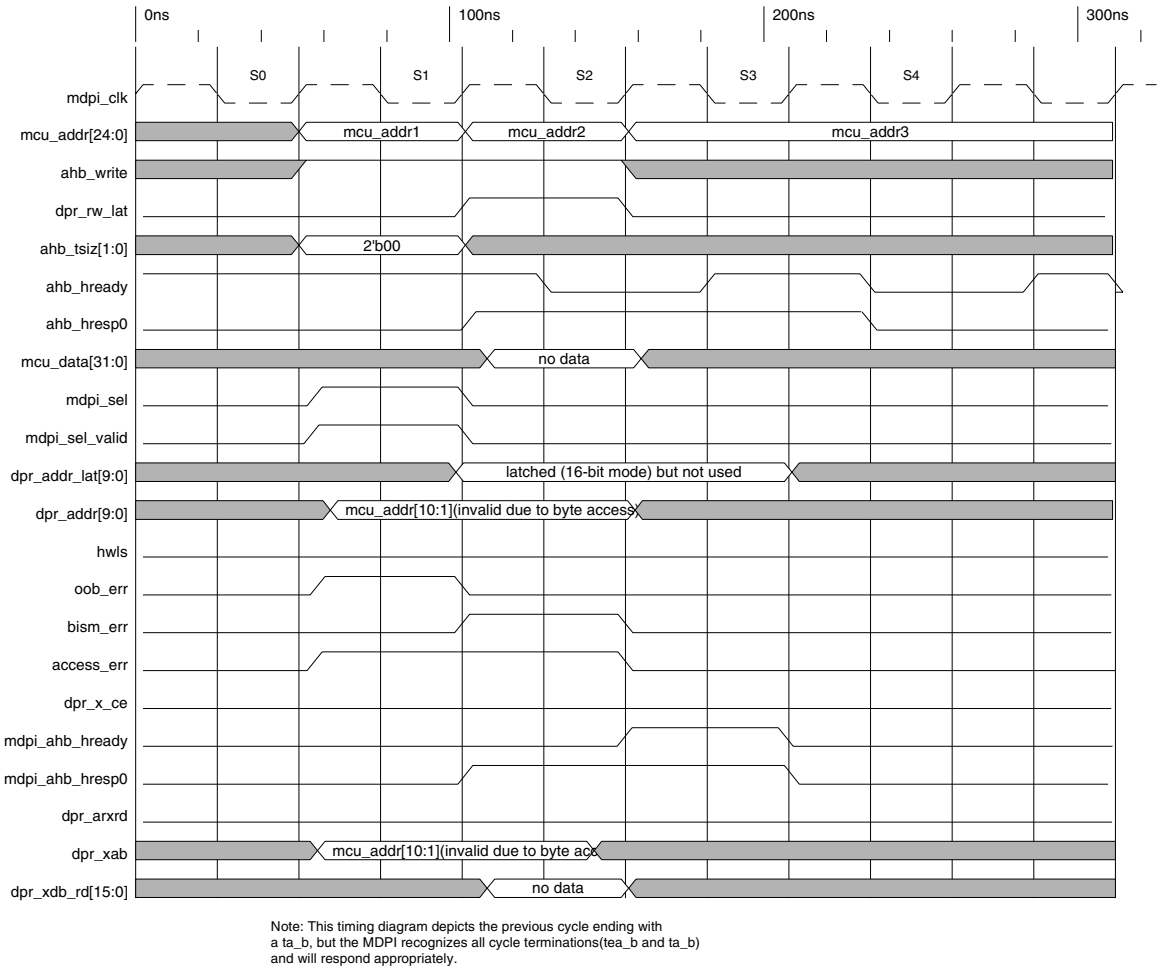
Note: This timing diagram depicts the previous cycle ending with a ta_b, but the MDPI recognizes all cycle terminations(tea_b and ta_b) and will respond appropriately.

Figure 65-6. MDPI_BISM Timing Diagram (16-bit write w/0 wait states)



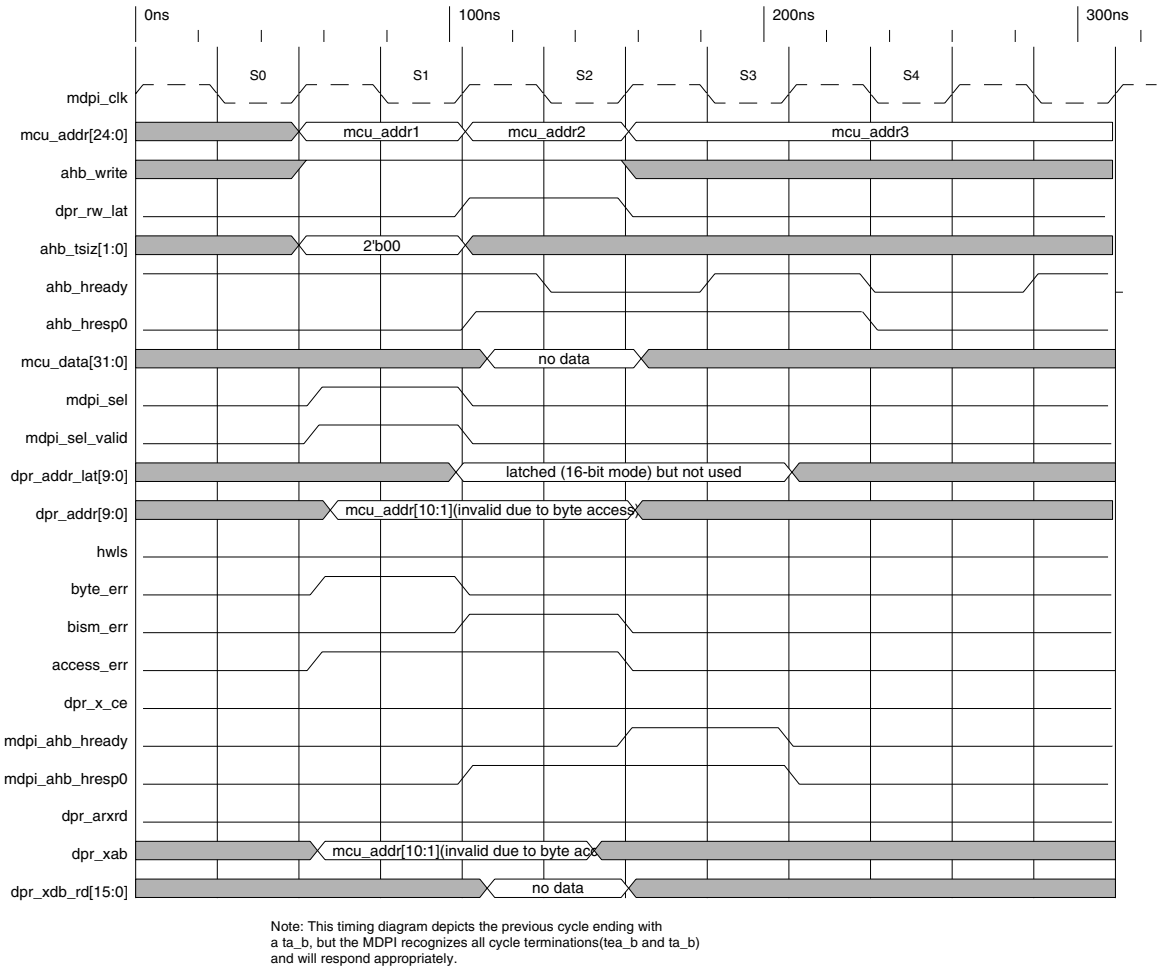
Note: This timing diagram depicts the previous cycle ending with a ta_b, but the MDPI recognizes all cycle terminations(tea_b and ta_b) and will respond appropriately.

Figure 65-7. MDPI_BISM Timing Diagram (16-bit read w/0 wait states)



Note: This timing diagram depicts the previous cycle ending with
a ta_b, but the MDPI recognizes all cycle terminations(tea_b and ta_b)
and will respond appropriately.

Figure 65-8. MDPI_BISM Timing Diagram (out of bounds tea_b generation)



Note: This timing diagram depicts the previous cycle ending with
a ta_b, but the MDPI recognizes all cycle terminations(tea_b and ta_b)
and will respond appropriately.

Figure 65-9. MDPI_BISM Timing Diagram (byte access error - hresp0 generation)

Chapter 66
One - Wire Interface (OWIRE)

Revision History Table

Rev	Date	Author	Changes
0.0	03/15/02		Initial Release. From Patriot-Indy Baseband Spec.
0.1	03/27/02	Ponch Gonzales	- Changed from PIG Interface to AIPI Interface - Updated Pin list - Changed Base Address of Registers to start at $2485_6000
0.2	04/17/02	Ponch Gonzales	- Updated Section 66.3, “Port Definitions.”
0.3	06/05/02	Ponch Gonzales	- Changed the owire_ips_rdata from a 16-bits bus to an 8-bit bus in Section 66.3, “Port Definitions.”
0.4	05/07/03		Updated for LTE specification release.

66.1 Introduction

The initial assumption is that the Neptune LTE 1-wire controller supports the same features as the Patriot-Indy version ([5]).

The 1-wire interface provides the communication line to a 1Kbit Add-Only Memory (DS2502). The interface will be able to send or receive one bit at a time. The required protocol for accessing the DS2502 is defined by Dallas Semiconductor. The DS2502 will hold battery characteristics information. The 1-wire module is a peripheral device to the ARM7 Core and communicates with it via the IP interface. This specification describes the 1-wire function, timing diagrams, port definitions as well as notes on testing the 1-wire module.

This document was adapted from the Patriot-Indy OneWire specification.

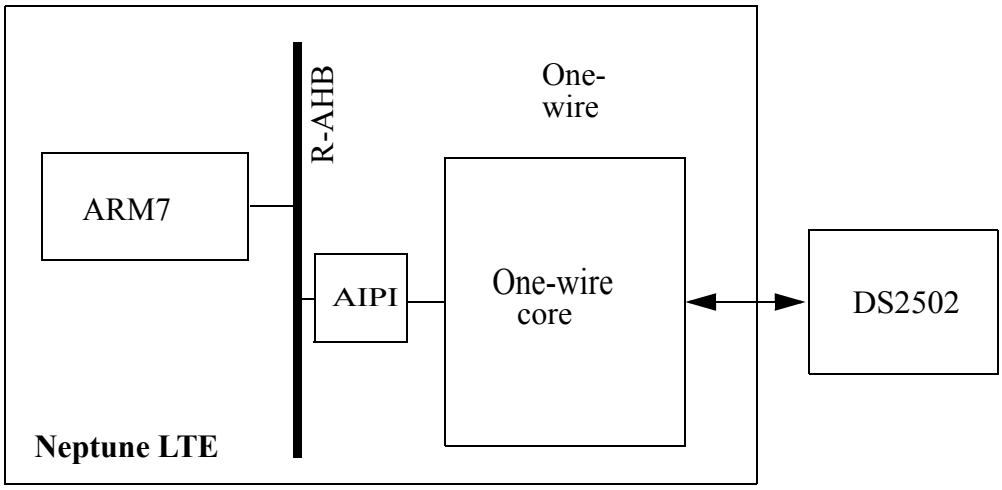


Figure 66-1. One-Wire Connection

66.2 Peripheral Architecture

A block-level description of the 1-wire module is contained in Figure 66-2.

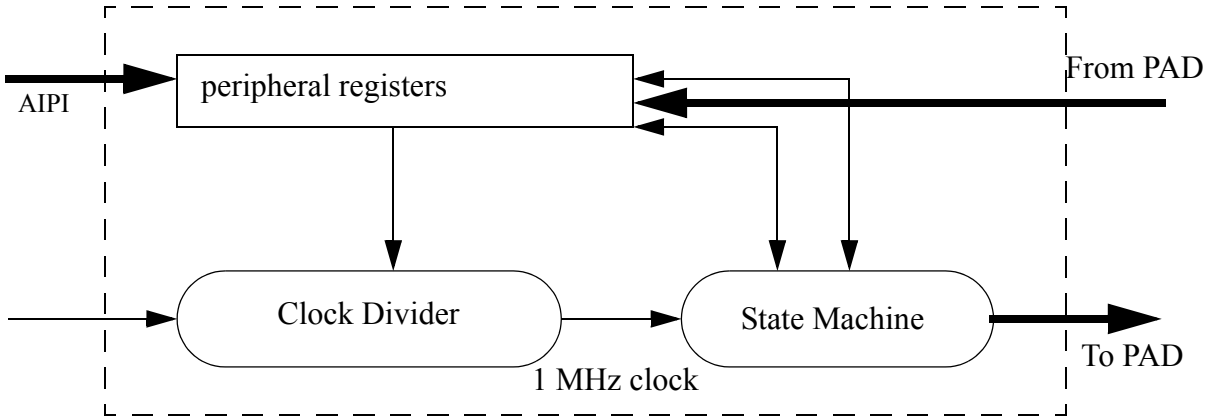


Figure 66-2.

The clock divider is used to generate a 1 MHz clock used as a time reference by the state machine. Transitions between the states of the state machine as well as actions triggered at precise time deadlines are expressed using this 1 MHz clock. The state machine performs all required actions to dialog with the external device.

66.3 Port Definitions

The inputs and outputs for the one-wire are listed in the four tables below. They are organized to show the major interfaces for the 1-wire. Table 66-1 lists the inputs and outputs relevant for the AIPI bus protocol. The DS2502 input and output lines listed in Table 66-2 are the lines that interface with the DS2502. In Table 66-3, the clocks are described. In Table 66-4, the test signals are described.

Table 66-1. 1-Wire Port Definitions. IP bus interface

Signal	I/O	Comments
ipg_async_hard_reset_b	input	active low asynchronous reset
ips_module_en	input	module select signal
ips_rwb	input	module select signal read/write control signal
ips_addr[11:1]	input	module address bus (11 bits)
ips_byte_7_0	input	Byte enable for data[7:0]
ips_wdata[7:0]	input	Write data bus on IP bus
owire_ips_rdata[7:0]	output	8-bit read data bus on IP bus
owire_ips_xfr_wait	output	Transfer will not complete in current cycle
owire_ips_xfr_err	output	Data Error during transfer

Table 66-2. 1-Wire Port Definitions. DS2502

Signal	I/O	Comments
battery_line_in	input	One-wire bus. Used in hdl model to set one-wire internal regs
battery_line_out	output	Connected to GND for open drain
output_enable	output	Enable for output driver One-wire bus. In hdl model, represents DS2502 input

Note: The outputs above have been set for a standard I/O pad.

Note: The DS2502 specifies an external pull-up should be used (value TBD).

Table 66-3. 1-Wire Port Definitions. System

Signal	I/O	Comments
ipg_clk	input	IP clock.
ckih_clk	input	High frequency clock to module.

Table 66-4. 1-Wire Port Definitions. Test

Signal	I/O	Comments
ipt_scan_mode	input	Global scan mode
ipt_clk_se	input	test scan enable signal

66.4 Functional Description

The 1-wire interfaces with the 1Kbit Add-only Memory (DS2502) through a simple 1 bit bus. The DS2502 1Kbit Add-only Memory, manufactured by Dallas Semiconductor, uses the 1-wire line to program and read a 1024 bit EPROM. The DS2502 also has a 64-bit lasered ROM and Status Bytes. The DS2502 requires a special protocol to access the EPROM. The protocol involves first issuing one of four ROM function commands before the EPROM is accessible: read ROM, match ROM, search ROM and skip ROM. Through the 1-wire bus, the ARM7 Core interfaces with the DS2502 and allows the required commands to be issued to control the EPROM. The details of the DS2502 required procedures of operation can be found in the referenced document [2]. The ARM7 Core (through the one-wire interface) is the bus master and the DS2502 device(s) are the slave(s). The 1-wire peripheral does not trigger interrupts; hence a polling of the 1-Wire by the MCU is necessary to manage a correct behavior of the block (refer to the description of Section 66.5).

66.4.1 Low Power Modes

The 1-wire goes into low power mode whenever it is not in use. It goes into low power mode by gating off the clock. The 1-wire is considered to not be in use whenever the 1-wire is not communicating with the DS2502. Another way of putting it is the 1-wire goes into low power mode when the RPP, WR0, and WR1 bits of the Control register are all zero.

66.4.2 Reset Sequence with Reset Pulse Presence Pulse

To begin any communications with the DS2502, it is required that an initialization procedure be issued. A reset pulse must be generated and then a presence pulse must be detected. The minimum reset pulse length is 480 us. The bus master (one-wire) will generate this pulse, then after the DS2502 detects a rising edge on the one-wire bus, it will wait 15-60 us before it will transmit back a presence pulse. The presence pulse will exist for 60-240 us.

The timing diagram for this sequence is shown in Figure 66-3.

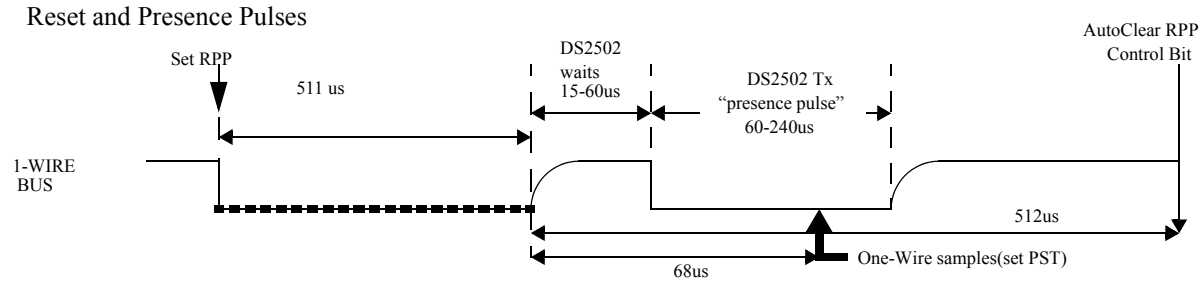


Figure 66-3. 1-Wire Initialization

The reset pulse begins the initialization sequence and it is initiated when the RPP control register bit is set. When the presence bit is detected, this bit will be cleared. The presence pulse is used by the bus master to determine if at least one DS2502 is connected. Software will determine if more than one DS2502 exists. The one-wire will sample for the DS2502 presence bit. The presence bit is latched in the one-wire control register PST. When the PST bit is set to a one, it means that a DS2502 is present; if the bit is set to a zero, then no device was found.

66.4.3 Write 0

The Write 0 function simply writes a zero bit to the DS2502. The sequence takes 117 us. The 1-wire bus is held low for 100us.

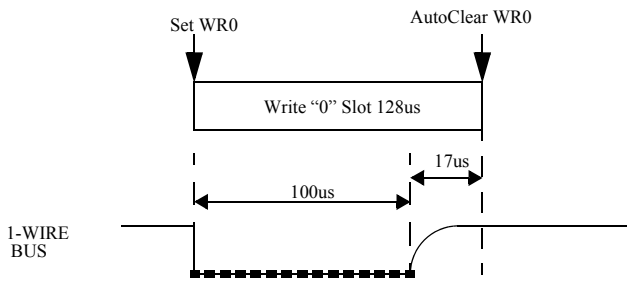


Figure 66-4. Write 0 Timing

The Write 0 pulse sequence is initiated when the WR0 control bit register is set. When the write is complete, the WR0 register will be auto cleared.

66.4.4 Write 1/Read Data

The Write 1 and Read timing is identical. The time slot is first driven low. According to the DS2502 documentation, the DS2502 has a delay circuit which is used to synchronize the DS2502 with the bus master (1-wire). This delay circuit is triggered off of the falling edge of the data line and is used to decide when the DS2502 should sample the line. In the case of a write 1 or read 1, after a delay, a 1 will be transmitted / received. When a read 0 slot is issued, the delay circuit will hold the data line low to override the 1 generated by the bus master (1-wire).

For the Write 1 or Read, the control register WR1/RD is set and auto-cleared when the sequence has been completed. For a Read, the control register RDST is set to the value of the read.

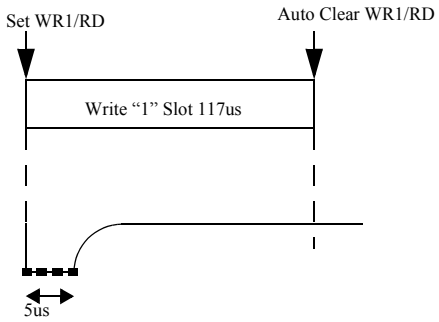


Figure 66-5. Write 1 Timing

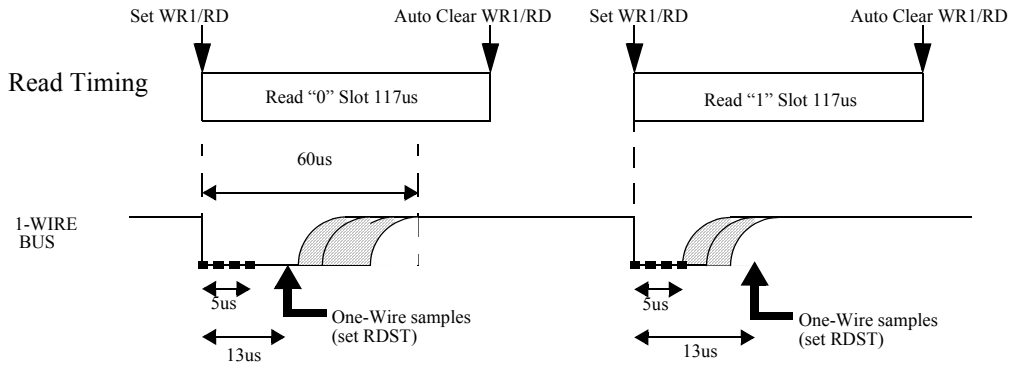


Figure 66-6. Read Timing

66.4.5 Program Pulse

The Program Pulse sequence is described in the DS2502 documentation as one of the functions of the one-wire signaling. The 12 volt programming pulse function is not being used in the 1-wire.

66.5 Registers

66.5.1 Register Summary

Figure 0-19.

KEY: Always Read One (1) | Always Read Zero (0) | Read/Write bit (bit) | Read only bit (bit) | Write only bit (bit) | Write 1 to clear (bit / *W1C*) | Self clear bit (0 / bit) | N/A

Table 66-5. 1-Wire Register Map Summary: $00211000 - $00211FFF

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Control ($2485_6000)	R	0	0	0	0	0	0	0	0	RPP	PST	WR0	WR1	RDST	0	0	0
	W																
Time Divider ($2485_6002)	R	0	0	0	0	0	0	0	0	dvdr[7:0]							
	W																
Reset ($2485_6004)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	reset
	W																

NOTE:

All Registers are byte writable.

66.5.2 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various 1-wire registers. The following definitions serve as a key for these figures:

- **u**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it is cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

66.5.2.1 Control Register

CONTROL Control Register **$2485_6000**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
									RPP	PST	WR0	WR1	RDST			
TYPE	r	r	r	r	r	r	r	r	rwm	ro	rwm	rwm	ro	0	0	0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 66-6. CONTROL Description

Name	Description	Settings
Bits 15-8	Reserved bits	N/A
RPP Bit 7	**RESET PRESENCE PULSE- This bit is self clearing and will be cleared after the presence is determined.**	0 = Do nothing / Reset pulse complete. 1 = Generate Reset Pulse and sample for DS2502 presence pulse. This bit is self clearing and will be cleared after the presence is determined.
PST Bit 6	**PRESENCE STATUS - This bit is valid after the RPP bit is self cleared.**	0 = Device is not present. 1 = Device is present. This bit is valid after the RPP bit is self cleared.
WR0 Bit 5	**WRITE 0 - This bit is self clearing and will be cleared when the write of the bit is complete.**	0 = Do nothing / Write sequence complete. 1 = Write a 0 bit to the interface. This bit is self clearing and will be cleared when the write of the bit is complete.
WR1 Bit 4	**WRITE 1 / READ - This bit is self clearing and will be cleared when the write of the bit is complete. This will also read a bit since the Write 1 and Read timing are identical. The value of the read bit is stored in RDST, and is valid after WR1/RD is self-cleared.**	0 = Do nothing / Write sequence complete. 1 = Write a 1 bit to the interface. This bit is self clearing and will be cleared when the write of the bit is complete. This will also read a bit since the Write 1 and Read timing are identical. The value of the read bit is stored in RDST, and is valid after WR1/RD is self-cleared.
RDST Bits 3	**READ STATUS - This bit is valid after the WR1/RD bit is self cleared.**	0 = A 0 was sampled during a read. 1 = A 1 was sampled during a read. This bit is valid after the WR1/RD bit is self cleared.
Bits 2-0	Reserved bits	N/A

66.5.2.2 Time Divider Register

TIME_DIVIDER Time Divider Register **$2485_6002**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
									dvdr[7:0]							
TYPE:	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 66-7. Time Divider Description

Name	Description	Settings
Bits 15-8	Reserved bits	N/A
dvdr [7:0] Bits 7-0	**Predivider Factor—**The 1-wire also contains a clock divider register used to generate the internal time base within the module. Internal time generation is made up by a clock divider. The purpose of this internal time generation is to make a 1 MHz clock from the main clock.	**dvdr[7:0]** / **Clock Divisor** $00 / 1 (default) $01 / 2 - - - $FF / 256

It is the user's responsibility to program this register so that the binary rate frequency is as close as possible to 1 MHz (1 MHz = clock / (divider + 1)). If the clock frequency is 30 MHz, then the proper value to write into the divider register is 29.

Note: The precision of the generated clock is very important to get a proper behavior of the 1-wire module. This module is based on a state machine which undertakes actions at defined times.

The precision of the generated clock is very important to get a proper behavior of the 1-wire module. This module is based on a state machine which undertakes actions at defined times.

Table 66-8. System timing requirements

times	values(microsec)	min(microsec)	max(microsec)	absolute precision	relative precision
RSTL	511	480	-	31	0.0645
PST	68	60	75	7	0.1
RSTH	512	480	-	32	0.0645
LOW0	100	60	120	20	0.2
LOWR	5	1	15	4	0.8
READ_sample	13		15	2	0.15

The most stringent constraint is 0.0645 as a relative time imprecision.

The time relative precision is directly derived from the frequency of the derivated clock f.

time relative precision = 1/f -1 = divider/clock(MHz) - 1

The table gathers relative time precision for different main clock frequencies.

Table 66-9. Examples of relative time imprecision

main clock frequency (MHz)	13	16.8	19.44
clock divide ratio	13	17	19
generated frequency (MHz)	1	0.9882	1.023
relative time imprecision	0	0.0117	0.023

This shows that the user should take care of the main clock frequency when using the one-wire module. If the main clock is an exact integer multiple of 1 MHz, then the generated frequency will be exactly 1 MHz.

Note: A main clock frequency below 10 MHz might cause a misbehavior of the module.

66.5.2.3 Reset Register

RESET Reset Register **$2485_6004**

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
																rst
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Name	Description	Settings
Bits 15-1	Reserved bits	N/A
rst Bit 0	**Software Reset—**The reset register is used to reset the module through software. This register is not self-clearing, therefore the programmer must write a 1 to reset the registers and then write a 0 to release the reset signal.	0 = 1-wire is out of reset 1 = 1-wire is in reset

66.6 Addendum

The note below illustrates the differences between the Whitecap and the Patriot IC version of the 1-wire module.

The addendum below is taken as is from the Whitecap1.0 specification. It has been updated to integrate Patriot IC notes. As can be noted through reading, the Patriot IC sticks closely to the previous WhiteCap RAM2 1-wire implementation.

The RAM1A version of the 1-wire was design to meet the DS2502 Specifications. Software review of the 1-Wire Specification requests that the OneWire delays on Whitecap be adjusted to be the same as delays in the corresponding BIC4.1 part. A RAM2 release of the OneWire will reflect these delays. The following diagrams and tables show the comparisons of the delays for the RPPD, WR0 and WR1 sequences for the RAM1A, BIC4.1 and DS2502 Specifications.

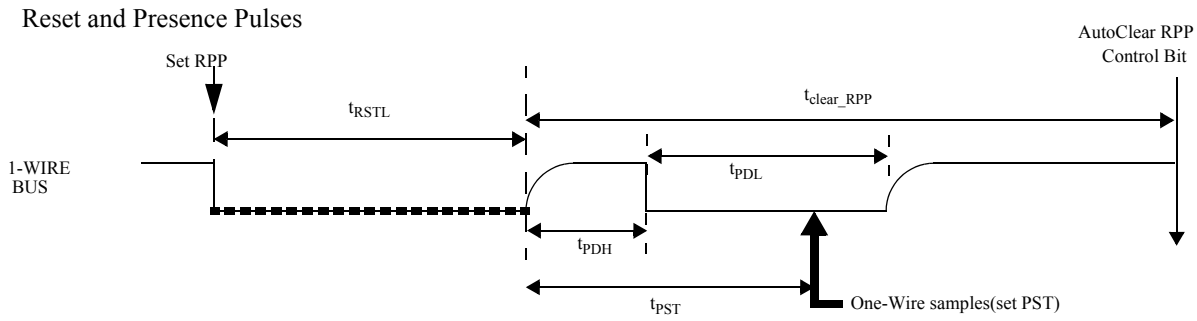


Figure 66-7. Reset and Presence Pulses

Table 66-10. RPP Sequence delay comparisons

Delay	RAM1A Onewire	BIC4.1 Onewire	Patriot	DS2502 Spec
t_{RSTL}	608 us	511 us	511 us	min 480 us
t_{PST}	91 us	68 us	68 us	not specified
t_{PDH}				min 15 us, max 60 us
t_{PDL}				min 60 us, max 240 us
t_{clear_RPP}	149 us	512 us	512 us	not specified

Note: RAM1A is with-in spec for t_{RSTL} (608 is greater than the minimum 480us). The Reset Presence Pulse should functionally work, however, since it is now requested to imitate the BIC4.1, the following change will be made:

CONCLUSION: RAM1A RPP should work. It is with-in specifications.

CHANGES FOR RAM2:

- ***CHANGES (t_{RSTL} to 511us, t_{clear_RPP} to 512us and t_{PST} to 68us) TO IMITATE BIC***

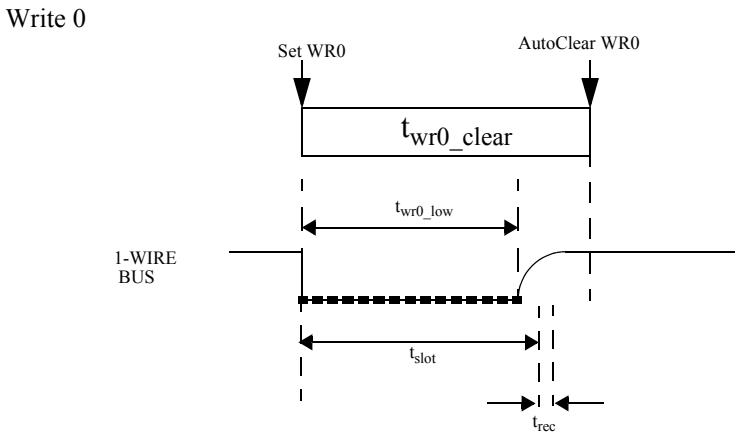


Figure 66-8. Write 0

Table 66-11. WR0 Sequence delay comparisons

Delay	RAM1A Onewire	BIC4.1 Onewire	Patriot	DS2502 Spec
t_{wr0_low}	126 us	100 us	100 us	=< 60 us, < 120 us
t_{wr0_clear}	161 us	117 us	117 us	not specified
t_{slot}				=< 60 us, < 120 us > t_{wr0_low}
t_{rec}				=< 1 us

Note: RAM1A delay t_{wr0_low} is 126 us which is 6 us longer than the DS2502 Spec requirement (< 120 us).

The DS2502 Specification indicates that the active part of the time slot is 60 us and that it can be extended beyond 60 us. Also, it is specified that a low pulse of 120 us may be falsely interpreted as a reset (according to worst-case tolerance ratio), thus, t_{wr0_low} should be < 120 us for Write 0. The delay t_{rec} is a minimum of 1 us (=< 1us), so that the DS2502 can prepare for the next bit.

CONCLUSION: RAM1A WR0 may be interpreted as a reset in worst case conditions because low for 126 us instead of 120 us.

***CHANGE FOR RAM2*:**

- ***t_{wr0_low} will be set to 100 us to avoid false reset***
- ***CHANGE t_{wr0_clear} to 117us TO IMITATE BIC***

Write 1

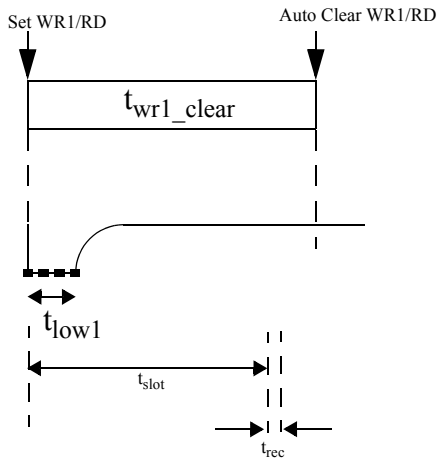


Figure 66-9. Write 1

Table 66-12. WR1 Sequence delay comparisons

Delay	RAM1A Onewire	BIC4.1 Onewire	Patriot	DS2502 Spec
t_{low1}	6 us	5 us	5 us	=< 1 us, < 15 us
t_{wr1_clear}	161 us	117 us	117 us	not specified
t_{slot}				=< 60 us, < 120 us > t_{wr1_low}
t_{rec}				=< 1 us

Note: RAM1A t_{low1} is 1 us longer than the BIC4.1 BUT is with-in the DS2502 specified range of =< 1us and < 15 us.

CONCLUSION: RAM1A WR1 should work. It is with-in specifications.

CHANGES:

- ***CHANGES (t_{low1} to 5us and t_{wr1_clear} to 117 us) TO IMITATE BIC***

66.7 Reference Documents

1. 1995 Automatic Identification Data Book, Dallas Semiconductor
2. DS2502 data sheet, Dallas Semiconductor
3. AIPI bus specification, Motorola
4. Rainbow specification, Motorola
5. Patriot-Indy OneWire specification, Motorola

Chapter 67
Test Control Module (TCM)

Revision History Table

	Date	Author	Changes
0.0	02/15/02		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	03/27/02	Ponch Gonzales	- Removed TCR1 and TCR2 Registers - Combinational Logic to get into the Different Test modes - Updated Pin list
0.2	04/18/02	Ponch Gonzales	- Removed IDDQ register bit from MTC Register, IDDQ will be in the Scan Test Control block - Created the TCR register with only the test_ts register bit, the same place is was for Neptune LT and added ipt_scan_mode signal coming from the Scan Test Control block to code the output test_ts signal correctly for the pads. - Added BIST requirements
0.3	04/22/02	Ponch Gonzales	- Removed DSP Incoming Bist registers - Updated Figure 67-1
0.4	05/2/02	Ponch Gonzales	- Removed tcm_sharedmem_sel signal: removed from Figures, Port list and the MTC Register - Removed DSP Incoming Bist signals from Figures and Port list - Removed “Debug for MCU SharedXmem” and “Debug for MCU SharedYmem” from combinational entry in Table 67-2. Added “DSP Bist Bypass mode” and “MCU Production Bist Mode -all memories” - Added tcm_dsp_bist_bypass_mode signal for Figures and port list - Added tcm_mbist_prod_mode and tcm_mbist_bitmap_mode to TCM - Changed the tcm_gpio_out bus from 42 bits to 17 bits

Revision History Table

	Date	Author	Changes
0.5	06/17/02	Ponch Gonzales	- Added STC Scan dataout[15:0] bus as input. Goes through bus mux to GPIO in Scan modes. Updated Port list. - Updated Figure 67-1and Figure 67-2 - Updated Section 67.3.2, “Bus Mux (BMM),” to include the muxing of the scan dataout bus -Updated Figure 67-1, Figure 67-2, Test Control Register section and Port list per DDTS DSPhl14349
0.6	07/12/02	Ponch Gonzales	-Updated description of output bus tcm_gpio_out in the Port list to include the STC Scan data. - Updated Table 67-7, “DSP Test Modes”.
0.7	09/10/02	Ponch Gonzales	-Updated description of output signal tcm_test_ts in Section 67.3.3, “Test Control Register (TCR),” on page 67-6. Per DDTS DSPhl15071 -Updated description of output bus tcm_boost_test[1:0] in Section 67.3.4, “Miscellaneous Test Control Register (MTCR),” on page 67-7. -Updated Section 67.3.5, “Test Modes,” on page 67-7. Per DDTS DSPhl15204
0.8	09/20/02	Ponch Gonzales	-Updated Section 67.3.5, “Test Modes,” on page 67-7. -Put an Overline on $\overline{\text{MCU\_DE}}$ and $\overline{\text{DSP\_DE}}$ signals because they’re both active low.
0.9	05/07/03		Updated for LTE specification release.

67.1 Overview

The Test Control Module (TCM) for Neptune is a block dedicated for controlling all the test modes that we can exercise on the chip. The TCM contains all the logic control for selecting all different test modes and controlling the dedicated external input/output pins for test purposes. This block is designed with IP Bus 2.0 protocol. It supports the following test modes:

1. Alternate Master Test (AMT) mode as an alternate to the ARM7 high performance bus (R-AHB)

   In the AMT mode, TCM is used for: a) providing the control and data bits for Analog test and debug b) provide address, data and control signals for bit-mapping and debugging internal RAM and ROM c) loading the control registers with proper values for MCU BIST testing d) exercising read and write accesses to any MCU peripheral or memory.

2. DSP Master mode, DSP Address Trace

   In the DSP Master mode, the configuration registers connect the dataout ports to GPIO, so that the DSP address or data can be brought out for observation

3. Input BIST registers for latching the incoming BIST data, fail and done signals.
4. Special register bits are available in the TCM:

   test_ts to tristate all output pads for leakage measurement.

   boost_test to permit the pad drivers to drive the added load of the tester load board.

5. Registering of Scan mode signals in Test Control Register to provide observability

67.1.1 Features

- Test Mode Definitions
- AMT Mode
- Analog Signal observability
- Non Scan Test
- Analog module output selection and configuration
- Memory BIST for ARM7 Memories
- Registered input BIST signals from the ARM7 BIST engine
- Registered input Scan Mode signals from STC module

67.2 Block Diagram

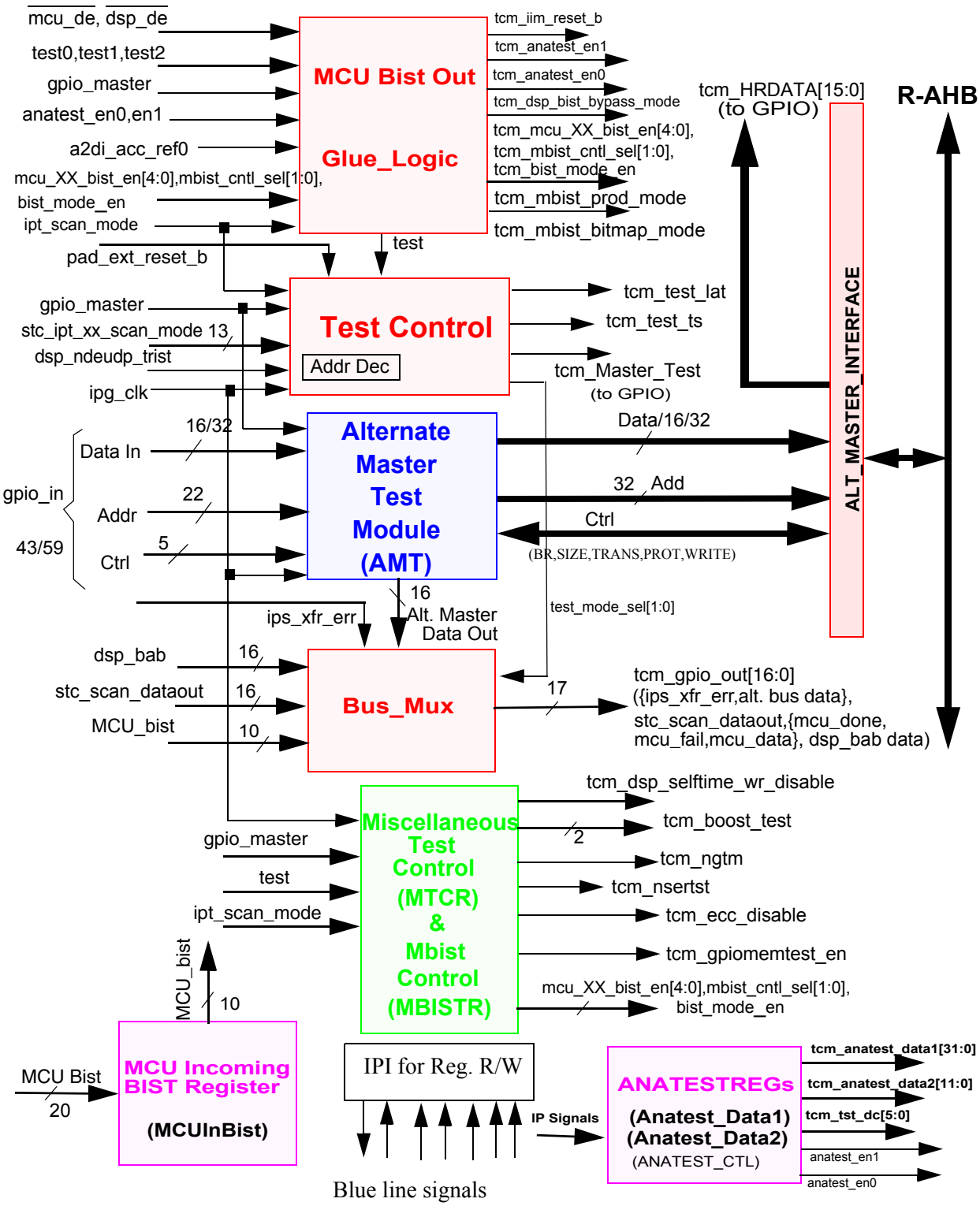


Figure 67-1. Test Control Module (TCM)

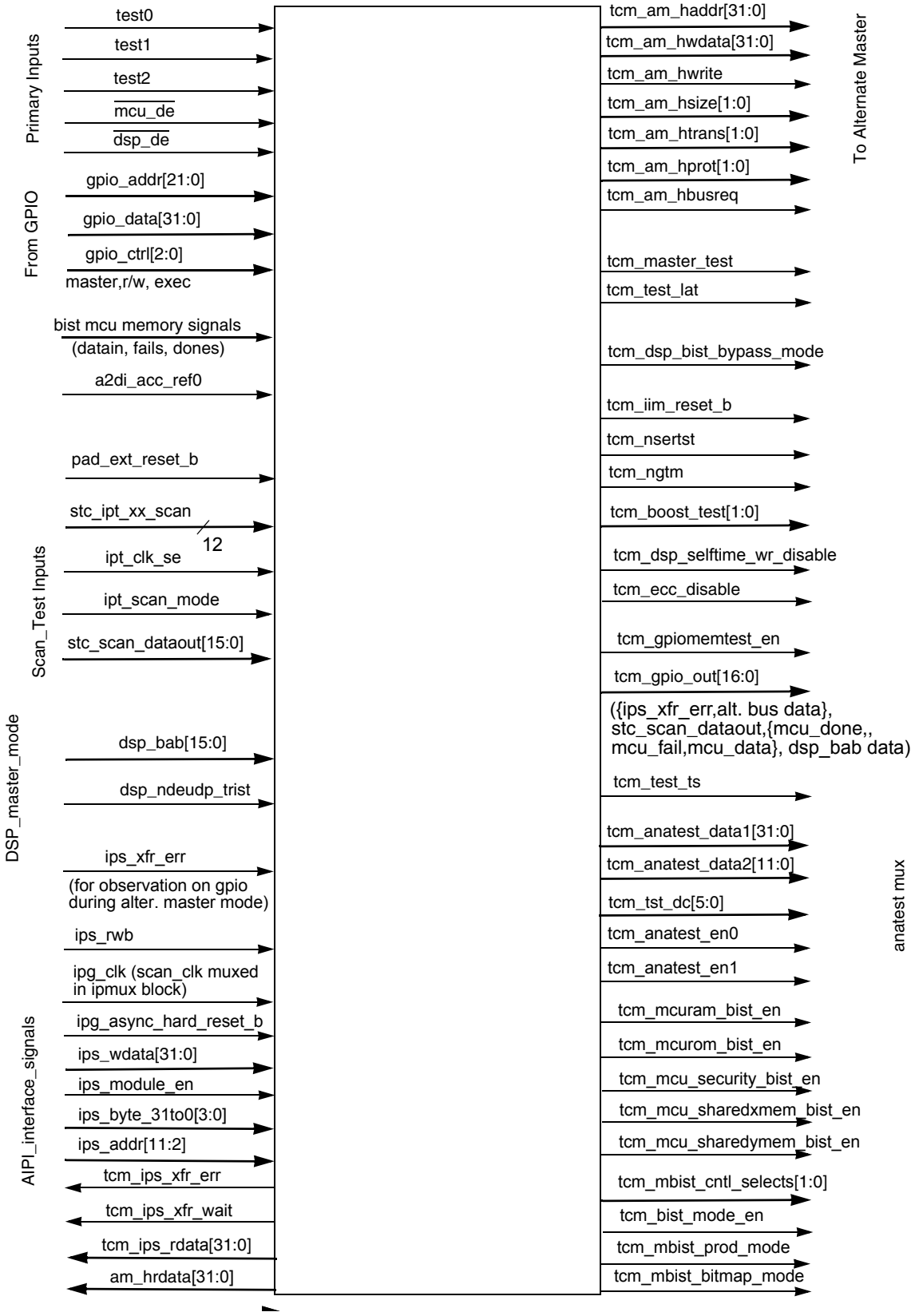


Figure 67-2. TCM Simplified Block Diagram

67.3 Description

67.3.1 Alternate Master Test (AMT)

Alternate Master Test submodule is a block that has the capability of being Master of the R-AHB. In Test mode, a bus request is sent to the ARM7 platform, initiated by a Master signal coming in through a GPIO pin. When the bus is granted by ARM7 platform, Alternate Master Test block takes the R-AHB and can exercise read and write accesses to any MCU peripheral or memory. Alternate Master Test block uses the MCU_CLK signal coming from the Clock Control Module (CCM). For low power reasons, the input clock to the Alternate Master Test block is gated off inside the block itself when not in Test mode.

67.3.2 Bus Mux (BMM)

Bus Mux submodule is used to multiplex test buses to and from the GPIO module. We do that to minimize the routing of the GPIO module. Alternate Master Test dataout bus is routed when exercising AMT mode. The ips_xfr_err input is also routed through Bus Mux during AMT mode. Scan data out {27:12] is routed when ipt_scan_mode is asserted and NOT in AMT mode. When in MCU Bist mode, the mcu data, done and fail signals are driven out of the MCU Bist controller to the TCM, where it is registered and then sent to the GPIO pins. When in “Test” mode and none of the previous tests are selected, we are in DSP stand alone test mode. So, the DSP Program address bus (bab) is routed out to the GPIO.

67.3.2.1 Bus Mux

Bus Mux submodule receives control signals from the GPIO and from primary inputs. These signals select the right test path to observe according to the test mode chosen previously. All the test buses are filtered into that block before going to the GPIO module. The Test signal in the following table is described in Section 67.3.5

DSP Stand alone test mode is the default mode selected during normal operation.

Table 67-1. Bus Mux States

DSP Master Mode and Address Trace	Test & $\overline{\text{MASTER}}$ & $\overline{\text{ipt\_scan\_mode}}$
AMT	Test & MASTER
Scan Mode	Test & $\overline{\text{MASTER}}$ & ipt_scan_mode
MCU Bist Mode	Test & $\overline{\text{MASTER}}$ & $\overline{\text{ipt\_scan\_mode}}$ & tcm_mbist_mode

67.3.3 Test Control Register (TCR)

Test Control submodule (TCR) contains the test_ts register bit and the different scan mode signals. The scan mode signals are read-only. The test_ts is programmable via the IP bus only in the Alternate Bus Master mode. The tcm_test_ts output signal also looks at the DSP tristate signal, the gpio_master input, and the scan mode input. The tcm_test_ts output signal is gated by the master and ipt_scan_mode inputs..

tcm_test_ts	(test_ts | dsp_trist) & ~(master | ipt_scan_mode)

The tcm_test_ts signal tristates output pads to measure leakage.

The TCM registers thirteen scan mode signals coming from the Scan Test Control module in the TCR. This is done for observability purposes.

67.3.4 Miscellaneous Test Control Register (MTCR)

Miscellaneous Test Control submodule (MTCR) consists of one 32-bit register. It is programmable through the IP bus and contains some glue logic.

Some special register bits are part of this module. One is boost_test to permit the pad drivers to drive the added load of the tester load board. The tcm_boost_test output is gated by ipt_scan_mode. If the chip is in scan mode, ipt_scan_mode == 1'b1, then the tcm _boost_test output bus will be deasserted, tcm_boost_test[1:0] == 2'b00.

Two other signals (ngtm and nsertst) allow DSP Master and DSP Address Trace modes.

The tcm_dsp_seltime_wr_disable bit inhibits selftime writes in the dsp.

The tcm_ecc_disable bit inhibits mcurom accesses during error correction.

The gpio_memtest_en bit configures the 16 dataouts from GPIO to be datain. Used in Alternate Master Mode.

67.3.5 Test Modes

In Neptune LTE, TCM has five primary inputs that are used for test modes. The five primary inputs are: $\overline{\text{MCU\_DE}}$, $\overline{\text{DSP\_DE}}$, Test2, Test1 and Test0. $\overline{\text{MCU\_DE}}$ is the debug pin to MCU, and $\overline{\text{DSP\_DE}}$ is the debug pin to the DSP interface.These two pins can be used as test pins when *Test* is asserted high. This *Test* signal will be equal to the following equation:

Test= (Test0 | Test1) & ~(Test2 & ~Test1 & ~Test0)

This *Test* signal will go to the SJC and GPIO module. The GPIO will use the *Test* signal as the Test mode signal now, from this signal it will be ready to configure for any Test mode. The SJC module will use this signal to insure that the $\overline{\text{DSP\_DE}}$ and $\overline{\text{MCU\_DE}}$ pads will be configured as input-only pads in the proposed architecture. The $\overline{\text{MCU\_DE}}$ and $\overline{\text{DSP\_DE}}$ pads are active low and the User has the ability to assert (low) these two pads before asserting the *Test* signal. The User is able to do this because the cores will not go into Debug states immediately after the assertion of the Debug pins. But the User must make sure and assert the *Test* signal (any combination wanted depending on Test mode desired) before the cores do go into Debug states.

For the ARM to get into Debug state: Must assert $\overline{\text{MCU\_DE}}$ for at least four mcu clock cycles.

For the DSP to get into Debug state: Must assert $\overline{\text{DSP\_DE}}$ for at least six ONYX clock cycles.

So, the User can set the Debug pads as desired, wait a cycle, and then set the Test pads as desired (depending on the Test mode the User is trying to achieve).

The User can also wait after asserting *Test0*, *Test1*, or *Test2 (Test* signal*)* before asserting $\overline{\text{DSP\_DE}}$ or $\overline{\text{MCU\_DE}}$ when going into Test modes. But the User must be aware that they may accidently go into an unwanted Test mode before going into the desired Test mode. This will happen if the Debug pad signals get to the TCM later than the Test signals. For example, say the User wanted to enter the default Test mode state, which is {*Test2,Test1,Test0,*$\overline{\text{MCU\_DE}}$,$\overline{\text{DSP\_DE}}$} == {0,0,1,0,0}. The default state of these pads is {*Test2,Test1,Test0,*$\overline{\text{MCU\_DE}}$,$\overline{\text{DSP\_DE}}$} == {0,0,0,1,1}, so the User will force the *Test0* pad to one and the Debug pads to zero. Now, if the Debug pad signals are slightly later than the *Test0* signal, then the chip will enter into a scan mode state (either ARM7 Platform Scan Mode, SONYXU Platform Scan Mode, or SOG-DSP Platform Scan Mode in this case) for a short time. Due to scan logic for reset in the CCM, an automatic system reset (reset stretching sequence occurs) is generated when exiting scan mode, eventhough the User gets into the desired state of {*Test2,Test1,Test0,*$\overline{\text{MCU\_DE}}$,$\overline{\text{DSP\_DE}}$} == {0,0,1,0,0}. The User must be aware of this possiblity.

TCM creates and drives, except for the scan modes, the outputs of a combinational block asserting the modes as shown in Table 67-2.

Table 67-2. Test Modes for LTE

Test Modes	Test2	Test1	Test0	$\overline{\text{MCU\_DE}}$	$\overline{\text{DSP\_DE}}$
Functional	0	0	0	x	x
TEST/Alt. Master Mode/DSP Master Mode and Address Trace Mode	0	0	1	0	0
ARM7 Platform Scan Mode	0	0	1	0	1
SONYXU Platform Scan Mode	0	0	1	1	0
SOG-DSP Platform Scan Mode	0	0	1	1	1
Digital Radio Platform Scan Mode	0	1	0	0	0
Scan Divergence Mode (includes IDDQ)	0	1	0	0	1
Embedded Memory Macrotest Mode 0	0	1	0	1	0
Embedded Memory Macrotest Mode 1	0	1	0	1	1
DSP Bist Bypass Mode	0	1	1	0	0
Debug for MCU ROM	0	1	1	0	1
Debug for MCU RAM	0	1	1	1	0
Debug for MCU Security RAM	0	1	1	1	1
Reserved for User_mode	1	0	0	X	X
MCU Production Bist mode - all memories (Test pins same as in Neptune LT for analog mode)	1	0	1	0	0
MCU RAM Production Bist mode (Test pins same as in Neptune LT for analog mode)	1	0	1	0	1
MCU ROM Production Bist mode (Test pins same as in Neptune LT for analog mode)	1	0	1	1	0
MCU Security Production Bist mode (Test pins same as in Neptune LT for analog mode)	1	0	1	1	1
MCU Shared Y-Memory Production Bist mode (Test pins same as in Neptune LT for analog mode)	1	1	0	0	0
MCU Shared X-Memory Production Bist mode	1	1	0	0	1
Bitmapping Mode for MCU RAM	1	1	0	1	0

Table 67-2. Test Modes for LTE

Test Modes	Test2	Test1	Test0	$\overline{\text{MCU\_DE}}$	$\overline{\text{DSP\_DE}}$
Bitmapping Mode for MCU ROM	1	1	0	1	1
Bitmapping Mode for MCU Security	1	1	1	0	0
Bitmapping Mode for MCU Shared Y-Memory	1	1	1	0	1
Bitmapping Mode for MCU Shared X-Memory	1	1	1	1	0
ROM MISR Mode	1	1	1	1	1

NOTE: Scan mode enables are coming from the Scan Test Control module (STC). The TCM no longer controls any scan mode functionality. The whole table is shown for completeness

NOTE: Master Mode can be entered anytime the generated *Test* signal is asserted, not just when *Test0* is asserted.

67.3.6 MCU BIST Operation

In Neptune LTE, the user will be able to enter different MCU BIST modes through the TCM MBIST Register (Address = $2484_400C) or through combinational logic using the dedicated test input pins, refer to Table 67-2. There are two methods of getting into all the MCU BIST modes.

NOTE: GPIO configuration for BIST depends on the tcm_mbist_prod_mode, tcm_mbist_bitmap_mode and the DSP bist enables. During BIST, it is required to hold master pin low. The TCM bus mux will route the MCU Bist fail and done signals if in MCU Bist Production (in order to do DSP Bist Production in parallel). If the MCU Bist Bitmap mode signal is asserted, then the data, fail and done signals go to the GPIO.

67.3.6.1 MCU Bist operation using MBIST Register

In the Alternate Master Mode, the TCM can write the MBIST register and program the select bits as well as enable bits for the five memory arrays.

The select bits are programmed to exercise the following modes in BIST:

Table 67-3. MCU Memory BIST modes

Select1	Select0	Mode
0	0	Production Test
0	1	Debug
1	0	Bitmap
1	1	ROM MISR dataout

During the MCU memory BIST operation, TCM receives 8 bits of data, ‘done’ flags and ‘fail’ flags from the five memory groups. The TCM also gets a ‘done’ and ‘fail’ flag that is the combined result of the RAM, ROM and Security RAM.

The data out mux decodes where if any BIST enable is set and BIST mode is set then that group is muxed to the output. This enables data out in either **Bitmap** or **MISR data out** mode, mode decode 10 or 11 respectively.

In BIST modes **Production Test** and **Debug** (mode decode 00 or 01 respectively) any combination of bist selects can be set.

In BIST modes **Bitmap** or **MISR data out**, only one bist select should be set. Setting more than one bit in these modes is undefined.

The TCM will register the data, fail and done signals, and then send them to GPIO. The register can be read in Alternate Master Mode. This will give the User the ability to stop BIST, assert the master pin (this will put the chip in Alternate Master Mode) and then read the register values. The register address will be at $2484_401C.

67.3.6.2 MCU Bist operation using Test Pins

The user will also be able to operate all MCU BIST modes using the five input pins, refer to Table 67-2.

67.3.6.3 Example of Both Methods

Here is an example of the two methods described above, this example involves getting into Bitmapping for MCU RAM, look below:

1) Asserting Test2, Test1, $\overline{\text{DSP\_DE}}$ and leaving Test0 low and $\overline{\text{MCU\_DE}}$ high. -> Test2&Test1&Test0&$\overline{\text{MCU\_DE}}$&$\overline{\text{DSP\_DE}}$ = {1,1,0,1,0} = Bitmapping for MCU RAM

2) Or the user can assert the following MBIST Register bits: MBIST[3:2] and MBIST[0]. So, the user would write 32'h0000_000D to address $2484_400C = Bitmapping for MCU RAM

The TCM output port signals, tcm_mcuram_bist_en and tcm_mbist_cntl_selects[1], will be asserted by either of the above methods. Method 1 is a simpler and faster way of getting into any MCU BIST mode. This is because, for method 2, the user will have to get into Alternate Master Mode to write into the MBIST register and then get out of Alternate Master Mode before the signals get asserted, where as, method 1 only requires the correct combination of the five dedicated input pins. These signals go to the MCU BIST controller.

The MCU BIST controller decodes the signals and then sends the data, the fail flags and the done flags back through the TCM. The TCM will register the data, fail and done signals, and then send to the GPIO. The registered input signals are in Section 67.6.1, "Register Description of Incoming Bist signals," .

This example will also assert the tcm_mbist_bitmap_mode signal which will configure the GPIO accordingly.

Switching between BIST mode and Alternate Master Mode is achieved by toggling the master pin.

67.3.7 DSP BIST Operation

DSP BIST testing can not be controlled through the TCM. The operation will be handled through JTAG in the SJC module. The TCM will, however, send out a DSP Bist Bypass mode signal. The signal is created by combinational logic using the five dedicated Test pins mentioned in "Test Modes" on page 7.

67.4 Non Scan Tests

The following sections describe the different kinds of Non Scan Test modes.

67.4.1 ARM7 Trace Mode

(Functional mode + GPIO pre-configured by Boot Rom). We load data+Program from MCU mem into DSP X-data, Y-data.

67.4.2 ARM7 Functional

(Trace mode disable, i.e.: GPIO not pre-configured). Same operation.

67.4.3 DSP56600 Test Mode

- Burn in mode. Used for Burn-in only. Test RAMs, exercise ROMs.
- Type 2. To measure typical power dissipation.
- Wait mode. DSP56600 command.
- Stop mode. DSP56600 command.
- Boot from external memory word-wide. Only used when DSP is master (DSP56600 stand alone test). We load through GPIO pins 24 bits data.
- Boot from external memory byte-wide
- Boot from Dual port RAM

The DSP boot code contains the definition of all these test modes depending on SAP and BBP inputs sampled during Reset.

Two register bits (tcm_ngtm and tcm_nsertst) are set in the MTCR to enable the GPIO to control the DSP Master or the DSP Address Trace modes. MTCR is also writable by MCU. This function is a mirror of what is already done in the DSP Once Module. This is a redundancy for a backup solution.

67.5 Register Map

The register map in the TCM is shared between the Test Control register, the Miscellaneous Test Control register, the MBIST register, the ANATEST registers, and the MCU InBist register. For all these registers we have the same Base Address (see MCU Memory Map).

Figure 0-20.

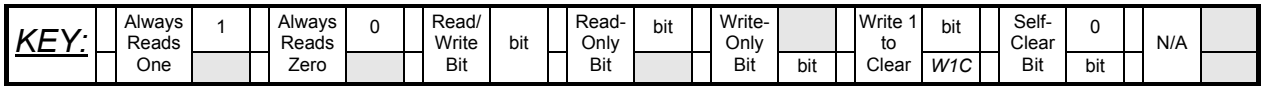


Table 67-4. TCM Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCR ($2484_4000)	R	test_ts															
	W																
	R				stc_ipt_macro test_mqspi	stc_ipt_macro test_16bit	stc_ipt_macro test_8bit	stc_ipt_scan_divergence	stc_ipt_sog_scan	stc_ipt_dig_radio_ext_scan	stc_ipt_dig_radio_int_scan	stc_ipt_onyx_ext_scan	stc_ipt_onyx_int_scan	stc_ipt_arm_ext_scan	stc_ipt_arm_int_scan	ipt_scan_mode	stc_ipt_test
	W																
($2484_4004)	R	Reserved Address Space															
	W																
	R																
	W																
MTCR ($2484_4008)	R		tcm_dsp_selfwr_disable	tcm_ecc_disable			gpio_memtest_en	Boost_test[1:0]				tcm_nsertst	tcm_ngtm				1
	W																
MBISTR ($2484_400C)	R																
	W																
	R									mcu_shared_xmem_en	mcu_shared_ymem_en	mcu_security_en	mcu_rom_en	mcu_ram_en	mcu_bist_mode_sel[1:0]		bist_mode_en
	W																
ANA_CNTL ($2484_4010)	R																
	W																
	R		anatest_en1	anatest_en0								TST_DC[5:0]					
	W																
ANA_DATA1 ($2484_4014)	R	TXSYN[5:0]						PAC[4:0]					CMON[4:0]				
	W																
	R	TUNEC[7:0]								VOC[1:0]		DET_TEST_MODE	RXAFE[4:0]				
	W																
ANA_DATA2 ($2484_4018	R																
	W																
	R W					REG[7:0]								GPADC[3:0]			

Table 67-4. TCM Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCUInBist ($2484_401C)	R													mcu_bist_done	mcu_bist_fail	mcu_shared_ymem_bist_done	mcu_shared_ymem_bist_fail
	W																
	R	mcu_shared_xmem_bist_done	mcu_shared_xmem_bist_fail	mcu_security_bist_done	mcu_security_bist_fail	mcu_ram_bist_done	mcu_ram_bist_fail	mcu_rom_bist_done	mcu_rom_bist_fail	MCU Bist Incoming Data[7:0]							
	W																

67.6 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various TCM registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and is cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0.
  - **1**: Will reset to a logic 1.
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset.

The Test Control Register (TCR) contains the test_ts register bit which can tristate output pads. It also contains scan mode signals from the STC for observability purposes. These thirteen scan registered bits can only be reset by the PAD reset.

This 32-bit read/write register is accessible via the IP bus only in AMT mode.

TCR **Test Control Register** **Addr $2484_4000**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
	test_ts															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
				stc_ipt_macrotest_mqspi	stc_ipt_macrotest_16bit	stc_ipt_macrotest_8bit	stc_ipt_scan_divergence	stc_ipt_sog_scan	stc_ipt_dig_radio_ext_scan	stc_ipt_dig_radio_int_scan	stc_ipt_onyx_ext_scan	stc_ipt_onyx_int_scan	stc_ipt_arm_ext_scan	stc_ipt_arm_int_scan	ipt_scan_mode	stc_ipt_test
TYPE	rw	rw	rw	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 67-5. TCR Description

Name	Description	Settings
test_ts Bit 31	**test_ts -** Determines output pads state.	0: disabled (default) 1: output pads tristated (for leakage measurement)
Unused Bits 30-13	**Unused**	N/A
stc_ipt_macrotest_mqspi Bit 12	**stc_ipt_macrotest_mqspi -** STC small embedded memory macrotest to mqspi	0: disabled (default) 1: scan test was enabled (can only be reset by PAD reset, **RESET_IN**)
stc_ipt_macrotest_16bit Bit 11	**stc_ipt_macrotest_16bit -** STC small embedded memory macrotest to L1T	0: disabled (default) 1: scan test was enabled (can only be reset by PAD reset, **RESET_IN**)
stc_ipt_macrotest_8bit Bit 10	**stc_ipt_macrotest_8bit -** STC small embedded memory macrotest to L1T	0: disabled (default) 1: scan test was enabled (can only be reset by PAD reset, **RESET_IN**)
stc_ipt_scan_divergence Bit 9	**stc_ipt_scan_divergence -** Scan divergence mode signal for chip	0: disabled (default) 1: scan test was enabled (can only be reset by PAD reset, **RESET_IN**)
stc_ipt_sog_scan Bit 8	**stc_ipt_sog_scan -** Sea of Gates scan mode signal	0: disabled (default) 1: scan test was enabled (can only be reset by PAD reset, **RESET_IN**)
stc_ipt_dig_radio_ext_scan Bit 7	**stc_ipt_dig_radio_ext_scan -** Dig_radio platform external scan test mode signal	0: disabled (default) 1: scan test was enabled (can only be reset by PAD reset, **RESET_IN**)

Name	Description	Settings
stc_ipt_dig_radio_int_scan Bit 6	**stc_ipt_dig_radio_int_scan -** Dig_radio platform internal scan test mode signal	0: disabled (default) 1: scan test was enabled (can only be reset by PAD reset, **RESET_IN**)
stc_ipt_onyx_ext_scan Bit 5	**stc_ipt_onyx_ext_scan -** Onyx platform external scan test mode signal	0: disabled (default) 1: scan test was enabled (can only be reset by PAD reset, **RESET_IN**)
stc_ipt_onyx_int_scan Bit 4	**stc_ipt_onyx_int_scan -** Onyx platform internal scan test mode signal	0: disabled (default) 1: scan test was enabled (can only be reset by PAD reset, **RESET_IN**)
stc_ipt_arm_ext_scan Bit 3	**stc_ipt_arm_ext_scan -** ARM platform external scan test mode signal	0: disabled (default) 1: scan test was enabled (can only be reset by PAD reset, **RESET_IN**)
stc_ipt_arm_int_scan Bit 2	**stc_ipt_arm_int_scan -** ARM platform internal scan test mode signal	0: disabled (default) 1: scan test was enabled (can only be reset by PAD reset, **RESET_IN**)
ipt_scan_mode Bit 1	**ipt_scan_mode -** chip in scan modes. OR of all scan mode signals coming out of STC	0: disabled (default) 1: scan test was enabled (can only be reset by PAD reset, **RESET_IN**)
stc_ipt_test Bit 0	**stc_ipt_test -** chip in test mode	0: disabled (default) 1: ipt_test was enabled (can only be reset by PAD reset, **RESET_IN**)

The Miscellaneous Test Control (MTCR) contains select and control bits for the new test modes in Neptune as well as the DSP control bits.

This 32-bit read/write register is accessible via the IP bus.

MTCR **Miscellaneous Test Control Register** **Addr $2484_4008**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
		tcm_dsp_selftime_wr_disable	tcm_ecc_disable			gpio_memtest_en	boost_test[1:0]				tcm_nsertst	tcm_ngtm				memtest_en
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 67-6. MTCR Description

Name	Description	Settings
Unused Bits 31-15	**Unused**	N/A
tcm_dsp_selftime_wr_disable Bit 14	**tcm_dsp_selftime_wr_disable-** Inhibits selftime write	0: allows selftime write 1:Inhibits selftime write
tcm_ecc_disable Bit 13	**tcm_ecc_disable-** Inhibits mcurom access during error correction.	0: allows mcurom access 1:Inhibits mcurom access
Unused Bits 12-11	**Unused**	N/A
gpio_memtest_en Bit 10	**gpio_memtest_en** - configures the 16 dataouts from GPIO to be datain. mode select for the memory controller	0: 16 datain and 16 dataout (default) 1: 32 datain
boost_test[1:0] Bits 9-8	**boost_test[1:0]** - Activates boost test to pads	00: 4 mA drive 01: 8 mA drive 10: 16 mA drive
Unused Bits 7-6	**Unused**	N/A
tcm_nsertst Bit 5	**tcm_nsertst** - Setup for DSP Master Mode and DSP Address Trace (see table below)	0: disabled (default) 1: Setup for DSP Master Mode and DSP Address Trace, see Table 67-7
tcm_ngtm Bit 4	**tcm_ngtm** - Setup for DSP Master Mode and DSP Address Trace (see table below)	0: disabled (default) 1: Setup for DSP Master Mode and DSP Address Trace, see Table 67-7

Table 67-6. MTCR Description

Name	Description	Settings
Unused Bits 3-1	**Unused**	N/A
Memtest_en Bit 0	**Memtest enable signal -** Only a dummy bit now, will leave asserted because the TCM transactor checks this bit to make sure it is on for Alternate Master Mode.	1: always asserted (default)

Table 67-7. DSP Test Modes

tcn_ngmt	tcm_nsertst	Function
0	0	Default
0	1	DSP Address Trace Mode Only
1	X	DSP Master Mode (including DSP Address Trace Mode)

The MCU Memory BIST Control Register (MBISTR) contains select and enable bits for Memory BIST operations.

This 32-bit read/write register is accessible via the IP bus only in AMT mode.

MBISTR MBist Control Register **Addr $2484_400C**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									mcu_ shared_x mem_en	mcu_ shared_y mem_en	mcu_ security _en	mcu_rom _en	mcu_ram _en	mcu_bist_mode _sel[1:0]		bist_ mode _en
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 67-8. MBIST Description

Name	Description	Settings
Bits 31-8	Unused	N/A
mcu_shared_xmem_en Bit 7	**mcu_shared_xmem_en** - enables the shared x-memory for bist testing	0: disable 1: Enable
mcu_shared_ymem_en Bit 6	**mcu_shared_ymem_en** - enables the shared y-memory for bist testing	0: disable 1: Enable
mcu_security_en Bit 5	**mcu_security_en** - enables the mcu security memory for bist testing	0: disable 1: Enable
mcu_rom_en Bit 4	**mcu_rom_en** - enables the mcu rom for bist testing	0: disable 1: Enable
mcu_ram_en Bit 3	**mcu_ram_en** - enables the mcu ram for bist testing	0: disable 1: Enable
mcu_bist_mode_sel[1:0] Bits 2-1	**mcu_bist_mode_sel[1:0]** - Selects which Bist mode to be in	00: Production Test mode 01: Debug mode 10: Bitmap mode 11: ROM MISR dataout mode
bist_mode_en Bits 0	**bist_mode_en** - Activates BIST Mode. Configures GPIO for BIST modes. ***Will be OR'd with Test pins?***	0: disables 1: enabled

This 32-bit read/write register is accessible via the IP bus. This register is used to control the ANATEST module and control the analog multiplexers.

ANA_CNTL Control Register **Addr $2484_4010**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
		ANA_EN1	ANA_EN0								tcm_TST_DC [5:4]		tcm_TST_DC[3:0]			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 67-9. ANA_CTL Description

Name	Description	Settings
Unused Bits 31-15	**Unused**	N/A
tcm_anatest_en1 Bit 14	**Anatest Enable1** — selects inputs from tcm or ana2dig.	0=tcm 1=ana2dig
tcm_anatest_en0 Bit 13	**Anatest Enable0** — Selects between the data reg 1 and 2	0=data1 1=data2
Unused Bits 12-6	**Unused**	N/A
tcm_TST_DC[5:4] Bits 5-4	**Anatest Control MUX2 —** Selects one out of 4 output connections.	See Table 67-10
tcm_TST_DC[3:0] Bits 3-0	**Anatest Control MUX1—** Selects one out of 16 signals from 8 blocks and bring out two outputs.	See Table 67-11

Analog Mux1

Table 67-10. Analog Mux1

TST_DC3	TST_DC2	TST_DC1	TST_DC0	Test_output select
0	0	0	0	RX_AFE_P,N
0	0	0	1	TUNEC_P,N
0	0	1	0	TX_P,N
0	0	1	1	PAC_P,N
0	1	0	0	SYN_P,N

Table 67-10. Analog Mux1

TST_DC3	TST_DC2	TST_DC1	TST_DC0	Test_output select
0	1	0	1	VOC_P,N
0	1	1	0	GPADC_P,N
0	1	1	1	CMON_P,N
1	0	0	0	REGUL_RXTX
1	0	0	1	Unused
1	0	1	0	REGUL_VOC
1	0	1	1	REGUL_SYN
1	1	0	0	REGUL_CORE
1	1	0	1	DET_TEST_MODE
1	1	1	0	Unused
1	1	1	1	Unused

ANALOG Mux2

Table 67-11. Analog Mux2

TST_DC5	TST_DC4	Output_type select
0	0	Unbuffered
0	1	Digital Buffer
1	0	Analog Buffer
1	1	LVDD

ANA_DATA1 and ANA_DATA2 are 32-bit read/write registers. The bits are input bits, programmed in to allow testing certain functionality of each block. The inputs are level shifted inside each block.

ANA_DATA1 — Data Register1 — **Addr $2484_4014**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
	TXSYN[5:0}						PAC[4:0]					CMON[4:0]				
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	TUNEC[7:0]								VOC[1:0]		DET_ TEST_ MODE	RXAFE[4:0]				
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ANA_DATA2 — Data Register2 — **Addr $2484_4018**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
					REG[7:0]								GPADC[3:0]			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 67-12. Data Register 1 Descriptions
Table 67-13.

Name	Description	Settings
TXSYN[5:0] Bits 31-26	**Transmitter and Synthesizer** - Input bits	0 = disable 1 = asserted
PAC[4:0] Bits 25-21	**Power Amplifier Control DAC/ADC** — Input bits.	0 = disable 1 = asserted
CMON[4:0] Bits 20-16	**System Clock Monitor/Clock Amplifier** — Input bits.	0 = disable 1 = asserted
TUNEC[7:0] Bits 15-8	**Tuning Circuit** — Input bits.	0 = disable 1 = asserted
VOC[1:0] Bits 7-6	**Voice Codec & Interface** — Input bits.	0 = disable 1 = asserted
DET_TEST_MODE Bit 5	**Saturation detect bit -** Input bit	0 = disable 1 = asserted
RXAFE[4:0] Bits 4-0	**Receiver Analog Front End —** Input bits.	0 = disable 1 = asserted

Table 67-14. Data Register 2 Descriptions

Name	Description	Settings
Unused Bits 31-12	**unused**	N/A
REG[7] Bits 11	**regul_core_test_in_p -** input	0 = disable 1 = asserted
REG[6] Bits 10	**regul_core_test_in_n -** input	0 = disable 1 = asserted
REG[5] Bits 9	**regul_codec_test_in_p -** input	0 = disable 1 = asserted
REG[4] Bits 8	**regul_codec_test_in_n -** input	0 = disable 1 = asserted
REG[3] Bits 7	**regul_synth_test_in_p -** input	0 = disable 1 = asserted
REG[2] Bits 6	**regul_synth_test_in_n -** input	0 = disable 1 = asserted
REG[1] Bits 5	**regul_txrx_test_in_p -** input	0 = disable 1 = asserted
REG[0] Bits 4	**regul_txrx_test_in_n -** input	0 = disable 1 = asserted
GPADC[3:0] Bits 3-0	**GPADC** - Inputs	0 = disable 1 = asserted

67.6.1 Register Description of Incoming Bist signals

In Neptune LTE, the MCU and DSP BIST engine outputs will come through the TCM. The TCM will register the incoming BIST signals so they will be available for observation in Alternate Bus Master mode. After being registered, the signals will go through to the GPIO.

Switching between BIST mode and Alternate Master Mode is achieved by toggling the master pin.

The MCU Incoming Bist Register is a 32-bit read register, accessible via the IP bus.

MCUInBist MCU Incoming Bist Register **Addr $2484_401C**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
													mcu_bist_done	mcu_bist_fail	mcu_shared_ymem_bist_done	mcu_shared_ymem_bist_fail
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	mcu_shared_xmem_bist_done	mcu_shared_xmem_bist_fail	mcu_security_bist_done	mcu_security_bist_fail	mcu_ram_bist_done	mcu_ram_bist_fail	mcu_rom_bist_done	mcu_rom_bist_fail	MCU Bist Incoming Data[7:0]							
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 67-15. MCUInBist Description

Name	Description	Settings
Unused Bits 31-20	**Unused**	N/A
mcu_bist_done Bit 19	**mcu_bist_done** — Status of the done flag, which is the combined result of the ram, rom and security ram done signals	0= Off 1= On
mcu_bist_fail Bit 18	**mcu_bist_fail** — Status of the fail flag, which is the combined result of the ram, rom and security ram done signals	0= Off 1= On
mcu_shared_ymem_bist_done Bit 17	**mcu_shared_ymem_bist_done** — Status of the done flag for the Shared Y-Memory	0= Off 1= On
mcu_shared_ymem_bist_fail Bit 16	**mcu_shared_ymem_bist_fail** — Status of the fail flag for the Shared Y-Memory	0= Off 1= On
mcu_shared_xmem_bist_done Bit 15	**mcu_shared_xmem_bist_done** — Status of the done flag for the Shared X-Memory	0= Off 1= On
mcu_shared_xmem_bist_fail Bit 14	**mcu_shared_xmem_bist_fail** — Status of the fail flag for the Shared X-Memory	0= Off 1= On
mcu_security_bist_done Bit 13	**mcu_security_bist_done** — Status of the done flag for the Security Memory	0= Off 1= On
mcu_security_bist_fail Bit 12	**mcu_security_bist_fail** — Status of the fail flag for the Security Memory	0= Off 1= On
mcu_ram_bist_done Bit 11	**mcu_ram_bist_done** — Status of the done flag for the RAM Memory	0= Off 1= On
mcu_ram_bist_fail Bit 10	**mcu_ram_bist_fail** — Status of the fail flag for the RAM Memory	0= Off 1= On
mcu_rom_bist_done Bit 9	**mcu_rom_bist_done** — Status of the done flag for the ROM Memory	0= Off 1= On
mcu_rom_bist_fail Bit 8	**mcu_rom_bist_fail** — Status of the fail flag for the ROM Memory	0= Off 1= On
mcu_bist_datain[7:0] Bits 7-0	**mcu_bist_datain[7:0]** - Status of the MCU Bist data coming into the TCM	

67.7 TCM Pin List

Table 67-16 lists all the pins in the TCM module.

Table 67-16. TCM Module Pin List

Pin Name	Direction	Comments
test0	Input	Chip level Primary Input. When “1” test mode. When “0” functional.
test1	Input	Chip level Primary Input.
test2	Input	Chip level Primary Input.
$\overline{\text{mcu\_de}}$	Input	Chip level Input, used for test modes when Test is asserted(active low)
$\overline{\text{dsp\_de}}$	Input	Chip level Input, used for test modes when Test is asserted(active low)
a2di_acc_ref0	Input	Input from a2di to generate tcm_anatest_en0
GPIO		
tcm_gpio_out[16:0]	Output	Selected Test outputs. (to GPIO). Muxed output of either DSP Master/Address trace mode signals, Scan data, Alternate Master data or MCU BIST data.
gpio_in_h[17:0]	Input	Input buses for Alternate Master Test module from GPIO. Data, and R/W, and Ctrl
gpio_in_l[37:0]	Input	Input buses for Alternate Master Test module from GPIO. Address and Data
DSP		
dsp_bab[15:0]	Input	DSP program address bus. from DSP.
dsp_ndeudp_trist	Input	Signal to tristate output pads. from DSP
IPS		
tcm_boost_test[1:0]	Output	Boost pads on board for test purpose. (to pads)
tcm_test_ts	Output	Output pads tristate enable. (to pads for leakage measurement)
tcm_ips_rdata[31:0]	Output	32-bit read data bus on IP bus
tcm_ips_xfr_err	Output	Data Error during transfer
tcm_ips_xfr_wait	Output	Transfer will not complete in current cycle
ips_rwb	Input	Read/Write signal
ips_byte_31_24	Input	Byte enable for data[31:24]
ips_byte_23_16	Input	Byte enable for data[23:16]
ips_byte_15_8	Input	Byte enable for data[15:8]

Table 67-16. TCM Module Pin List

Pin Name	Direction	Comments
ips_byte_7_0	Input	Byte enable for data[7:0]
ips_addr[11:2]	Input	System address bus
ips_module_en	Input	Module select signal
ips_wdata[31:0]	Input	Write data bus on IP bus
ipg_clk	Input	IP clock, from the CCM
ipg_async_hard_reset_b	Input	IP reset
pad_ext_reset_b	Input	PAD reset (RESET_IN). Used to reset the TCR Scan observability register bits.
SCAN		
stc_ipt_macrotest_mqspi	Input	Small embedded memory macrotest to mqspi. Comes from STC to be registered in TCM for observability
stc_ipt_macrotest_16bit	Input	Small embedded memory macrotest to L1T. Comes from STC to be registered in TCM for observability
stc_ipt_macrotest_8bit	Input	Small embedded memory macrotest to L1T. Comes from STC to be registered in TCM for observability
stc_ipt_scan_divergence	Input	Chip Scan divergence mode signal. Comes from STC to be registered in TCM for observability
stc_ipt_sog_scan	Input	Sea of Gates scan mode signal. Comes from STC to be registered in TCM for observability
stc_ipt_dig_radio_ext_scan	Input	Dig_radio platform external scan mode signal. Comes from STC to be registered in TCM for observability
stc_ipt_dig_radio_int_scan	Input	Dig_radio platform internal scan mode signal. Comes from STC to be registered in TCM for observability
stc_ipt_onyx_ext_scan	Input	Onyx platform external scan mode signal. Comes from STC to be registered in TCM for observability
stc_ipt_onyx_int_scan	Input	Onyx platform internal scan mode signal. Comes from STC to be registered in TCM for observability
stc_ipt_arm_ext_scan	Input	ARM platform external scan mode signal. Comes from STC to be registered in TCM for observability
stc_ipt_arm_int_scan	Input	ARM platform internal scan mode signal. Comes from STC to be registered in TCM for observability
stc_ipt_test	Input	Chip ipt_test signal. Comes from STC to be registered in TCM for observability
stc_scan_datout[15:0]	Input	STC Scan dataout [16:1] going through Bus Mux to GPIO
ipt_clk_se	Input	Scan enable, works as test clock control

Table 67-16. TCM Module Pin List

Pin Name	Direction	Comments
ipt_scan_mode	Input	Chip Scan mode signal.
tcm_master_test	Output	Test gated with MASTER for GPIO control purpose (to GPIO)
tcm_test_lat	Output	Test signal latched with falling edge of CCM_MCU_CLK to CCM.
Memory BIST		
tcm_mcu_sharedxmem_bist_en	Output	Enables the MCU Shared X-Memory for BIST testing
tcm_mcu_sharedymem_bist_en	Output	Enables the MCU Shared Y-Memory for BIST testing
tcm_mcu_security_bist_en	Output	Enables the MCU Security memory for bist testing
tcm_mcurom_bist_en	Output	Enables the MCU ROM for bist testing
tcm_mcuram_bist_en	Output	Enables the MCU RAM for bist testing
tcm_mcu_bist_mode_sel[1:0]	Output	Selects between different Bist modes: Production, Bitmapping, Debug or MISR
tcm_bist_mode_en	Output	Enable Bist mode. Is the OR of all bist modes
tcm_mbist_prod_mode	Output	Configures GPIO for BIST Production mode. Able to do DSP Bist Production in parallel
tcm_mbist_bitmap_mode	Output	Configures GPIO for MCU BIST Bitmapping mode.
mcu_bist_datain[7:0]	Input	MCU bist data coming from MCU BIST engine
mcu_bist_fail	Input	MCU bist fail flag, this input is the combined result of the ram, rom and security ram fail signals
mcu_bist_done	Input	MCU bist done flag, this input is the combined result of the ram, rom and security ram done signals
mcu_rom_bist_fail	Input	MCU ROM bist fail flag
mcu_rom_bist_done	Input	MCU ROM bist done flag
mcu_ram_bist_fail	Input	MCU RAM bist fail flag
mcu_ram_bist_done	Input	MCU RAM bist done flag
mcu_security_bist_fail	Input	MCU Security bist fail flag
mcu_security_bist_done	Input	MCU Security bist done flag
mcu_shared_xmem_bist_fail	Input	MCU Shared x-mem bist fail flag
mcu_shared_xmem_bist_done	Input	MCU Shared x-mem bist done flag
mcu_shared_ymem_bist_fail	Input	MCU Shared y-mem bist fail flag
mcu_shared_ymem_bist_done	Input	MCU Shared y-mem bist done flag

Table 67-16. TCM Module Pin List

Pin Name	Direction	Comments
tcm_dsp_bist_bypass_mode	Output	DSP Bist Bypass mode signal. Will be used to bypass DSP ABIST.
ANATEST		
tcm_anatest_data1[31:0]	Output	Selected data inputs to the analog blocks f/ Anatest data1 register
tcm_anatest_data2[11:0]	Output	Selected data inputs to the analog blocks f/ Anatest data2 register
tcm_tst_dc[5:0]	Output	Anatest control signals for selecting analog outputs
tcm_anatest_en1	Output	Anatest enable signal
tcm_anatest_en0	Output	Anatest enable signal
Misc. Test		
tcm_iim_reset_b	Output	Reset signal for the IIM module
tcm_dsp_selftime_wr_disable	Output	Inhibits selftime write for DSP
tcm_ecc_disable	Output	Inhibits mcurom access during error correction
tcm_ngtm	Output	Control signal for DSP Master Mode and DSP Address Trace to GPIO
tcm_nsertst	Output	Control signal for DSP Master Mode and DSP Address Trace to GPIO
Alternate Master Mode		
ips_xfr_err	Input	Transfer error signal coming through TCM out to GPIO. Used during Alternate Master Mode (through Bus Mux) to monitor whether Register Address accesses are valid. (Look at DDTS DSPhl11056)
am_hrdata[31:0]	Input	Read Data bus
gpio_master	Input	GPIO as input when TEST0 is asserted. Allows AMT mode on ALT.
tcm_gpiomemtest_en	Output	Enable for configuration of GPIO data in AMT mode
tcm_am_haddr[31:0]	Output	Address bits for the memory
tcm_am_hsize[1:0]	Output	0 0 - Byte Data 0 1 - Half word 1 0 - Word
tcm_am_hwdata[31:16]	Output	Alternate Master output data bus
tcm_am_htrans[1:0]	Output	Type of transfer bits to Alt. Master bus
tcm_am_hprot[1:0]	Output	Protection bits to Alt. Master bus

Table 67-16. TCM Module Pin List

Pin Name	Direction	Comments
tcm_am_hwrite	Output	Write bit to Alt. Master bus
tcm_am_hbusreq	Output	Bus request to Arbiter

NOTE:

There are no bi-directional signals in the TCM module.

67.8 Clocks During Test Modes

The TCM will switch from functional mode to Test mode, when the Test signal is asserted, refer to Section 67.3.5 for Test signal definition. The Test clock is driven from a dedicated GPIO pin already set as a primary input. The AMT Mode is clocked by a gated MCU_CLK. Scan Test for new modules are all synchronized on this external test clock. This clock control during test mode do not affect the following domain:

- For DSP stand alone test mode, we still have the control on the clock sourcing. ROSC or Bypass clock through GPIO is the source, depending on the setup of the bypass control.

67.8.1 Clocks During AMT Mode

When the AMT is in its Idle state (i.e. MASTER is negated), the AM_CLK is disabled to reduce power consumption.

The AMT clock is the MCU_CLK gated with the MASTER signal to reduce power consumption in the application when AMT Mode is not operating.

67.8.2 Clocks During MBIST Mode

Please refer to Appendix B, “Design for Testabilty Methodology (DFT).”

67.9 Alternate Master Test Module (AMT)

This section describes the Alternate Master Test (AMT) module which is a subset of the TCM.

67.9.1 Overview

The Alternate Master Test (AMT) module can request and be granted control of the R-AHB to allow direct access of peripherals and memories for functional testing. Since these tests are performed independent of the ARM7 core each peripheral or memory can also be characterized for speed performance.

67.9.2 Features

External connection to the AMT is an interface using GPIO port pins when configured in AMT Mode. By using GPIO port pins and the AMT the External Interface Module (AEIM) can be tested and external memory to be accessed from the AMT. There are 22 GPIO pins for address, 16 GPIO pins for datain and

dataout, one GPIO pin read/write, and one GPIO pin for EXEC. When in mode one, the 16 GPIO dataout pins are reconfigured to be an additional 16 datain pins, thus allowing a full 32 bits of datain for write functions.

- Alternate master access to the R-AHB
- Combination Serial/Parallel interface speeds test time
- Read/Write access to memory mapped devices on the R-AHB
- Support for all size transfers
  - Word
  - Half-word
  - Byte
- Support for all transfer types
  - User or Privileged
  - Data access or Opcode fetch
  - Sequential or Non-Sequential
- External interface through GPIO module pins
- AMT modes, 16 or 32-bit

67.9.3 Block Diagrams

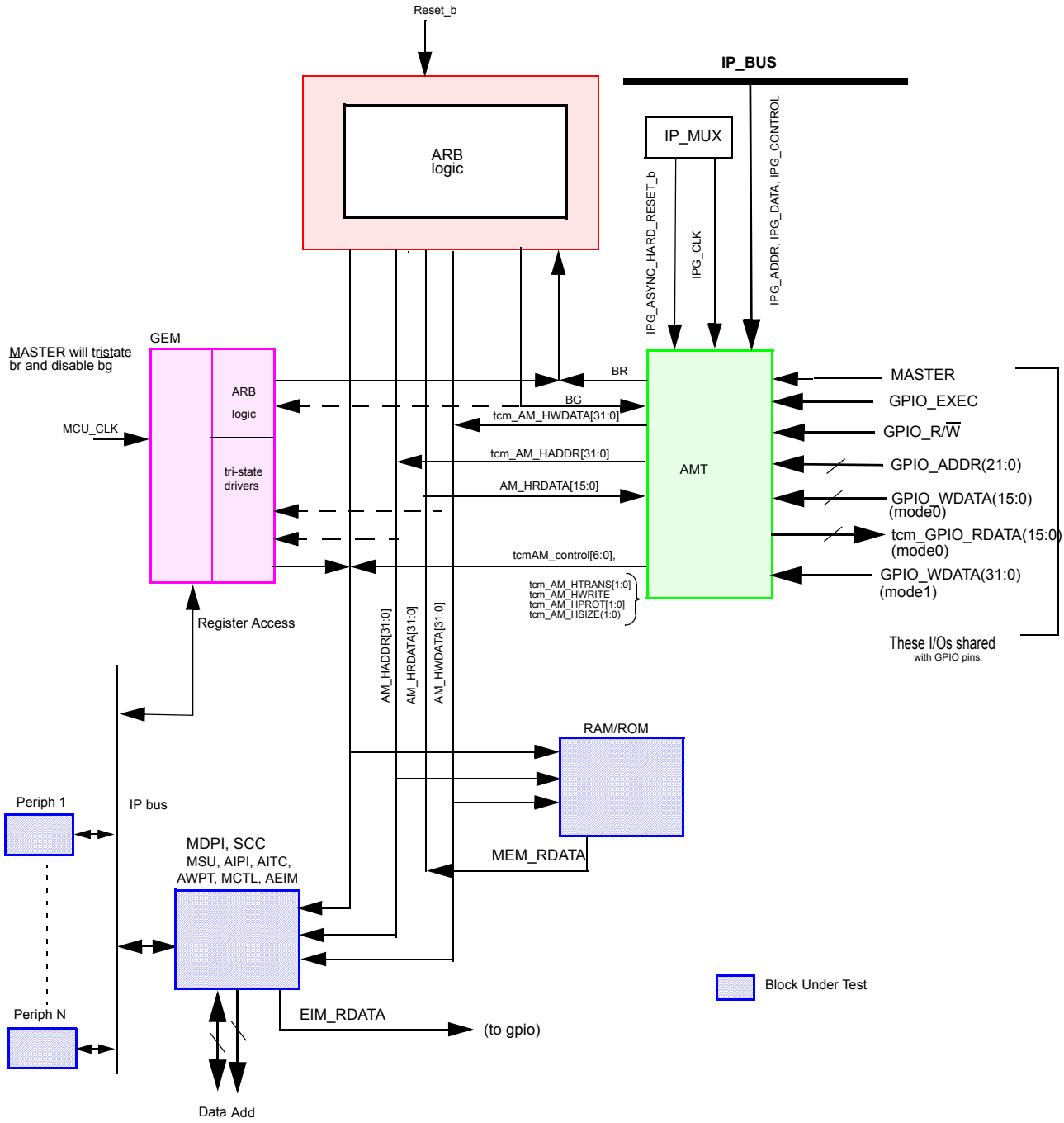


Figure 67-3. Alternate Master Test (AMT) Mode Block Diagram

67.9.4 AMT Register Map

There are no memory mapped registers in the AMT Module. This block is simply an alternate master of the R-AHB in a special test mode to allow access of the memory and peripherals through GPIO port pins. All registers are accessible directly from the external GPIO pins.

67.9.5 Functional Description

The ARM7 allows alternate master devices to obtain ownership of the R-AHB through bus arbitration logic. The AMT becomes a valid alternate master to access memory mapped devices on the R-AHB. The AMT requests ownership via the bus request BR signal and is granted ownership by the arbiter with the assertion of the bus grant BG signal once any outstanding accesses are resolved.

R-AHB arbitration is not available during reset. Therefore, the AMT may not perform arbitration sequences while the ARM7 is held in reset. Once reset is released the assertion of BG will occur if the $\overline{BR}$ input had been previously asserted and remains so.

67.9.5.1 R-AHB Arbitration

The AMT module and the GEM(GPRS Encryption Module) can be alternate masters of the R-AHB. However, only one device at a time can request ownership of the R-AHB. In Neptune, R-AHB arbitration is handled by the CARB and AMARB modules.

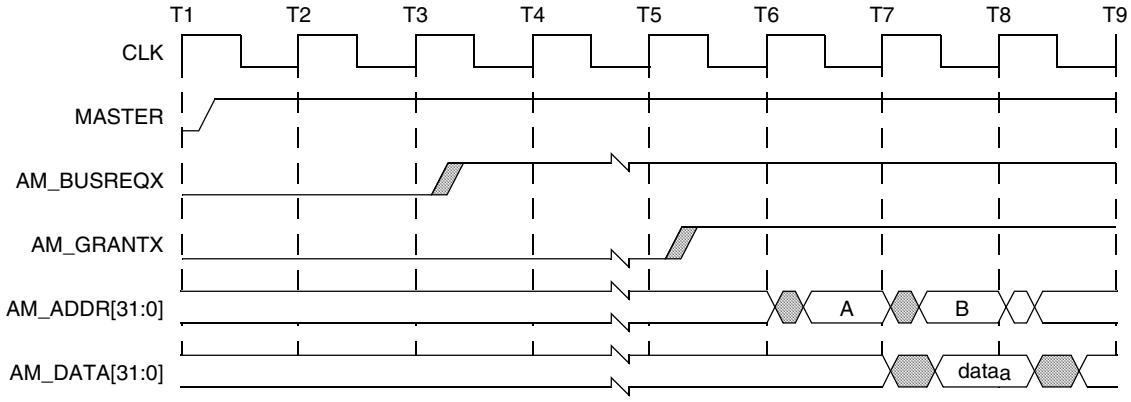


Figure 67-4. Bus Arbitration BR-> BG

67.9.5.2 AMT Mode

Once the AMT has been granted control of the bus the attributes, address, and data are latched into the AMT to begin an access. These I/O's are shared with GPIO pins. The attributes, address, and data in are latched into AMT registers in one cycle (for 16 bit data read/writes or mode 0). In mode 1, 32 bits of data, address, and control are latched in one cycle. While EXEC is low (deasserted), set the base address on the lower 10 bits of the GPIO address pins (gpio_addr[9:0]), the size of the transaction on the next two pins (gpio_addr[11:10]), the protection signals on the next two pins (gpio_addr[13:12]), and the transaction on the next pin (gpio_addr[14]). If the transaction pin is set (gpio_addr[14]), the transaction will be sequential otherwise it will be nonsequential. The base address and the controls are latched and held when EXEC is asserted and starts the bus transaction. During 16-bit mode reads after bus transactions begin, the gpio_addr[1] determines which 16 bits of data from the 32 bit data bus are read. If gpio_addr[1] is asserted, then the lower 16 bits are read, otherwise the upper 16 bits are read. Figure 67-7 on page 67-37 shows writes and reads in 16-bit mode (mode0) with no wait states. Figure 67-8 on page 67-38 shows writes and reads in 32-bit mode (mode1) with no wait states.

To handle accesses with wait states, EXEC is asserted for the second cycle of the input shift in the attributes, address, and datain as in the no wait state case. Then EXEC will not be asserted again until one cycle plus the number of wait states. Figure 67-5 on page 67-35 shows write and reads in 16-bit mode (mode0) with two wait states. Figure 67-6 on page 67-36 shows write and reads in 32-bit mode (mode1) with two wait states.

To get into mode1 (32-bit write), the gpio_memtest_en bit in the MTCR must be set, otherwise the TCM is in mode0.

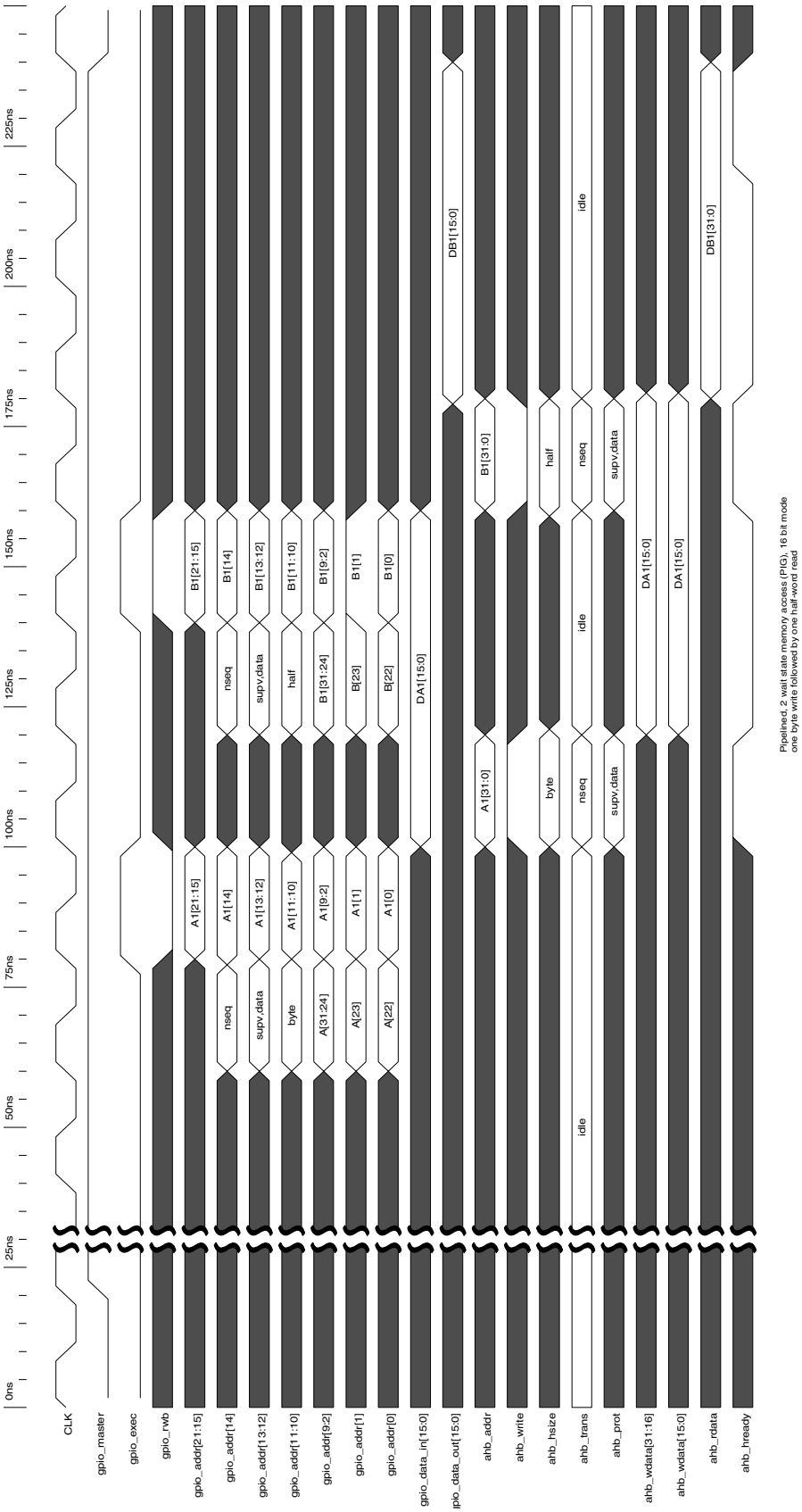


Figure 67-5. Pipelined, 2 wait state memory access (PIG), 16-bit mode

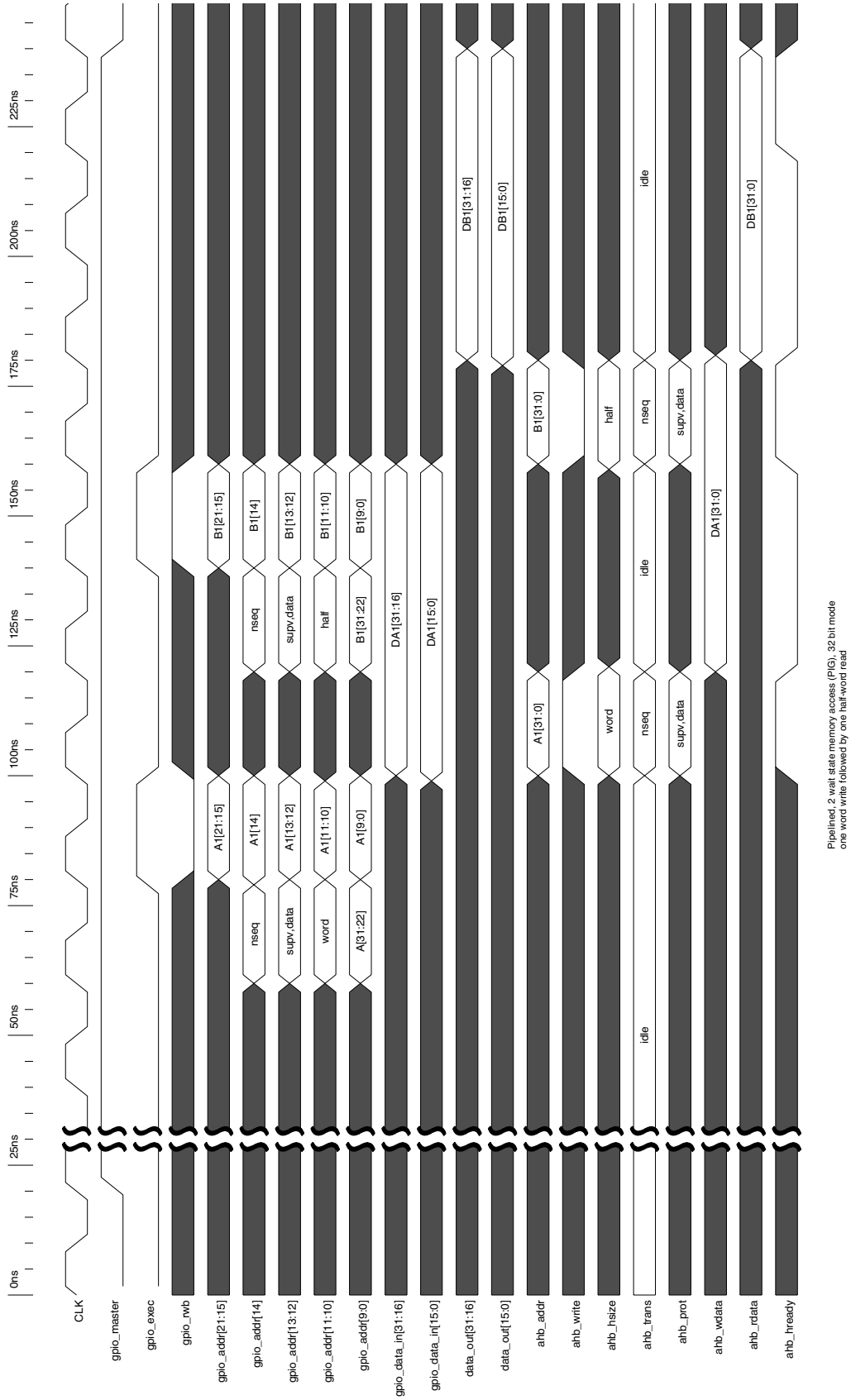


Figure 67-6. Pipelined, 2 wait state memory access (PIG), 32-bit mode

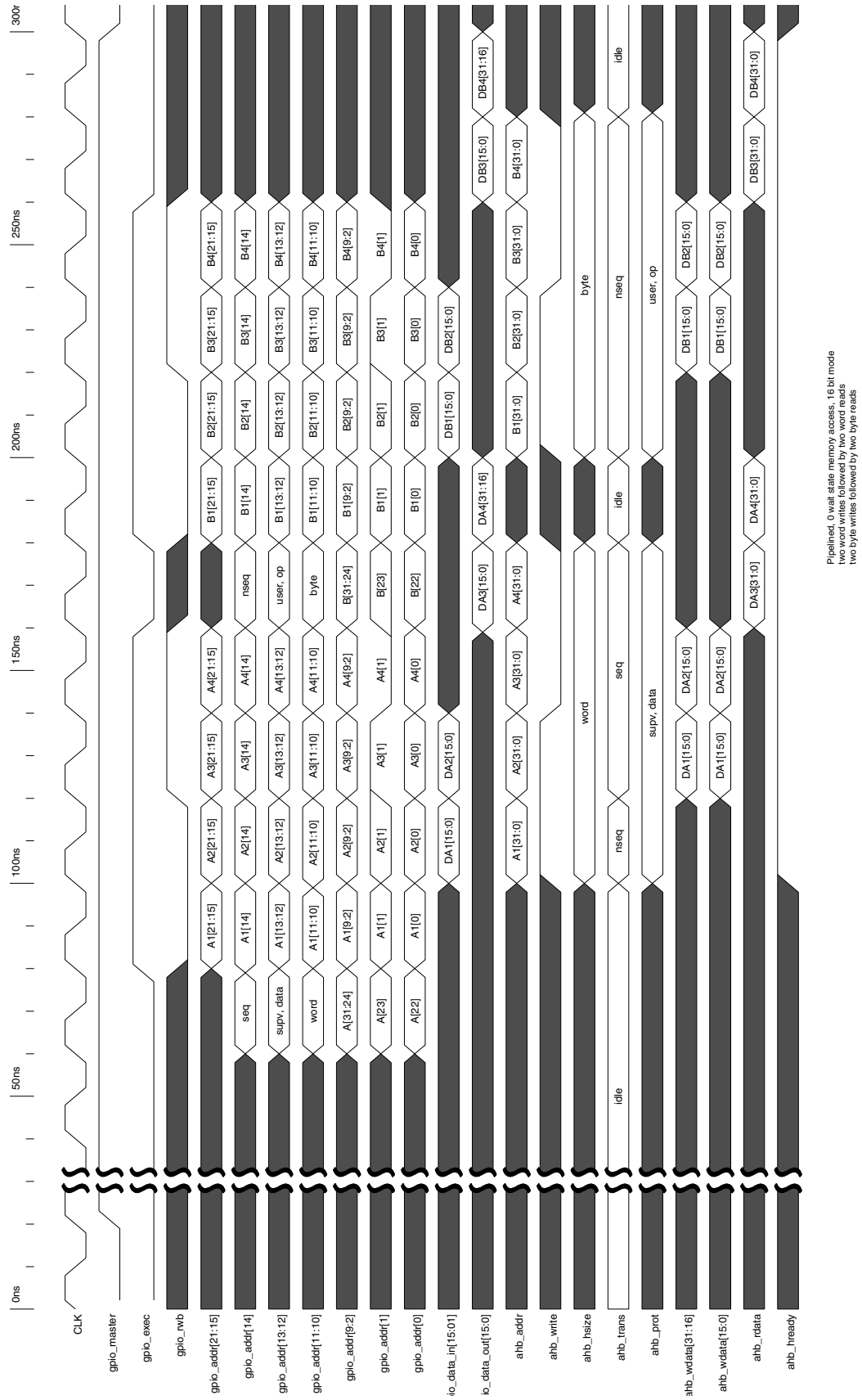


Figure 67-7. Pipelined, 0 wait state memory access, 16-bit mode

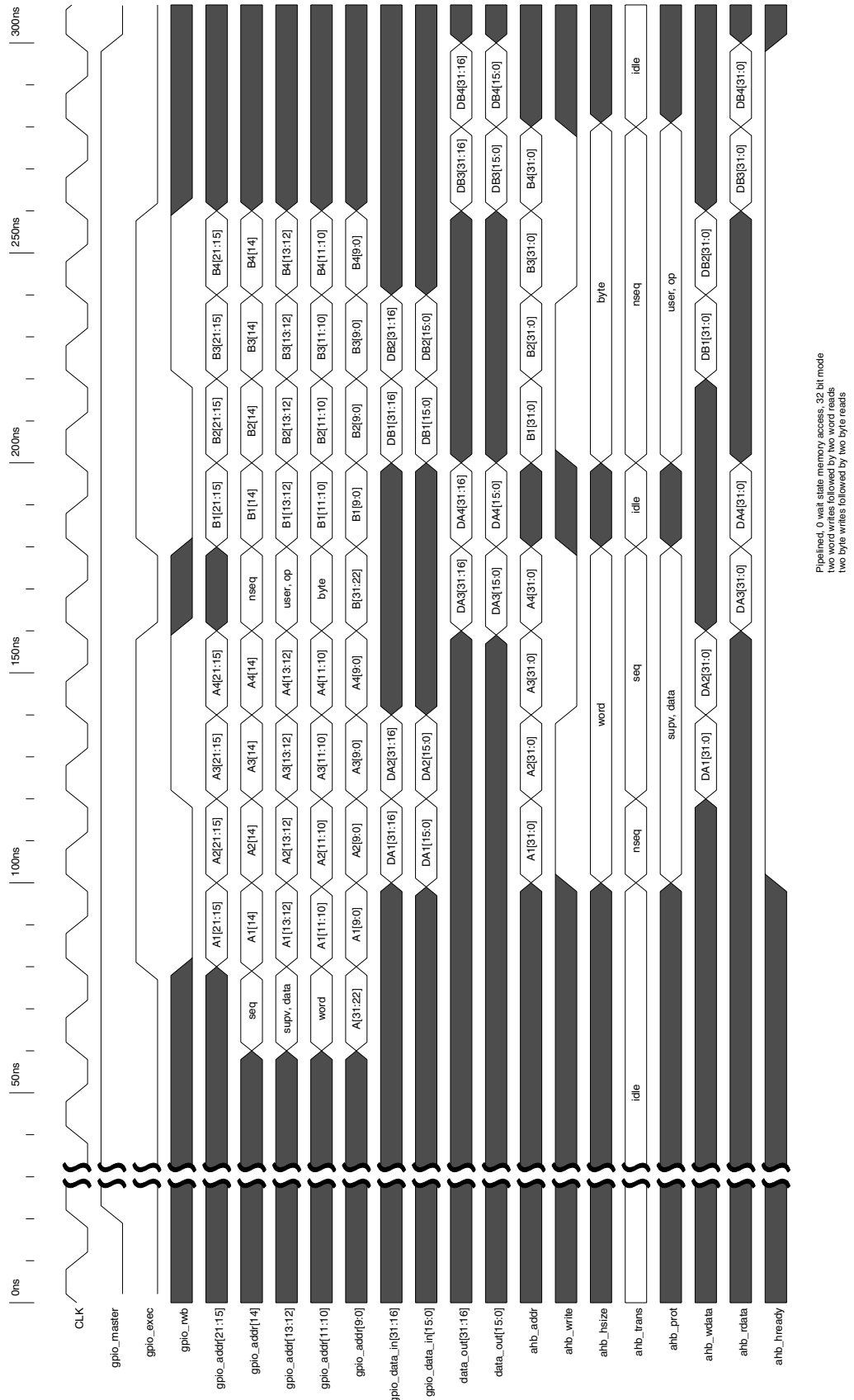


Figure 67-8. Pipelined, 0 wait state memory access, 32-bit mode

67.9.5.3 I/O Registers

The I/O registers shift in the Alternate Master Bus attributes, Alternate Master address, and Alternate Master write data in one clock cycle and then drives these onto the Alternate Master Bus on the rising edge of the clock when EXEC is asserted. During a read, the read data goes straight through to the GPIO pinouts.

The I/O Registers in the AMT are not memory mapped or accessible to the ARM7 or the DSP.

NOTE:

All bits of all registers in this module are completely synchronous to internal clocks.

67.9.6 External Pin Descriptions

GPIO_MASTER

Initiates the bus request for the AMT to take control of the R-AHB. It forces the necessary GPIO signals to the appropriate direction for the AMT and asserts the bus request to the R-AHB arbitration logic.

GPIO_EXEC

This signal is asserted on the second cycle of the input shift to start the R-AHB.

GPIO_RWB

The attributes of the bus transfer

— R/$\overline{W}$: An attribute that indicates if the access is a read or write.

GPIO_ADDR[21:0]

The address of the access shifted in.

GPIO_DATAIN[15:0]

The input data for a write.

tcm_AM_DATAOUT[15:0]

The output data from a read.

67.9.7 Modes of Operation

There no special modes of operation for the AMT. It is either in AMT Mode in which it has control of the R-AHB or it is idle.

67.9.8 Interrupt Operation

This module does not generate interrupts.

67.9.9 AMT Module Pin List

Table 67-17. AMT Module Pin List

Inputs	Outputs
gpio_master test gpio_EXEC gpio_ADDR[21:0] gpio_DATAIN[15:0] gpio_WRITE ipg_clk ipg_async_hard_reset_b tcm_gpiomemtest_en am_hgrant am_hready am_hrdata[31:0]	tcm_am_hwdata[31:0] tcm_am_haddr[31:0] tcm_am_htrans[1:0] tcm_am_hsize[1:0] tcm_am_hprot[1:0] tcm_am_hwrite tcm_am_hbusreq tcm_gpio_dataout[31:16]

Chapter 68
Scan Test Controller (STC)

Revision History Table

Revision	Date	Author	Changes
0.0	03/20/2002	Kelvin Ge	Initial Release.
1.0	04/18/2002	Kelvin Ge	Updated Release - Updated the test mode table. - Added macrotest modes section.
1.1	05/03/2002	Kelvin Ge	Added reset signals, res and pires, from sonyxu platform and the muxing logic for scan mode.
1.2	05/07/03		Updated for LTE specification release.

68.1 Module Overview

The Scan Test Controller (STC) is a new module for Neptune LTE. It provides various scan test mode entry signals through purely combinational logic. Therefore, scan test modes can be configured on a per clock basis if required. This can be useful during debug operations.There will be five main scan domains on the LTE chip: SONYXU, ARM7_PLATFORM, Sea Of Gates (SOG), Dig_Radio, and Scan Divergence Modes. In addition, there are two test modes for embedded memory testing, the Embedded memory test mode (0 and 1), which provide the test control signals for memory macrotests in L1T and MQSPI modules. The DSP_PLATFORM will be a part of SOG in LTE. Each scan domain needs to be put into an isolation mode during scan testing. While the scan test control signals are generated in the STC, the test control signals for MBIST modes are still remained in the Test Control Module (TCM). STC provides the following test modes and functions for scan test.

1. Main scan domain test modes
   - — Arm7_platform scan test mode
   - — Sonyxu scan test mode
   - — SOG scan test mode
   - — Dig_Radio scan mode
   - — Embedded memory test mode (0 and 1)
2. Scan divergence mode

In scan divergence mode, all scan chains are connected forming a single long scan chain for debug and failure analysis.

3. IDDQ test mode

   IDDQ test mode is enable in conjuction with scan divergence mode.

4. Scan chain routing

   The scan chain router in STC provides the logic to route the appropriate scan chains to the Chip I/O pads per each scan test mode. The router will stich all scan chains for each test mode into a single scan chain in for the scan_divergence mode at chip level.

5. Reset signal muxing for scan mode on ‘res’ and ‘pires’ from SONYXU.

68.1.1 General Description

The scan test controller for Neptune LTE is completely changed from Neptune LT. The scan test controller will be a separate control unit removing the scan test control from the TCM. The Scan Test controller will not provide the control signals for the MBIST modes which will still be controlled from the TCM. The LTE scan test controller will contain the logic for scan control including a combinational test mode entry, the scan chain router, and additional scan control for SOG. A bock diagram of the top level scan test controller is shown below in Figure 68-1.

There are three main blocks in the STC.

- Scan Control Unit - Generate all scan test mode signals.
- Scan Chain Router - Provide scan chain routing at chip level.
- Scan Logic Additions - Provide reset signal muxing for ‘res’ and ‘pires’ from SONYXU.

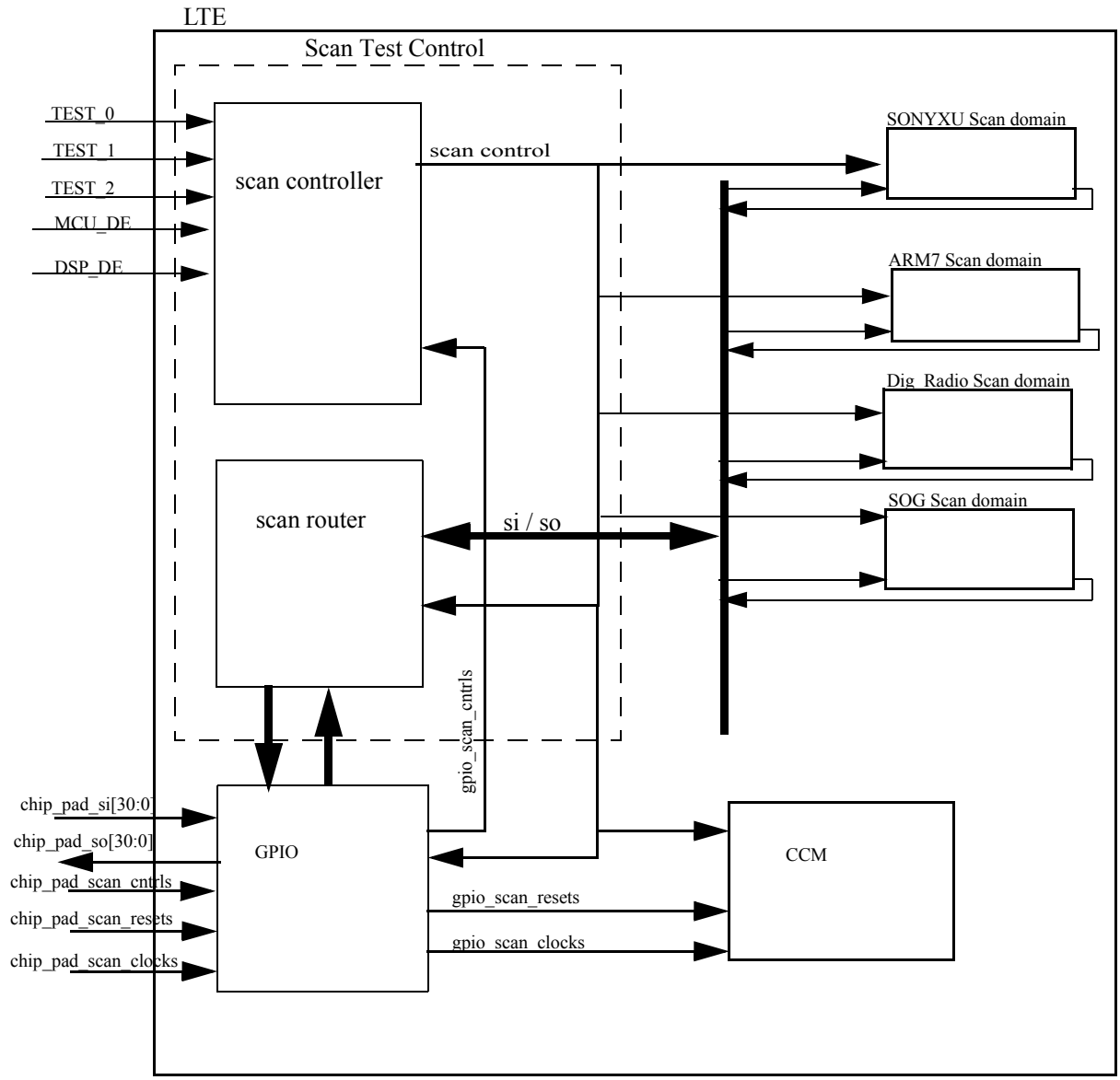


Figure 68-1. LTE Scan Test Controller Block Diagram

68.2 Scan Control Unit

The scan control unit contains the logic for scan mode entry, and to generate the necessary control signals for all scan mode required on the chip. For the LTE the scan mode entry will be changed to be a combinational logic entry. Besides simplifying the scan entry used in the atpg tools a combinational test mode entry allows more flexibility during silicon debug, there is less circuitry required to enter test mode as opposed to functional entry to bus master mode. A combinational test mode entry also supports

scan_divergence for debug. The application of scan_divergence is to have the ability to stop the clock during a functional run, enter scan_divergence test mode and shift out the contents of all reg's from the point the clock was stopped.

For Neptune LT, the test mode entry and scan control are part of the ip_platform in TCM. For Neptune LTE, the scan_test entry and control needs to be removed from the ip_platform logic, but all other test mode will stay. The initial entry into scan test will use the same entry as the Neptune LT, but a new test enable signal, TEST, is used instead of TEST0, which is a combinationa of TEST0/1/2. In addition to the three TEST pins, additional two input pins for scan_test_mode signal configuration will be allocated, and they are the MCU_DE and DSP_DE pins. When TEST is asserted, the MCU_DE and DSP_DE pins are configured as input pad pins only.

68.2.1 Neptune LTE Test Modes

The following table shows all the test modes available for Neptune LTE. A new TEST signal is generated and taking the similar role, but not exactly, of TEST0 in LT.

TEST= (TEST0 + TEST1) & ~(TEST2 & ~TEST1 & ~TEST0)

When TEST=0, the chip is in normal functional mode. While TEST=1, the chip is in test modes (including scan test and BIST). Table 68-1 shows all the combinations available from the five inputs, of which five combinations are used for scan test mode control signals including the memory macrotest.

Table 68-1. All the test modes available for Neptune LTE

TEST2	TEST1	TEST0	MCU_DE	DSP_DE	Test Modes
0	0	0	x	x	normal functional
0	0	1	0	0	***Reserved for master mode***
0	0	1	0	1	ARM7 Platform scan mode
0	0	1	1	0	SONYX Platform scan mode
0	0	1	1	1	SOG scan mode
0	1	0	0	0	DIG_RADIO Platform scan mode
0	1	0	0	1	Scan Divergence mode (includes IDDQ)
0	1	0	1	0	Embedded Memory Macrotest Mode 0
0	1	0	1	1	Embedded Memory Macrotest Mode 1
0	1	1	0	0	BIST modes (see TCM specs)
0	1	1	0	1	BIST modes (see TCM specs)
0	1	1	1	0	BIST modes (see TCM specs)
0	1	1	1	1	BIST modes (see TCM specs)
1	0	0	X	X	***Resvered for User_mode***
1	0	1	0	0	BIST modes (see TCM specs)
1	0	1	0	1	BIST modes (see TCM specs)
1	0	1	1	0	BIST modes (see TCM specs)
1	0	1	1	1	BIST modes (see TCM specs)
1	1	0	0	0	BIST modes (see TCM specs)
1	1	0	0	1	BIST modes (see TCM specs)
1	1	0	1	0	BIST modes (see TCM specs)
1	1	0	1	1	BIST modes (see TCM specs)
1	1	1	0	0	BIST modes (see TCM specs)
1	1	1	0	1	BIST modes (see TCM specs)
1	1	1	1	0	BIST modes (see TCM specs)
1	1	1	1	1	BIST modes (see TCM specs)

68.2.2 Scan Test Modes

In Neptune LTE, the STC module generates the scan test mode signals for the five main scan domains on the LTE chip: SONYXU, ARM7_PLATFORM, Sea Of Gates (SOG) including DSP_PLATFORM, Dig_Radio, and scan divergence. The test mode signals are generated through 5 primary input pins: TEST0, TEST1, TEST2, MCU_DE and DSP_DE. The STC creates and drives the outputs of a combinational block asserting the modes as shown in the table Table 68-1.

Note: SCKA (master pin) needs to be 0 in test modes. SCKA, DSP_DE, and MCU_DE are bidirectional pins. They need to be configured as inputs for test modes configuration. In LT tcm sends test_latch to gpio, configure scka to input pin. dsp_de and mcu_de were not used for test mode configuration in LT.

68.2.3 DSP Master Mode

(see the TCM specs).

68.2.4 MBIST Test Modes

(see TCM spec for details).

68.2.5 Scan Operations

For the five scan domains in Neptune LTE, each scan domain needs to be put into an isolation mode during scan testing. For the platforms in the Neptune LTE, test wrappers are implemented to increase the core's testability and maximize test converage. Therefore, the platforms consist of core and test wrapper. And both the core and wrapper are scanned in the platforms. The SOG block does not have a test wrapper. The wrappers of the platforms will be tested in the SOG scan test mode. Table 68-2 shows all the scan test mode signals generated by STC, and Table 68-3 shows the scan test signal states in different scan test modes.

68.2.5.1 Scan Test Clock and Test Reset

In Neptune LTE, there will be no separated scan test clock and test reset. The functional clocks and resets will be used in scan mode. Due to the multiple functional clocks used and pin limilations in the LTE, three fuctional clocks will be chosen and used for test clock during scan test. This leads to 4 test clock domains, and they are:

- ccm_104clk - 104MHz test clock From CCM
- ccm_52clk - 52MHz test clock From CCM
- ccm_26clk - 26MHz test clock From CCM
- ccm_13clk - 13MHz test clock From CCM

68.2.5.2 Chip In Scan Mode

Once in test mode (TEST=1), the **ipt_scan_mode** signal is always active (high) no matter which scan domain or block is under scan testing. This scan mode signal indicates if the chip is in scan test mode.

68.2.5.3 Platform Internal Scan Test

In each platform, the core and wrapper are scan tested separately. The internal scan test is on the core and the external scan test is on the wrapper.

In the internal scan test mode, wrapper cell supplies control to input signals from Platform, and captures data on output signals of Platform. Wrapper cells supply safe state values on all output signals from the Platform. Internal test mode is activated via **ipt_xxx_int_scan** signal during the platform scan test mode. The followings are the Internal test mode sigals for platforms.

- ipt_arm_int_scan - arm platform internal scan test mode
- ipt_onyx_int_scan - onyx platform internal scan test mode
- ipt_dig_rad_int_scan - dig_radio platform internal scan test mode

68.2.5.4 Platform External Scan Test

In the platform external scan test mode, wrapper cell supplies control on output signals from Platform, and captures data on input signals to Platform. Wrapper cell supplies safe state values to Platform input signals. The wrapper cells are scan tested during the SOG scan test mode (described below). Therefore, the External test mode is activated via **ipt_xxx_ext_scan** signal during the SOG scan test mode. The followings are the external test mode sigals for platforms.

- ipt_arm_ext_scan - arm platform external scan test mode
- ipt_onyx_ext_scan - onyx platform external scan test mode
- ipt_dig_rad_ext_scan - dig_radio platform external scan test mode

68.2.5.5 SOG Scan Test

In the SOG scan test mode, all the modules in the SOG will be in scan test mode. In addition, all the platform external test mode signals will be active (high) also. During the SOG scan testing, the platform wrappers are being tested in platform external scan test modes.

68.2.5.6 Scan Divergence Mode and IDDQ Test

The STC provides a scan divergence signal, **ipt_scan_divergence,** for silicon debug. In the scan divergence mode (when ipt_scan_divergence is high), all the scan chains in the LTE will be concatenated into a single long scan chain. This will allow shifting out the contents of all scan flops.

For the LTE DFT methodology, the IDDQ mode will be combined with the scan divergence mode. When scan divergence mode is activated, an IDDQ signal will be generated for use of disabling pull up / down's and 3-state devices.

68.2.5.7 Small Embedded Memory Macrotest Modes

The small embedded memory macrotests are performed on the small embedded memory blocks in L1T and MQSPI modules in Neptune LTE. There are two small embedded memory blocks (8-bit and 16-bit) in L1T and one small embedded memory block in MQSPI. The macrotest is a capability of Mentor Graphics' Fastscan to get the memories tested and improve the test coverge on the overall chip, which is a test carried out in scan mode operation. Since L1T and MQSPI are in the Sea of Gates in LTE, the macrotests will be performed in the SOG scan test mode. In the SOG scan test mode, all the modules in the SOG will be in scan test mode. In addition, all the platform external test mode signals will be active (high) also. During macrotest, like the SOG scan testing, the platform wrappers are being tested in platform external scan test modes.

To improve the testability and shorten the test times, two macrotest modes are impelemented in such a way that parallel tests are realized on the small embedded memory blocks in L1T and MQSPI. The descriptions of the macrotest modes are follows.

68.2.5.7.1 Small Memory Macrotest mode 0

In this mode, the 8-bit memory block in L1T and the memory block in MQSPI will tested at the same time through Macrotest. The followings are the control signals and active in this mode.

- ipt_macrotest_8bit - To L1T module.
- ipt_macrotest_mqspi- To MQSPI module.

68.2.5.7.2 Small Memory Macrotest mode 1

In this mode, the 16-bit memory block in L1T and the memory block in MQSPI will tested at the same time through Macrotest. The followings are the control signals and active in this mode.

- ipt_macrotest_16bit - To L1T module.
- ipt_macrotest_mqspi - To MQSPI module.

Table 68-2. Scan Test Mode Description

Name	Description	Settings
ipt_scan_mode	chip in scan modes.	0: disabled (default) 1: scan mode enabled
ipt_arm_int_scan	arm platform internal scan test mode	0: Internal Scan Test disabled. 1: Internal Scan test enabled.
ipt_arm_ext_scan	arm platform external scan test mode	0: External Scan Test disabled. 1: External Scan test enabled.
ipt_arm_scan_en	arm platform scan enable	0: Capture during Scan Test. 1: Shiftting during Scan Test.
ipt_arm_wrapper_se	arm platform wrapper scan enable	0: Capture during Scan Test. 1: Shiftting during Scan Test.
ipt_onyx_int_scan	onyx platform internal scan test mode	0: Internal Scan Test disabled. 1: Internal Scan test enabled.
ipt_onyx_ext_scan	onyx platform external scan test mode	0: External Scan Test disabled. 1: External Scan test enabled.
ipt_onyx_scan_en	onyx platform scan enable	0: Capture during Scan Test. 1: Shiftting during Scan Test.
ipt_onyx_wrapper_se	onyx platform wrapper scan enable	0: Capture during Scan Test. 1: Shiftting during Scan Test.
ipt_dig_rad_int_scan	dig_radio platform internal scan test mode	0: Internal Scan Test disabled. 1: Internal Scan test enabled.
ipt_dig_rad_ext_scan	dig_radio platform external scan test mode	0: External Scan Test disabled. 1: External Scan test enabled.
ipt_dig_rad_scan_en	dig_radio platform scan enable	0: Capture during Scan Test. 1: Shiftting during Scan Test.
ipt_dig_rad_wrapper_se	dig_radio platform wrapper scan enable	0: Capture during Scan Test. 1: Shiftting during Scan Test.
ipt_sog_scan	SOG scan test mode	0: SOG Scan Test disabled. 1: SOG Scan test enabled.
ipt_sog_scan_en	SOG scan enable	0: Capture during Scan Test. 1: Shiftting during Scan Test.
ipt_scan_divergence	chip in scan divergence mode	0: disabled. 1: scan divergence enabled.
ipt_macrotest_8bit	small embedded memory macrotest to L1T	0: disabled. 1: macrotest-8bit enabled.
ipt_macrotest_16bit	small embedded memory macrotest to L1T	0: disabled. 1: macrotest-16bit enabled.
ipt_macrotest_mqspi	small embedded memory macrotest to mqspi	0: disabled. 1: macrotest-mqspi enabled.

Table 68-3. Scan Control Signal Generation

scan test mode	ipt_scan_mode	ipt_arm_int_scan	ipt_arm_ext_scan	ipt_arm_scan_en	ipt_arm_wrapper_se	ipt_onyx_int_scan	ipt_onyx_ext_scan	ipt_onyx_scan_en	ipt_onyx_wrapper_se	ipt_dig_rad_int_scan	ipt_dig_rad_ext_scan	ipt_dig_rad_scan_en	ipt_dig_rad_wrapper_se	ipt_sog_scan	ipt_sog_scan_en	ipt_scan_divergence	ipt_macrotest_8bit	ipt_macrotest_16bit	ipt_macrotest_mqspi
normal functional	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
arm7scan mode	1	1	0	se	se	0	0	0	0	0	0	0	0	0	0	0	0	0	0
onyx scan mode	1	0	0	0	0	1	0	se	se	0	0	0	0	0	0	0	0	0	0
dig_radio scan mode	1	0	0	0	0	0	0	0	0	1	0	se	se	0	0	0	0	0	0
sog scan mode	1	0	1	0	se	0	1	0	se	0	1	0	se	1	se	0	0	0	0
scan_divergence mode /iddq	1	0	0	se	se	0	0	se	se	0	0	se	se	1	se	1	0	0	0
macrotest mode 0	1	0	1	0	se	0	1	0	se	0	1	0	se	1	se	0	1	0	1
macrotest mode 1	1	0	1	0	se	0	1	0	se	0	1	0	se	1	se	0	0	1	1

Note: se = The Chip level scan_enable is passed through

Note: ipt_scan_mode is active high whenever a scan mode signal is active high. In LTE, this signal is replacing SCAN_TM in LT.

68.3 Scan Chain Router

The LTE scan chain router contains the logic to route the appropriate scan chains to the Chip I/O pads per each scan test mode. The router will multiplex the chain chains for each test mode, and for the scan_divergence mode will add in lock up latches in between clock domain chains. The following diagrams are a high level schematic of the router scan inputs and scan outputs.

The routing of the scan chains for different scan domains and test modes will be controlled by the scan test controller. For the scan_divergence mode the scan router should have lock up latches added in between the different scan domains, the diagram show's lockup latches between all scan domains with may not be required if they use the same scan clock. The router will also gate the scan_in unused signals to a static value for each test mode. The following tables give the scan chain connections in the router for each scan domain vs. the scan mode.

68.3.1 Scan Chains from Platforms and SOG to Be Routed

In LTE, there are 62 primary IOs allocated to form 31 scan chains at the chip level. The following are the total numbers of scan chains from each of the platform and the Sea of Gates.

- onyx platform: 27
- arm platform: 27
- dig_radio platform: 27
- sea of gates: 26 (determined by 31-5 wrapper chains since all wrappers are tested in the mode)

The platforms have scan divergence mode implemented to form one long chain. In the one chain mode, si[0], the first scan input, becomes scan divergence in, and so[26], the last scan output, becomes scan divergence out.

The SOG is different that it does not have scan divergence mode implemented since there is no wrapper to th e SOG. There will be 26 scan chains in SOG. This number is derived from the fact that there are total 31 scan chains allowed at the chip level, and 5 wrapper chains from th e 3 platforms are to be scan tested in the SOG scan mode. This leads to 31 - 5 = 26 scan chains for SOG alone. The 26 scan chains in SOG are not concatenated in scan divergence mode within the SOG. Therefore, the one scan chain formation in scan divergence mode for SOG has to be done in STC.

Table 68-4. Scan Chains from Platforms and SOG to be routed in LTE

scan domains	#scan chains	#wrapper chains
arm platform	27	2
onyx platform	27	1
dig_radio platform	27	2
sea of gates	26	(5)
max scan chains	31	

The following shows the scan chains formations in scan modes and scan divergence mode.

In **scan modes**:

pads ---> gpio_si[30:0} -- muxing{arm_chains[26:0], onyx_chains[26:0], dig_radio_chains[26:0], sog_chains[25:0]} -- gpio_so[30:0] --->pads

In **scan divergence mode**:

pad[0] ---> gpio_si[0] --->arm_si[0] ---> arm_so[26] --->onyx_si[0] ---> onyx_so[26] --->dig_radio_si[0]--->dig_radio_so[26] ---> {(sog_si[0] -- sog_so[0]) -->mux--> (sog_si[1] -- sog_so[1]) -->mux --> ... (sog_si[25] -- sog_so[25]) } ---> gpio_so[30] --->pad31

68.3.2 Scan Input Router

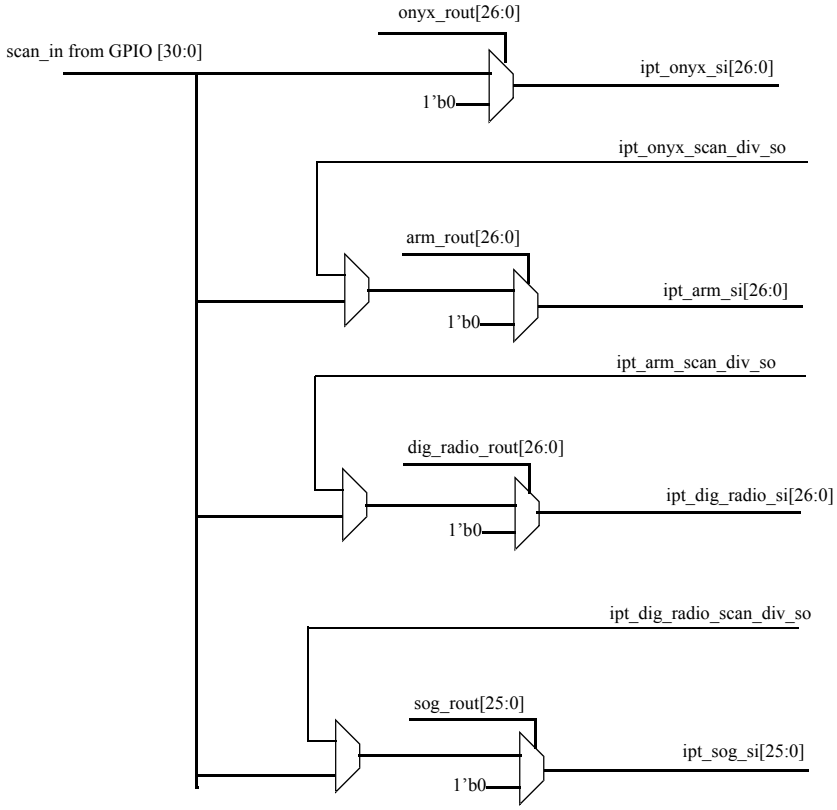


Figure 68-2. high level scan input router

68.3.3 Scan Output Router

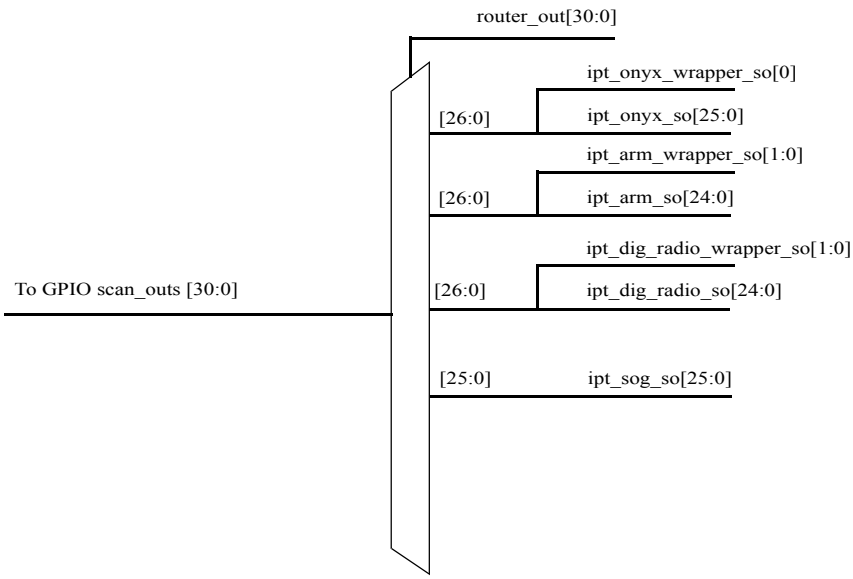


Figure 68-3. High level scan output router

Table 68-5. Top scan router ipt_arm7 scan chain connections.

STC router signals	normal functioanl mode	ARM7 scan mode	SONYX scan mode	SOG scan mode	Dig_radio Scan mode	DSP_platform scan mode	Scan Divergence mode
ipt_arm_si[0]	0	si	0	si (a)	0	0	prev_so
ipt_arm_so[0]	0	so	0	so (a)	0	0	0
ipt_arm_si[1]	0	si	0	0	0	0	0
ipt_arm_so[1]	0	so	0	0	0	0	0
ipt_arm_si[2]	0	si	0	0	0	0	0
ipt_arm_so[2]	0	so	0	0	0	0	0
ipt_arm_si[3]	0	si	0	0	0	0	0
ipt_arm_so[3]	0	so	0	0	0	0	0
ipt_arm_si[4]	0	si	0	0	0	0	0
ipt_arm_so[4]	0	so	0	0	0	0	0
ipt_arm_si[5]	0	si	0	0	0	0	0
ipt_arm_so[5]	0	so	0	0	0	0	0
ipt_arm_si[6]	0	si	0	0	0	0	0
ipt_arm_so[6]	0	so	0	0	0	0	0
ipt_arm_si[7]	0	si	0	0	0	0	0
ipt_arm_so[7]	0	so	0	0	0	0	0
ipt_arm_si[8]	0	si	0	0	0	0	0
ipt_arm_so[8]	0	so	0	0	0	0	0
ipt_arm_si[9]	0	si	0	0	0	0	0
ipt_arm_so[9]	0	so	0	0	0	0	0
ipt_arm_si[10]	0	si	0	0	0	0	0
ipt_arm_so[10]	0	so	0	0	0	0	0
ipt_arm_si[11]	0	si	0	0	0	0	0
ipt_arm_so[11]	0	so	0	0	0	0	0
ipt_arm_si[12]	0	si	0	0	0	0	0
ipt_arm_so[12]	0	so	0	0	0	0	0
ipt_arm_si[13]	0	si	0	0	0	0	0
ipt_arm_so[13]	0	so	0	0	0	0	0
ipt_arm_si[14]	0	si	0	0	0	0	0
ipt_arm_so[14]	0	so	0	0	0	0	0
ipt_arm_si[15]	0	si	0	0	0	0	0
ipt_arm_so[15]	0	so	0	0	0	0	0

STC router signals	normal functioanl mode	ARM7 scan mode	SONYX scan mode	SOG scan mode	Dig_radio Scan mode	DSP_platform scan mode	Scan Divergence mode
ipt_arm_si[16]	0	si	0	0	0	0	0
ipt_arm_so[16]	0	so	0	0	0	0	0
ipt_arm_si[17]	0	si	0	0	0	0	0
ipt_arm_so[17]	0	so	0	0	0	0	0
ipt_arm_si[18]	0	si	0	0	0	0	0
ipt_arm_so[18]	0	so	0	0	0	0	0
ipt_arm_si[19]	0	si	0	0	0	0	0
ipt_arm_so[19]	0	so	0	0	0	0	0
ipt_arm_si[20]	0	si	0	0	0	0	0
ipt_arm_so[20]	0	so	0	0	0	0	0
ipt_arm_si[21]	0	si	0	0	0	0	0
ipt_arm_so[21]	0	so	0	0	0	0	0
ipt_arm_si[22]	0	si	0	0	0	0	0
ipt_arm_so[22]	0	so	0	0	0	0	0
ipt_arm_si[23]	0	si	0	0	0	0	0
ipt_arm_so[23]	0	so	0	0	0	0	0
ipt_arm_si[24]	0	si	0	0	0	0	0
ipt_arm_so[24]	0	so	0	0	0	0	0
ipt_arm_si[25]	0	si	0	0	0	0	0
ipt_arm_so[25]	0	so	0	0	0	0	0
ipt_arm_si[26]	0	si	0	0	0	0	0
ipt_arm_so[26]	0	so	0	0	0	0	next_si

Note: (a) = arm7 wrapper is appened to one chain in this mode.

Note: si = scan chain connected to pad si via GPIO

Note: so = scan chain connected to pad so via GPIO

Note: prev_so / next_si = scan chains appended during scan divergence

Table 68-6. Top scan router ipt_onyx scan chain connections.

STC router signals	normal functioanl mode	ARM7 scan mode	SONYX scan mode	SOG scan mode	Dig_radio Scan mode	DSP_platform scan mode	Scan Divergence mode
ipt_onyx_si[0]	0	0	si	si (a)	0	0	prev_so
ipt_onyx_so[0]	0	0	so	so (a)	0	0	0
ipt_onyx_si[1]	0	0	si	0	0	0	0
ipt_onyx_so[1]	0	0	so	0	0	0	0
ipt_onyx_si[2]	0	0	si	0	0	0	0
ipt_onyx_so[2]	0	0	so	0	0	0	0
ipt_onyx_si[3]	0	0	si	0	0	0	0
ipt_onyx_so[3]	0	0	so	0	0	0	0
ipt_onyx_si[4]	0	0	si	0	0	0	0
ipt_onyx_so[4]	0	0	so	0	0	0	0
ipt_onyx_si[5]	0	0	si	0	0	0	0
ipt_onyx_so[5]	0	0	so	0	0	0	0
ipt_onyx_si[6]	0	0	si	0	0	0	0
ipt_onyx_so[6]	0	0	so	0	0	0	0
ipt_onyx_si[7]	0	0	si	0	0	0	0
ipt_onyx_so[7]	0	0	so	0	0	0	0
ipt_onyx_si[8]	0	0	si	0	0	0	0
ipt_onyx_so[8]	0	0	so	0	0	0	0
ipt_onyx_si[9]	0	0	si	0	0	0	0
ipt_onyx_so[9]	0	0	so	0	0	0	0
ipt_onyx_si[10]	0	0	si	0	0	0	0
ipt_onyx_so[10]	0	0	so	0	0	0	0
ipt_onyx_si[11]	0	0	si	0	0	0	0
ipt_onyx_so[11]	0	0	so	0	0	0	0
ipt_onyx_si[12]	0	0	si	0	0	0	0
ipt_onyx_so[12]	0	0	so	0	0	0	0
ipt_onyx_si[13]	0	0	si	0	0	0	0
ipt_onyx_so[13]	0	0	so	0	0	0	0
ipt_onyx_si[14]	0	0	si	0	0	0	0
ipt_onyx_so[14]	0	0	so	0	0	0	0
ipt_onyx_si[15]	0	0	si	0	0	0	0
ipt_onyx_so[15]	0	0	so	0	0	0	0

STC router signals	normal functioanl mode	ARM7 scan mode	SONYX scan mode	SOG scan mode	Dig_radio Scan mode	DSP_platform scan mode	Scan Divergence mode
ipt_onyx_si[16]	0	0	si	0	0	0	0
ipt_onyx_so[16]	0	0	so	0	0	0	0
ipt_onyx_si[17]	0	0	si	0	0	0	0
ipt_onyx_so[17]	0	0	so	0	0	0	0
ipt_onyx_si[18]	0	0	si	0	0	0	0
ipt_onyx_so[18]	0	0	so	0	0	0	0
ipt_onyx_si[19]	0	0	si	0	0	0	0
ipt_onyx_so[19]	0	0	so	0	0	0	0
ipt_onyx_si[20]	0	0	si	0	0	0	0
ipt_onyx_so[20]	0	0	so	0	0	0	0
ipt_onyx_si[21]	0	0	si	0	0	0	0
ipt_onyx_so[21]	0	0	so	0	0	0	0
ipt_onyx_si[22]	0	0	si	0	0	0	0
ipt_onyx_so[22]	0	0	so	0	0	0	0
ipt_onyx_si[23]	0	0	si	0	0	0	0
ipt_onyx_so[23]	0	0	so	0	0	0	0
ipt_onyx_si[24]	0	0	si	0	0	0	0
ipt_onyx_so[24]	0	0	so	0	0	0	0
ipt_onyx_si[25]	0	0	si	0	0	0	0
ipt_onyx_so[25]	0	0	so	0	0	0	0
ipt_onyx_si[26]	0	0	si	0	0	0	0
ipt_onyx_so[26]	0	0	so	0	0	0	next_si

Note: (a) = onyx wrapper is appened to one chain in this mode.

Note: si = scan chain connected to pad si via GPIO

Note: so = scan chain connected to pad so via GPIO

Note: prev_so / next_si = scan chains appended during scan divergence

Table 68-7. Top scan router ipt_sog scan chain connections.

STC router signals	normal functioanl mode	ARM7 scan mode	SONYX scan mode	SOG scan mode	Dig_radio Scan mode	DSP_platform scan mode	Scan Divergence mode
ipt_sog_si[0]	0	0	0	si	0	0	prev_so
ipt_sog_so[0]	0	0	0	so	0	0	0
ipt_sog_si[1]	0	0	0	si	0	0	0
ipt_sog_so[1]	0	0	0	so	0	0	0
ipt_sog_si[2]	0	0	0	si	0	0	0
ipt_sog_so[2]	0	0	0	so	0	0	0
ipt_sog_si[3]	0	0	0	si	0	0	0
ipt_sog_so[3]	0	0	0	so	0	0	0
ipt_sog_si[4]	0	0	0	si	0	0	0
ipt_sog_so[4]	0	0	0	so	0	0	0
ipt_sog_si[5]	0	0	0	si	0	0	0
ipt_sog_so[5]	0	0	0	so	0	0	0
ipt_sog_si[6]	0	0	0	si	0	0	0
ipt_sog_so[6]	0	0	0	so	0	0	0
ipt_sog_si[7]	0	0	0	si	0	0	0
ipt_sog_so[7]	0	0	0	so	0	0	0
ipt_sog_si[8]	0	0	0	si	0	0	0
ipt_sog_so[8]	0	0	0	so	0	0	0
ipt_sog_si[9]	0	0	0	si	0	0	0
ipt_sog_so[9]	0	0	0	so	0	0	0
ipt_sog_si[10]	0	0	0	si	0	0	0
ipt_sog_so[10]	0	0	0	so	0	0	0
ipt_sog_si[11]	0	0	0	si	0	0	0
ipt_sog_so[11]	0	0	0	so	0	0	0
ipt_sog_si[12]	0	0	0	si	0	0	0
ipt_sog_so[12]	0	0	0	so	0	0	0
ipt_sog_si[13]	0	0	0	si	0	0	0
ipt_sog_so[13]	0	0	0	so	0	0	0
ipt_sog_si[14]	0	0	0	si	0	0	0
ipt_sog_so[14]	0	0	0	so	0	0	0
ipt_sog_si[15]	0	0	0	si	0	0	0
ipt_sog_so[15]	0	0	0	so	0	0	0

STC router signals	normal functioanl mode	ARM7 scan mode	SONYX scan mode	SOG scan mode	Dig_radio Scan mode	DSP_platform scan mode	Scan Divergence mode
ipt_sog_si[16]	0	0	0	si	si	0	0
ipt_sog_so[16]	0	0	0	so	so	0	0
ipt_sog_si[17]	0	0	0	si	si	0	0
ipt_sog_so[17]	0	0	0	so	so	0	0
ipt_sog_si[18]	0	0	0	si	si	0	0
ipt_sog_so[18]	0	0	0	so	so	0	0
ipt_sog_si[19]	0	0	0	si	si	0	0
ipt_sog_so[19]	0	0	0	so	so	0	0
ipt_sog_si[20]	0	0	0	si	si	0	0
ipt_sog_so[20]	0	0	0	so	so	0	0
ipt_sog_si[21]	0	0	0	si	si	0	0
ipt_sog_so[21]	0	0	0	so	so	0	0
ipt_sog_si[22]	0	0	0	si	si	0	0
ipt_sog_so[22]	0	0	0	so	so	0	0
ipt_sog_si[23]	0	0	0	si	si	0	0
ipt_sog_so[23]	0	0	0	so	so	0	0
ipt_sog_si[24]	0	0	0	si	si	0	0
ipt_sog_so[24]	0	0	0	so	so	0	0
ipt_sog_si[25]	0	0	0	si	si	0	0
ipt_sog_so[25]	0	0	0	so	so	0	next_si
ipt_sog_si[26]	-	-	-	-	-	-	-
ipt_sog_so[26]	-	-	-	-	-	-	-
ipt_sog_si[27]	-	-	-	-	-	-	-
ipt_sog_so[27]	-	-	-	-	-	-	-
ipt_sog_si[28]	-	-	-	-	-	-	-
ipt_sog_so[28]	-	-	-	-	-	-	-
ipt_sog_si[29]	-	-	-	-	-	-	-
ipt_sog_so[29]	-	-	-	-	-	-	-
ipt_sog_si[30]	-	-	-	-	-	-	-
ipt_sog_so[30]	-	-	-	-	-	-	-

Note: si = scan chain connected to pad si via GPIO

Note: so = scan chain connected to pad so via GPIO

Note: prev_so / next_si = scan chains appended during scan divergence

Note: - = These scan chains will be used by platform wrapper chains.

Table 68-8. Top scan router ipt_dig_radio scan chain connections.

STC router signals	normal functioanl mode	ARM7 scan mode	SONYX scan mode	SOG scan mode	Dig_radio Scan mode	DSP_platform scan mode	Scan Divergence mode
ipt_dig_radio_si[0]	0	0	0	si (a)	si	0	prev_so
ipt_dig_radio_so[0]	0	0	0	so (a)	so	0	0
ipt_dig_radio_si[1]	0	0	0	0	si	0	0
ipt_dig_radio_so[1]	0	0	0	0	so	0	0
ipt_dig_radio_si[2]	0	0	0	0	si	0	0
ipt_dig_radio_so[2]	0	0	0	0	so	0	0
ipt_dig_radio_si[3]	0	0	0	0	si	0	0
ipt_dig_radio_so[3]	0	0	0	0	so	0	0
ipt_dig_radio_si[4]	0	0	0	0	si	0	0
ipt_dig_radio_so[4]	0	0	0	0	so	0	0
ipt_dig_radio_si[5]	0	0	0	0	si	0	0
ipt_dig_radio_so[5]	0	0	0	0	so	0	0
ipt_dig_radio_si[6]	0	0	0	0	si	0	0
ipt_dig_radio_so[6]	0	0	0	0	so	0	0
ipt_dig_radio_si[7]	0	0	0	0	si	0	0
ipt_dig_radio_so[7]	0	0	0	0	so	0	0
ipt_dig_radio_si[8]	0	0	0	0	si	0	0
ipt_dig_radio_so[8]	0	0	0	0	so	0	0
ipt_dig_radio_si[9]	0	0	0	0	si	0	0
ipt_dig_radio_so[9]	0	0	0	0	so	0	0
ipt_dig_radio_si[10]	0	0	0	0	si	0	0
ipt_dig_radio_so[10]	0	0	0	0	so	0	0
ipt_dig_radio_si[11]	0	0	0	0	si	0	0
ipt_dig_radio_so[11]	0	0	0	0	so	0	0
ipt_dig_radio_si[12]	0	0	0	0	si	0	0
ipt_dig_radio_so[12]	0	0	0	0	so	0	0
ipt_dig_radio_si[13]	0	0	0	0	si	0	0
ipt_dig_radio_so[13]	0	0	0	0	so	0	0
ipt_dig_radio_si[14]	0	0	0	0	si	0	0
ipt_dig_radio_so[14]	0	0	0	0	so	0	0
ipt_dig_radio_si[15]	0	0	0	0	si	0	0
ipt_dig_radio_so[15]	0	0	0	0	so	0	0

STC router signals	normal functioanl mode	ARM7 scan mode	SONYX scan mode	SOG scan mode	Dig_radio Scan mode	DSP_platform scan mode	Scan Divergence mode
ipt_dig_radio_si[16]	0	0	0	0	si	0	0
ipt_dig_radio_so[16]	0	0	0	0	so	0	0
ipt_dig_radio_si[17]	0	0	0	0	si	0	0
ipt_dig_radio_so[17]	0	0	0	0	so	0	0
ipt_dig_radio_si[18]	0	0	0	0	si	0	0
ipt_dig_radio_so[18]	0	0	0	0	so	0	0
ipt_dig_radio_si[19]	0	0	0	0	si	0	0
ipt_dig_radio_so[19]	0	0	0	0	so	0	0
ipt_dig_radio_si[20]	0	0	0	0	si	0	0
ipt_dig_radio_so[20]	0	0	0	0	so	0	0
ipt_dig_radio_si[21]	0	0	0	0	si	0	0
ipt_dig_radio_so[21]	0	0	0	0	so	0	0
ipt_dig_radio_si[22]	0	0	0	0	si	0	0
ipt_dig_radio_so[22]	0	0	0	0	so	0	0
ipt_dig_radio_si[23]	0	0	0	0	si	0	0
ipt_dig_radio_so[23]	0	0	0	0	so	0	0
ipt_dig_radio_si[24]	0	0	0	0	si	0	0
ipt_dig_radio_so[24]	0	0	0	0	so	0	0
ipt_dig_radio_si[25]	0	0	0	0	si	0	0
ipt_dig_radio_so[25]	0	0	0	0	so	0	0
ipt_dig_radio_si[26]	0	0	0	0	si	0	0
ipt_dig_radio_so[26]	0	0	0	0	so	0	next_si

Note: (a) = onyx wrapper is appened to one chain in this mode.

Note: si = scan chain connected to pad si via GPIO

Note: so = scan chain connected to pad so via GPIO

Note: prev_so / next_si = scan chains appended during scan divergence

68.4 Scan Reset Logic Additions

STC provides the reset muxing for scan mode on two reset signal from SONYXU platform. The two reset signals are “res” and “pires”. They are muxed with ccm_reset_b from CCM. In scan modes, the reset signals become the “ccm_reset_b”. The outputs of the reset signals are named as “stc_res” and “stc_pires”.

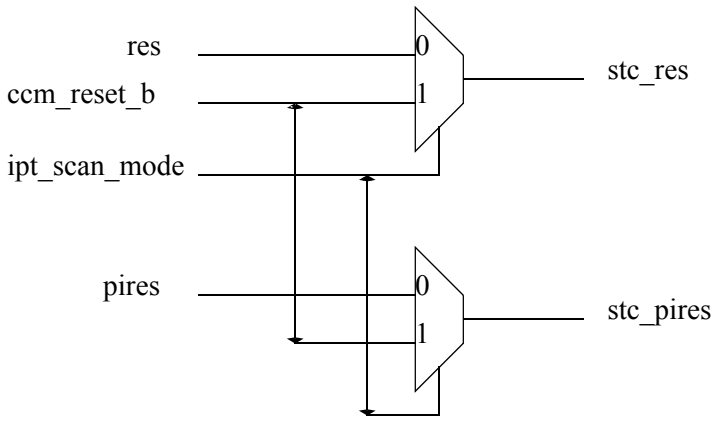


Figure 68-4. Reset signals's muxing for scan modes

68.5 STC Module Pin List

Table 68-9 is a listing of all the pins in the Scan Test Controller Module.

Table 68-9. STC Module Pin List

Pin Name	Direction	Description
Primary Inputs		
test0	INPUT	Primary input, TEST0, test mode configuration pin
test1	INPUT	Primary input, TEST1, test mode configuration pin
test2	INPUT	Primary input, TEST2, test mode configuration pin
mcu_de	INPUT	Primary input, MCU_DE, test mode configuration pin
dsp_de	INPUT	Primary input, DSP_DE, test mode configuration pin
scka	INPUT	Primary input, SCKA, master mode configuration pin
scan_en	INPUT	Primary input, chip scan enable
GPIO		
gpio_si[30:0]	INPUT	scan data inputs via From GPIO, Muxed from external Pads
gpio_so[30:0]	OUTPUT	scan data outputs via To GPIO to be mux'd to external pads.
From Scan Outputs of Different Scan Domains		
ipt_onyx_so[26:0]	INPUT	scan data outputs From dsp platform
ipt_arm_so[26:0]	INPUT	scan data outputs From arm7 platform

Table 68-9. STC Module Pin List

Pin Name	Direction	Description
ipt_dig_radio_so[26:0]	INPUT	scan data outputs From dig_radio platform
ipt_sog_so[25:0]	INPUT	scan data outputs From sog
Scan Control Signals		
ipt_scan_mode	OUTPUT	Chip level scan_mode signal
ipt_arm_int_scan	OUTPUT	arm platform internal scan test mode
ipt_arm_ext_scan	OUTPUT	arm platform external scan test mode
ipt_arm_scan_en	OUTPUT	arm platform scan enable
ipt_arm_wrapper_se	OUTPUT	arm platform wrapper scan enable
ipt_onyx_int_scan	OUTPUT	onyx platform internal scan test mode
ipt_onyx_ext_scan	OUTPUT	onyx platform external scan test mode
ipt_onyx_scan_en	OUTPUT	onyx platform scan enable
ipt_onyx_wrapper_se	OUTPUT	onyx platform wrapper scan enable
ipt_dig_radio_int_scan	OUTPUT	dig_radio platform internal scan test mode
ipt_dig_radio_ext_scan	OUTPUT	dig_radio platform external scan test mode
ipt_dig_radio_scan_en	OUTPUT	dig_radio platform scan enable
ipt_dig_radio_wrapper_se	OUTPUT	dig_radio platform wrapper scan enable
ipt_sog_scan	OUTPUT	SOG scan test mode
ipt_sog_scan_en	OUTPUT	SOG scan enable
ipt_scan_divergence	OUTPUT	scan divergence test mode
ipt_macrotest_8bit	OUTPUT	macrotest mode for 8-bit memory in L1T
ipt_macrotest_16bit	OUTPUT	macrotest mode for 16-bit memory in L1T
ipt_macrotest_mqspi	OUTPUT	macrotest mode for x-bit memory in MQSPI
To Scan Inputs of Different Scan Domains		
ipt_onyx_si[26:0]	OUTPUT	scan data inputs To Onyx Platform
ipt_arm_si[26:0]	OUTPUT	scan data inputs To arm7 platform
ipt_dig_radio_si[26:0]	OUTPUT	scan data inputs To dig_radio platform
ipt_sog_si[25:0]	OUTPUT	scan data inputs To SOG
Reset Signals		
res	INPUT	reset signal from sonyxu

Table 68-9. STC Module Pin List

Pin Name	Direction	Description
pires	INPUT	reset signal from sonyxu
ccm_reset_b	INPUT	reset signal from ccm
stc_res	OUTPUT	muxed version of res with ccm_reset_b for scan mode - to sonyx and dsp platforms
stc_pires	OUTPUT	muxed version of pires with ccm_reset_b for scan mode - to sonyx and dsp platforms

Chapter 69
MCU External Interrupts (INT)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	03/19/02	Naveen Mishra	Interface changes from MIG/PIG to IPBUS. Bi-direction pins to GPIO changed to Uni-directions. Removed Output enables (intr_ext_intr_oe) for interrupts pin output buffer.
0.2	04/18/02	Naveen Mishra	Signals follow the recommended naming convention.Data synchronizer added for read_data bus.
0.3	05/21/02	Naveen Mishra	Corrected Address for Registers.Added description for xfr_wait signal assertion.
0.4	6/19/02	Shannon Osgood	Entered base address variables in register tables.
0.5	05/07/03		Updated for LTE specification release.

69.1 Block Diagram

This module provides control for eight external interrupt sources.

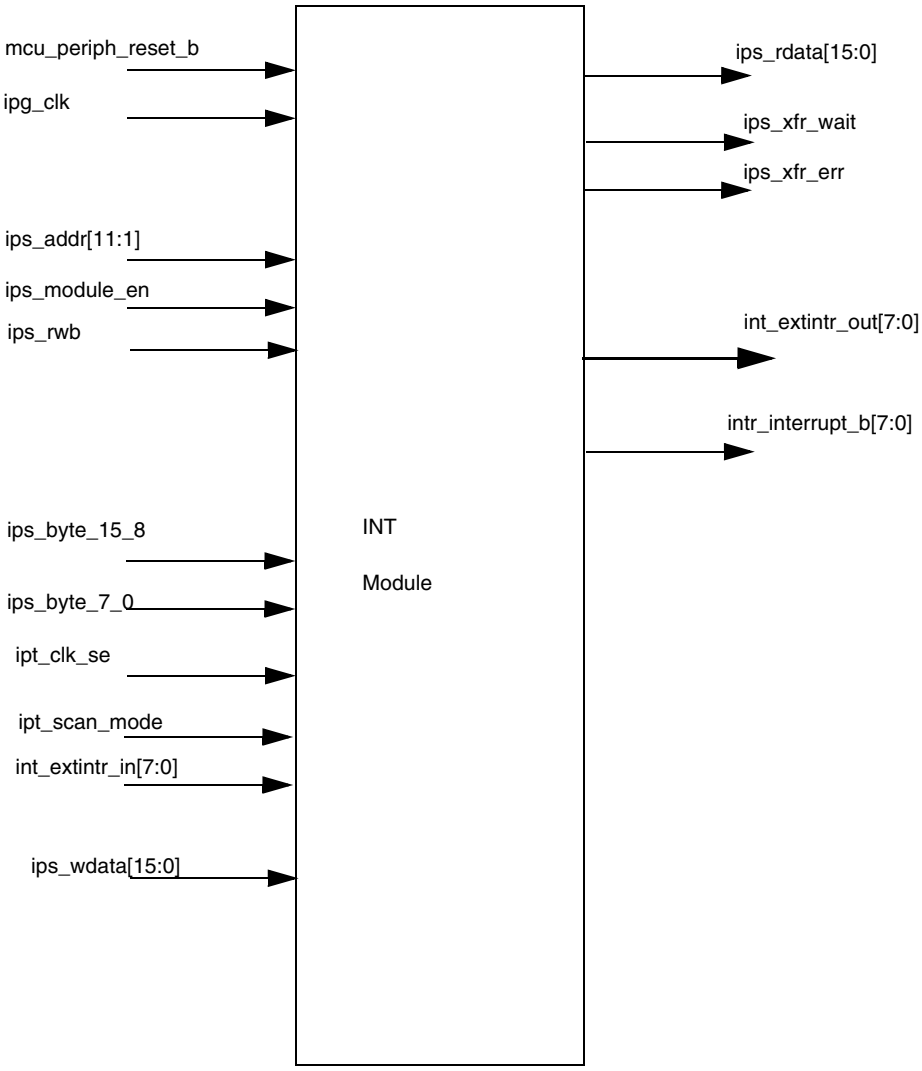


Figure 69-1. Block diagram of INT module

69.2 Ports List

Table 69-1 lists all the pins in the INT module.

Table 69-1. INT Pin List

Pin Name	Direction	Description	SNO
PIG Pins			
ips_module_en	Input	module select	1
ips_byte_15_8	Input	module byte write enable	2
ips_byte_7_0	Input	module byte write enable (LSB)	3
ipg_rwb	Input	1 - read, 0 - write	4
ips_addr[11:1]	Input	address bus	5
ips_wdata[15:0]	Input	write data bus	6
ips_rdata[15:0]	output	read data bus	7
ips_xfr_wait	Output	transfer acknowledge signal	8
ips_xfr_err	Output	transfer error acknowledge signal	9
Clock and Interrupt Pins			
mcu_periph_reset_b	Input	MCU peripheral module reset	1
ipg_clk	Input	System Clock (MCU or Test)	2
int_exintr_in[7:0]	Input	Interrupts pins Input	3
int_extintr_out[7:0]	IOutput	Output data pins	4
intr_interrupt_b[7:0]	Output	Interrupts to ITC	5
Scan Pins			
ipt_scan_mode	Input	scan mode	1
ipt_clk_se	Input	Scan enable	2

69.3 Pins Description

The Neptune LTE has eight interrupt pins. Each pin is individually configurable as a level-sensitive interrupt, an edge-detecting (rising, falling, or both) interrupt, or a general purpose input. In Neptune LTE each pin has a dedicated interrupt line.

INT0-INT7 Some of these are internally pulled-up and some are internally pulled-down. All default to general purpose input pins at reset.

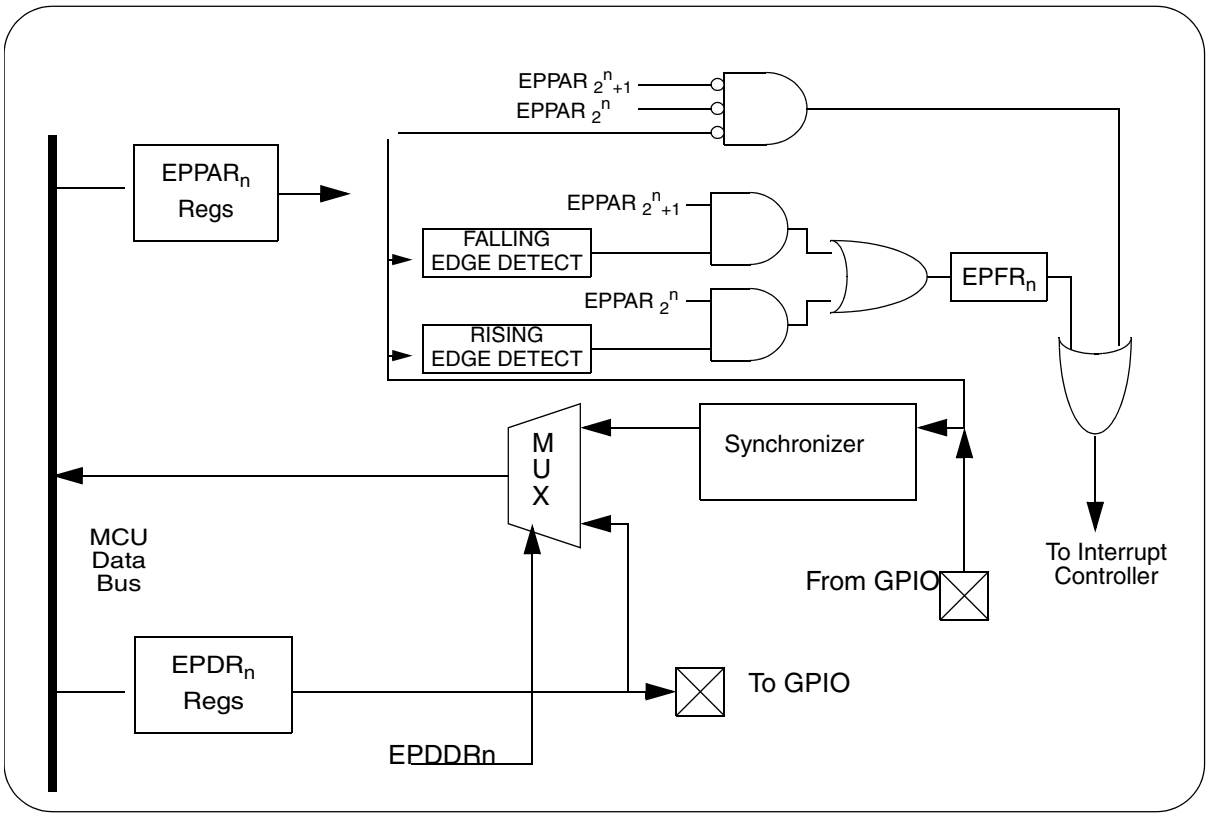


Figure 69-2. External Interrupt / GPIO Block Diagram (each pin)

NOTE:

In Neptune LTE there is no output functionality provided for INT pins. So the I/O from GPIO is always used as input and the EPDR register remains unused (See EPDR Description).

69.4 Programming Model

The programming model consists of four registers: the Edge Port Pin Assignment Register (EPPAR) which controls the function of each one of the pins individually, the Edge Port Data Direction Register (EPDDR) which controls the direction of each one of the pins individually, the Edge Port Data Register (EPDR) which holds the data in an internal latch, and the Edge Port Flag Register (EPFR) which latches the edge event for each one of the pins individually.

69.4.1 Register Summary

KEY: Always Reads One (1) | Always Reads Zero (0) | Read/Write Bit (bit) | Read-Only Bit (bit) | Write-Only Bit (bit) | Write 1 to Clear (bit, *W1C*) | Self-Clear Bit (0, bit) | N/A

Base_Address = $2484_C000

Absolute_ Address = Base_Address + Offset

Table 69-2. MCU External Interrupts Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPPAR $2484_C000	R	EPPAR7[1:0]		EPPAR6[1:0]		EPPAR5[1:0]		EPPAR4[1:0]		EPPAR3[1:0]		EPPAR2[1:0]		EPPAR1[1:0]		EPPAR0[1:0]	
	W																
EPDDR $2484_C002	R	0	0	0	0	0	0	0	0	EPDD7	EPDD6	EPDD5	EPDD4	EPDD3	EPDD2	EPDD1	EPDD0
	W																
EPDR $2484_C004	R	0	0	0	0	0	0	0	0	EPD7	EPD6	EPD5	EPD4	EPD3	EPD2	EPD1	EPD0
	W																
EPFR $2484_C006	R	0	0	0	0	0	0	0	0	EPF7	EPF6	EPF5	EPF4	EPF3	EPF2	EPF1	EPF0
	W																

69.4.2 Detailed Register Descriptions

The following figures and associated text give detailed descriptions of the various MCU External Interrupt registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

EPPAR Edge Port Pin Assignment Register $2484_C000

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
	EPPAR7[1:0]		EPPAR6[1:0]		EPPAR5[1:0]		EPPAR4[1:0]		EPPAR3[1:0]		EPPAR2[1:0]		EPPAR1[1:0]		EPPAR0[1:0]	
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 69-3. EPPAR Description

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>EPPAR7 [1:0]
Bits 15-14</td><td>Edge Port Pin 7 Assignment — configures interrupt pin 7 as either level sensitive, or edge-triggered</td><td>
<table>
<tr><th>EPPAR7 [1:0]</th><th>INT7 Assignment</th></tr>
<tr><td>00</td><td>Pin INT7 defined as level sensitive.</td></tr>
<tr><td>01</td><td>Pin INT7 defined as rising edge detect.</td></tr>
<tr><td>10</td><td>Pin INT7 defined as falling edge detect.</td></tr>
<tr><td>11</td><td>Pin INT7 defined as both falling and rising edge detect.</td></tr>
</table>
</td></tr>
<tr><td>EPPAR6 [1:0]
Bits 13-12</td><td>Edge Port Pin 6 Assignment — configures interrupt pin 6 as either level sensitive, or edge-triggered</td><td>
<table>
<tr><th>EPPAR6 [1:0]</th><th>INT6 Assignment</th></tr>
<tr><td>00</td><td>Pin INT6 defined as level sensitive.</td></tr>
<tr><td>01</td><td>Pin INT6 defined as rising edge detect.</td></tr>
<tr><td>10</td><td>Pin INT6 defined as falling edge detect.</td></tr>
<tr><td>11</td><td>Pin INT6 defined as both falling and rising edge detect.</td></tr>
</table>
</td></tr>
</table>

Table 69-3. EPPAR Description

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>EPPAR5 [1:0]
Bits 11-10</td><td>Edge Port Pin 5 Assignment — configures interrupt pin 5 as either level sensitive, or edge-triggered</td><td>
<table>
<tr><th>EPPAR5 [1:0]</th><th>INT5 Assignment</th></tr>
<tr><td>00</td><td>Pin INT5 defined as level sensitive.</td></tr>
<tr><td>01</td><td>Pin INT5 defined as rising edge detect.</td></tr>
<tr><td>10</td><td>Pin INT5 defined as falling edge detect.</td></tr>
<tr><td>11</td><td>Pin INT5 defined as both falling and rising edge detect.</td></tr>
</table>
</td></tr>
<tr><td>EPPAR4 [1:0]
Bits 9-8</td><td>Edge Port Pin 4 Assignment — configures interrupt pin 4 as either level sensitive, or edge-triggered</td><td>
<table>
<tr><th>EPPAR4 [1:0]</th><th>INT4 Assignment</th></tr>
<tr><td>00</td><td>Pin INT4 defined as level sensitive.</td></tr>
<tr><td>01</td><td>Pin INT4 defined as rising edge detect.</td></tr>
<tr><td>10</td><td>Pin INT4 defined as falling edge detect.</td></tr>
<tr><td>11</td><td>Pin INT4 defined as both falling and rising edge detect.</td></tr>
</table>
</td></tr>
<tr><td>EPPAR3 [1:0]
Bits 7-6</td><td>Edge Port Pin 3 Assignment — configures interrupt pin 3 as either level sensitive, or edge-triggered</td><td>
<table>
<tr><th>EPPAR3 [1:0]</th><th>INT3 Assignment</th></tr>
<tr><td>00</td><td>Pin INT3 defined as level sensitive.</td></tr>
<tr><td>01</td><td>Pin INT3 defined as rising edge detect.</td></tr>
<tr><td>10</td><td>Pin INT3 defined as falling edge detect.</td></tr>
<tr><td>11</td><td>Pin INT3 defined as both falling and rising edge detect.</td></tr>
</table>
</td></tr>
</table>

Table 69-3. EPPAR Description

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td>EPPAR2 [1:0]
Bits 5-4</td><td>Edge Port Pin 2 Assignment — configures interrupt pin 2 as either level sensitive, or edge-triggered</td><td>
<table>
<tr><th>EPPAR2 [1:0]</th><th>INT2 Assignment</th></tr>
<tr><td>00</td><td>Pin INT2 defined as level sensitive.</td></tr>
<tr><td>01</td><td>Pin INT2 defined as rising edge detect.</td></tr>
<tr><td>10</td><td>Pin INT2 defined as falling edge detect.</td></tr>
<tr><td>11</td><td>Pin INT2 defined as both falling and rising edge detect.</td></tr>
</table>
</td></tr>
<tr><td>EPPAR1 [1:0]
Bits 3-2</td><td>Edge Port Pin 1 Assignment — configures interrupt pin 1 as either level sensitive, or edge-triggered</td><td>
<table>
<tr><th>EPPAR1 [1:0]</th><th>INT1 Assignment</th></tr>
<tr><td>00</td><td>Pin INT1 defined as level sensitive.</td></tr>
<tr><td>01</td><td>Pin INT1 defined as rising edge detect.</td></tr>
<tr><td>10</td><td>Pin INT1 defined as falling edge detect.</td></tr>
<tr><td>11</td><td>Pin INT1 defined as both falling and rising edge detect.</td></tr>
</table>
</td></tr>
<tr><td>EPPAR0 [1:0]
Bits 1-0</td><td>Edge Port Pin 0 Assignment — configures interrupt pin 0 as either level sensitive, or edge-triggered</td><td>
<table>
<tr><th>EPPAR0 [1:0]</th><th>INT0 Assignment</th></tr>
<tr><td>00</td><td>Pin INT0 defined as level sensitive.</td></tr>
<tr><td>01</td><td>Pin INT0 defined as rising edge detect.</td></tr>
<tr><td>10</td><td>Pin INT0 defined as falling edge detect.</td></tr>
<tr><td>11</td><td>Pin INT0 defined as both falling and rising edge detect.</td></tr>
</table>
</td></tr>
</table>

The 16-bit read/write Edge Port Pin Assignment Register (EPPAR) configures each one of the interrupt pins as either level sensitive, or edge-triggered. The EPPAR bits are described in Table 69-3.

The EPPAR register controls the characteristics of the logic between the pin and the interrupt controller. Level sensitive or edge-detecting modes may be selected. Rising, falling, or both edges can be selected as the active edge when programmed as an edge-detecting source. Requests are always generated out of this block, but may be masked within the MCU interrupt controller. The functionality of this register is independent of the programmed pin direction.

Pins configured as level-sensitive are inverted so that a logic low on the external pin represents a valid interrupt request. Level sensitive interrupt inputs are not latched. To guarantee that a level-sensitive interrupt request is acknowledged the interrupt source must keep the signal asserted until acknowledged by software.

Pins configured as edge-sensitive interrupts are latched and need not remain asserted. When programmed to use the edge detecting circuit, the state of the I/O pins is monitored regardless of the configuration as input or output.

EPDDR Edge Port Data Direction Register $2484_C002

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
									EPDD 7	EPDD 6	EPDD 5	EPDD 4	EPDD 3	EPDD 2	EPDD 1	EPDD 0
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 69-4. EPDDR Description

Name	Description	Settings
Bits 15-8	**Reserved —** These bits are read as zero. These bits should be written as zeros for future compatibility.	N/A
EPDD[7:0] Bits 7-0	**Edge Port Data Direction —** The 16-bit read/write Edge Port Data Direction Register (EPDDR) controls the read data from port pins. Setting any bit in this register reads back the corresponding bit from EPDR register.Clearing any bit in this register reads the corresponding input port pin. **Note:** For Neptune devices, the INT[7:0] pins are muxed through the GPIO module. To program an INT pin as an input, the appropriate alternate input function must be selected in the GPIO module, as well as the appropriate bit in EPDDR. The INT[7:0] pins are not assigned an alternate output function in the GPIO module, so programming a bit in EPDDR as an output has no effect on the external pin. Refer to Chapter 2 for the GPIO alternate input and output assignments.	0 = EPDD***n*** data read from corresponding input port pins. 1 = EPDD***n*** Data is read from corresponding EPDR bit.

EPDR Edge Port Data Register $2484_C004

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
									EPD7	EPD6	EPD5	EPD4	EPD3	EPD2	EPD1	EPD0
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	??	??	??	??	??	??	??	??

Table 69-5. EPDR Description

Name	Description	Settings
Bits 15-8	**Reserved —** These bits are read as zero. These bits should be written as zeros for future compatibility.	N/A
EPD[7:0] Bits 7-0	**Edge Port Data —** Writes to EPDR are stored in an internal latch. Reads of this register return the value sensed on the pins for those pins configured as inputs, or the data stored in the register for the pins configured as outputs. **Note:** For Neptune devices, the INT[7:0] pins are muxed through the GPIO module. The INT[7:0] pins are not assigned an alternate output function in the GPIO module, so if any pin of the port is configured as an output, the data stored for that pin in EPDR is not driven on the pin. EPDR can be considered a read only register. Refer to Chapter 2 for the GPIO alternate input and output assignments. **Note:** The default value of EPDR register depends upon the state of pins at the chip boundary and GPIO configuration for the corresponding bit. **Note:** The assertion of logic level on the chip boundary is Asynchronous in nature, therefore the inputs when reaches module boundary are synchronised before the module reads the values. BUT the interrupt generation logic uses the unsynchronized version of the signal comming from the chip boundary as the interrupts needs to be recognised even when the chip is in low power mode.Because of the synchronizer in place, now the read data bus will require at max two clock cycle before reflecting the actual data from the chip boundary.	N/A

NOTE: Since the asynchronous data from the chip boundary reaches the read_bus with a delay of two mcu clock cycles(because of the insertion of two stage flop synchronizer) , therefore whenever an access is done to EPDR register for reading the data, it generates a transfer wait signal (2 ipg clock cycle) for the mcu. The assertion of xfr_wait results in two additional clocks cycle wait before mcu finally reads the data, which is the mirror image of the port signals.

EPFR Edge Port Flag Register $2484_C**006**

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT0
									EPF7	EPF6	EPF5	EPF4	EPF3	EPF2	EPF1	EPF0
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 69-6. EPFR Description

Name	Description	Settings
Bits 15-8	**Reserved —** These bits are read as zero. These bits should be written as zeros for future compatibility.	N/A
EPF[7:0] Bits 7-0	**Edge Port Flag —** The 16-bit read/write Edge Port Flag Register (EPFR) indicates whether the selected edge has been detected on the port pins. Bits in this register are set when the programmed edge is detected on the corresponding pin. The bits remain set until cleared by writing a “1” into the bit position after reading the bit set. Pin transitions do not affect this register if the pin is configured as level sensitive ($EPPAR_n$=00). The outputs of this register drive the corresponding input of the interrupt controller for those bits configured as edge detecting. These bits are cleared by hardware reset.	0 = EPF***n*** selected edge for this pin has not been detected. 1= EPF***n*** selected edge for this pin has been detected.

Chapter 70
Built-In Self Test for RAM and ROM (BIST)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	05/07/03		Updated for LTE specification release.

70.1 Introduction

BIST performs functional test on connected memory arrays. BIST provides functional and structural fault coverage. The memory arrays and all data, address, and control logic are tested with BIST. The BIST can operate in different test modes for production, debug, failure analysis, and burn-in testing. Random access memory (RAM) and read only memory (ROM) are tested with BIST.

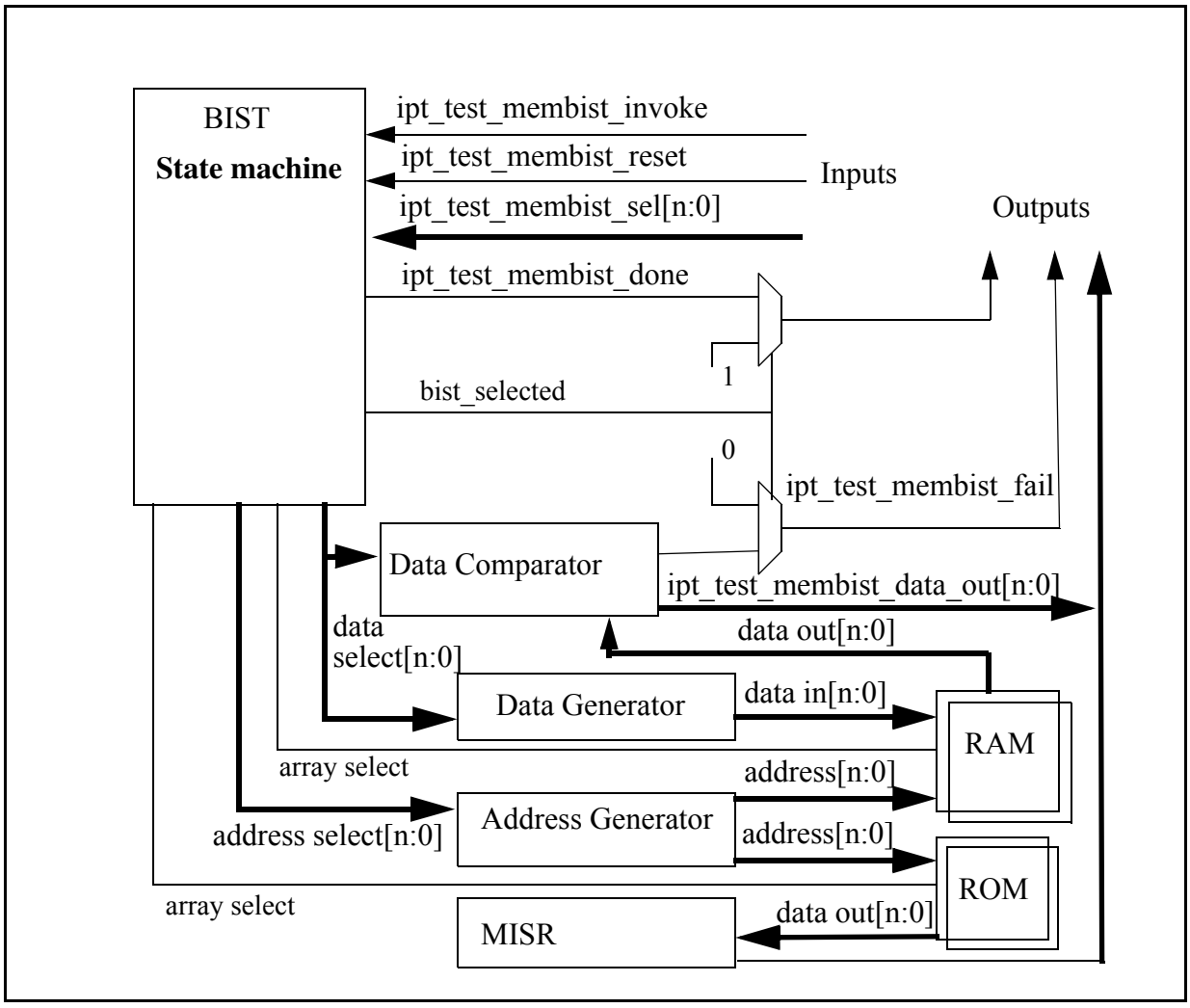


Figure 70-1. BIST Block Diagram

70.2 BIST Module Pin List

The BIST port pins follow the Motorola SRS IPI tan-line naming convention. Eight signals are required.

Table 70-1. BIST Module Pin List

Pin Name	Direction	Description
Input Pins		
ipt_test_membist_sel[n:0]	I	Enables BIST operation on selected arrays in a specified mode. These are static signals that are set before BIST execution starts.
ipt_test_membist_reset	I	Resets BIST to initial state. Assertion during execution aborts BIST execution and returns BIST to initial state. This signal is normally held deasserted during BIST execution.
ipt_test_membist_invoke	I	Invoke controls BIST execution. Assertion causes BIST to start executing the selected test mode. BIST operation when invoke is deasserted depends on selected mode.
Output Pins		
ipt_test_membist_done	O	Asserts at the completion of each defined test sequence.
ipt_test_membist_fail	O	Asserts when fault is detected during a read compare operation of the algorithms. In production mode, this signal is “sticky”, where it asserts and remains asserted until reset is asserted.
ipt_test_membist_data_out[n:0]	O	Data compare results are output through this port.

70.2.1 Input Pins

ipt_test_membist_sel[n:0]

These signals are enable and mode selection for BIST operation. These are static signals set before BIST execution begins. At least four signals are required: one signal for BIST target array selection and one signal to select normal or debug mode. More signals are required if more than one memory array is targeted by this BIST.

ipt_test_membist_sel[1:0] - mode selection signals. These two mode select signals provide four BIST operating modes (Table 70-3 on page 70-7).

ipt_test_membist_sel[n:2] - individual array selects. One select signal for each physical array connected to this BIST. Individual array selects provide scheduling and fault isolation during debug.

ipt_test_membist_reset

BIST reset signal. Assertion aborts BIST execution if BIST is executing and initializes BIST by deasserting outputs **ipt_test_membist_fail** and **ipt_test_membist_done**. Deasserting **ipt_test_membist_reset** after assertion causes BIST to register mode and begin execution with new mode if **ipt_test_membist_invoke** is asserted.

ipt_test_membist_invoke

BIST invoke signal. Assertion starts BIST execution in the selected mode. BIST operation when invoke is deasserted depends on currently selected mode. When production test mode is selected, BIST continues execution until assertion of **ipt_test_membist_reset**, or a fault or done condition are detected. Two scenarios are possible at the end of a test sequence depending on the state of **ipt_test_membist_invoke**. The two scenarios are described here and in Figure 70-2 on page 70-6.

1) First case is where **ipt_test_membist_invoke** is asserted when **ipt_test_membist_done** asserts and there are more test sequences to run (i.e. additional algorithms, data retention, etc.). Signal **ipt_test_membist_done** is deasserted by BIST and the next test sequence starts. When the last sequence completes, done asserts and stays asserted until **ipt_test_membist_reset** is asserted.

2) Second case is where **ipt_test_membist_invoke** is deasserted when **ipt_test_membist_done** asserts and there are more test sequences to run. BIST pauses in current state until **ipt_test_membist_invoke** is asserted. BIST deasserts **ipt_test_membist_done** and starts executing next test sequence. When the last sequence completes, done asserts and stays asserted until **ipt_test_membist_reset** is asserted. This second case is used to synchronize multiple BIST modules where control signals are shared.

When BIST is operating in debug or bitmap mode, **ipt_test_membist_invoke** governs BIST execution dynamically. BIST executes when invoke is asserted. When invoke is deasserted, BIST pauses at the end of current cycle and holds the last state. If **ipt_test_membist_invoke** is asserted again, BIST continues execution from the current state and completes the sequence.

70.2.2 Output Pins

ipt_test_membist_done

This signal asserts on the last cycle of each test sequence. This signal can be “sticky” or dynamic determined by the state of **ipt_test_membist_invoke**. At any time, **ipt_test_membist_reset** can be asserted to clear **ipt_test_membist_done**.

The state of **ipt_test_membist_invoke** during the last cycle of a test sequence determines the next state of **ipt_test_membist_done** (refer to Figure 70-2 on page 70-6). If **ipt_test_membist_invoke** is asserted and more test sequences exist, done deasserts on the next clock cycle and execution of the next test sequence starts. If **ipt_test_membist_invoke** is deasserted, done asserts and stays asserted until **ipt_test_membist_invoke** is asserted again. The BIST is held at the last state until invoke is asserted. This provides a synchronization mechanism when multiple BIST are executing.

If the BIST is disabled, **ipt_test_membist_done** must be switched to the safe state outputting a one.

ipt_test_membist_fail

This signal asserts when a fault is detected. This signal can be “sticky” or dynamic depending on the operating mode selected. When **ipt_test_membist_fail** is sticky, it asserts and stays asserted until **ipt_test_membist_reset** is asserted. When **ipt_test_membist_fail** is a dynamic signal, it asserts on cycles where a fault is detected (read compare operation) and deasserts on cycles where no fault is detected (read compare operation). If **ipt_test_membist_fail** is asserted and the next cycle is not a read compare operation, **ipt_test_membist_fail** is held asserted until the next read compare operation in the test sequence.

If the BIST is disabled, **ipt_test_membist_fail** must be switched to the safe state outputting a zero.

Table 70-2. ipt_test_membist_fail Operating Modes

BIST Operating Mode	ipt_test_membist_fail Signal Operating Mode
production test - stop on first fail	sticky
debug	dynamic
bitmap	dynamic

ipt_test_membist_data_out[n:0]

Output data port for BIST. These signals are dynamic outputs that present the BIST comparator results for external capture. When bitmap mode is selected, BIST executes the normal algorithm sequence outputting compare results from each read cycle. If the internal data width is larger than the data out port, multiple cycles must be used to output the results. The algorithm can be run multiple times where each pass outputs a subset of the data pins or extra data out cycles can be added to the normal BIST sequence.

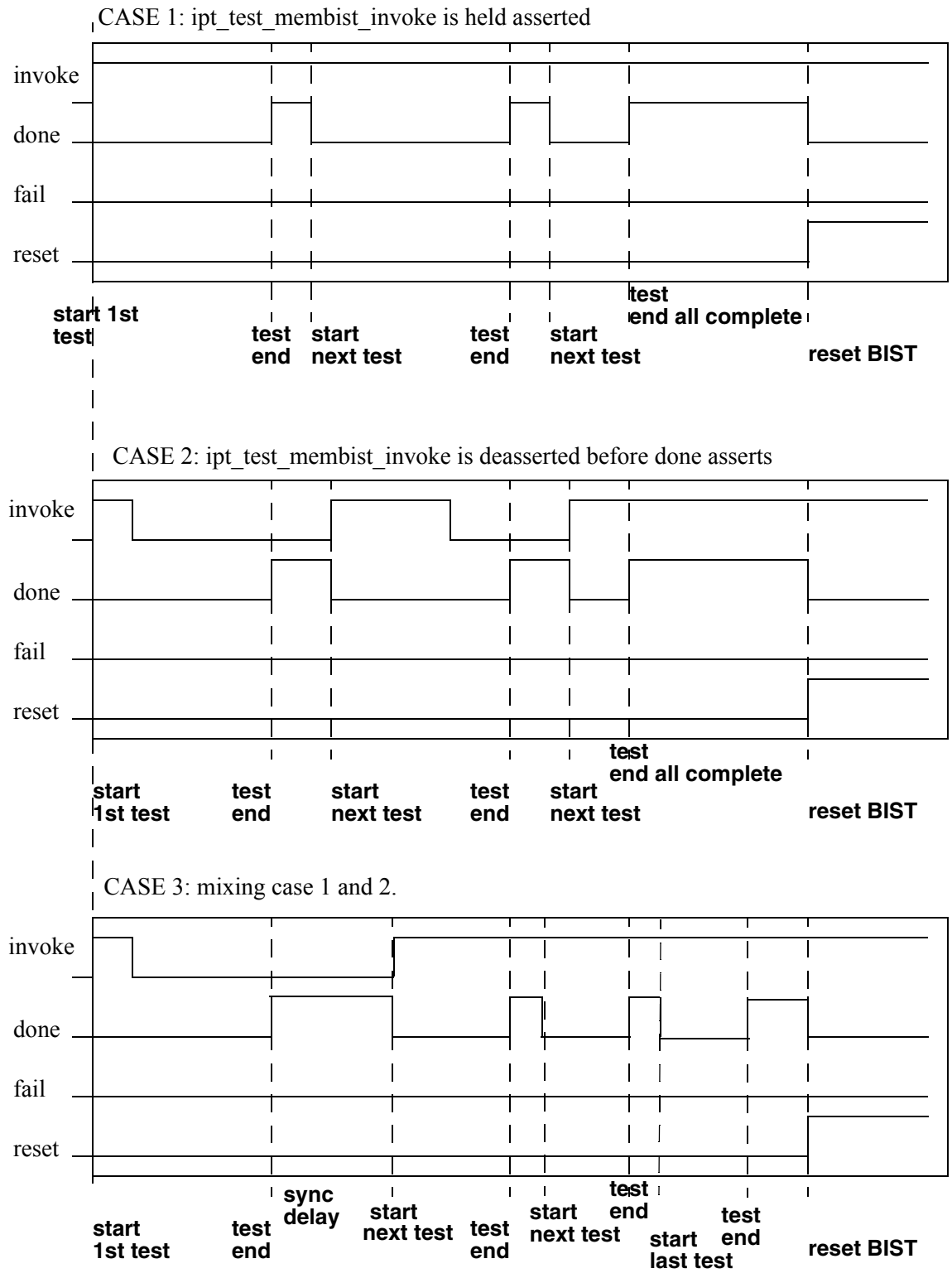


Figure 70-2. BIST Execution Controlled by ipt_test_membist_invoke in Production Test Mode

70.3 BIST Operation

BIST operating modes are selected with the input signals **ipt_test_membist_sel[n:0]**. Signals **ipt_test_membist_sel[1:0]** specify the BIST operation mode. Signals **ipt_test_membist_sel[n:2]** select the arrays connected to the BIST. One or more arrays can be selected at the same time. At least one array must be selected for the BIST to operate.

Table 70-3. BIST Operating Modes

MODE	_sel[1] (mode select 1)	_sel[0] (mode select 0)
production test - stop on first fail	0	0
debug	0	1
bitmap	1	0
ROM MISR data out	1	1

70.3.1 BIST Disabled

BIST is disabled when **ipt_test_membist_sel[n:2]** are not asserted. Output signals **ipt_test_membist_done** and **ipt_test_membist_fail** must be held in a safe state when BIST is disabled (refer to Table 70-4 on page 70-7). If a mux is used to switch to safe state, the internal state can be retained and made visible by activating the array selects.

Table 70-4. Safe States for Output Signals When BIST is Disabled

Signal Name	Safe State
ipt_test_membist_done	1
ipt_test_membist_fail	0

70.3.2 Production Test Mode - Stop on First Fail

Production test mode is enabled when **ipt_test_membist_sel[1:0]** are all zero and at least one array is selected by asserting one or more signals **ipt_test_membist_sel[n:2]**. BIST executes the production test algorithms when **ipt_test_membist_invoke** is asserted and continues running until end of test, a fault is detected, or **ipt_test_membist_invoke** is deasserted.

In production test mode, **ipt_test_membist_fail** is a “sticky” signal where it asserts and stays asserted until **ipt_test_membist_reset** is asserted. BIST executes the algorithms and stops executing when first fault is detected or algorithm is complete.

As described in the port pin description section, **ipt_test_membist_invoke** can be used to control BIST execution during production test mode. If **ipt_test_membist_invoke** is held asserted while BIST is executing, BIST continues executing all test sequences without stopping. Asserting **ipt_test_membist_reset** aborts execution and puts BIST in initial state. If the end of a test sequence is detected and **ipt_test_membist_invoke** is not asserted, BIST pauses at the end of the test sequence and retains the last state until assertion of either **ipt_test_membist_invoke** or **ipt_test_membist_reset**. If assertion of invoke is detected and there are more test sequences, BIST resumes operation with the next test sequence.

Table 70-5. Production Test Mode Stop on Fail Input and Output Interaction

Operation	Inputs		Outputs	
	Invoke	Reset	Done	Fail
BIST executes production test algorithm.	asserted	not asserted	not asserted	not asserted
Deassertion of invoke while BIST is executing does not affect operation unless BIST is at the last cycle of a test sequence.	not asserted	not asserted	not asserted	not asserted
Done asserts during the last cycle of each defined test sequence.If invoke is asserted during this cycle the BIST continues execution at the start of the next test sequence and deasserts done on the next cycle.	asserted	not asserted	asserted	not asserted
Done asserts during the last cycle of each defined test sequence. If invoke is not asserted during the cycle, BIST pauses at end of cycle and holds the last state until invoke or reset is asserted.	not asserted	not asserted	asserted	not asserted
Reset BIST to initial state. If BIST is executing when reset is asserted, execution stops and BIST is reset to initial state.	don't care	asserted	not asserted	not asserted
When BIST detects a fail, BIST execution halts and fail and done are asserted. Execution stops at first fault detected.	don't care	not asserted	asserted	asserted
If BIST completes all algorithms and does not detect any faults, done is asserted and BIST execution stops. Done remains asserted until reset is asserted.	asserted	not asserted	asserted	not asserted

70.3.3 Debug Test Mode - Continue on Fail

Debug test mode executes the production test algorithms. The signal **ipt_test_membist_invoke** starts and stops BIST execution. BIST executes while **ipt_test_membist_invoke** is asserted. Deassertion forces BIST to pause at end of current cycle and hold state. Assertion forces BIST to continue execution.

The signal **ipt_test_membist_fail** is a dynamic signal that asserts when a fault is detected and deasserts when a pass is detected. This signal only changes state during the read cycles where a data comparison is performed. The signal must retain the present state until the next read cycle.

The signal **ipt_test_membist_done** is a dynamic signal that asserts on the last cycle of each defined test sequence. The signal deasserts at the start of the next test sequence.

Debug mode can be used to continuously run BIST. This feature can be used for burn-in exercising the memory. If **ipt_test_membist_invoke** is held asserted then BIST restarts from the beginning after the last cycle of all defined test sequences. Asserting invoke at any time while in debug mode causes BIST to run.

Table 70-6. Debug Test Mode Continue on Fail Input and Output Interaction

Operation	Inputs		Outputs	
	Invoke	Reset	Done	Fail
BIST executes production test algorithm.	asserted	not asserted	not asserted	not asserted
Deassertion of invoke while BIST is executing forces BIST to stop execution and hold state. The pause occurs on the cycle following the detection of invoke deassertion.	not asserted	not asserted	not asserted	not asserted
Done asserts during the last cycle of each defined test sequence. If invoke is asserted during this cycle the BIST continues execution at the start of the next test sequence and deasserts done on the next cycle.	asserted	not asserted	asserted	not asserted
Done asserts during the last cycle of each defined test sequence. If invoke is not asserted during the cycle, BIST pauses at end of cycle and holds the last state until invoke or reset is asserted.	not asserted	not asserted	asserted	not asserted
Reset BIST to initial state. If BIST is executing when reset is asserted, execution stops and BIST is reset to initial state.	don't care	asserted	not asserted	not asserted
When BIST detects a fail, BIST asserts fail. Execution continues even when fault is detected. Fail signal is reset when next valid read cycle occurs. If sequential addresses fail, fail signal stays asserted until passing read operation occurs.	don't care	not asserted	don't care	asserted
If BIST completes all algorithms and does not detect any faults, done is asserted on the last cycle. If invoke is held asserted then BIST restarts from beginning as if reset had been asserted.	asserted	not asserted	asserted	not asserted

70.3.4 ROM MISR Data Out Mode

ROM MISR data out mode outputs the internally stored ROM MISR value on the **ipt_test_membist_data_out[n:0]** output pins. Asserting **ipt_test_membist_invoke** starts output of the MISR value. Where the **ipt_test_membist_data_out** port is smaller than the MISR width, multiple clock cycles are applied to output the full MISR value.

After normal completion of a ROM BIST module production test execution, ROM BIST stops and the MISR contains the final compressed result. The BIST holds that MISR result until **ipt_test_membist_reset** is asserted. Selecting this mode and a ROM array causes the BIST to output the stored MISR value.

70.3.5 Bitmap Mode

Bitmap mode provides visibility of internal operation during algorithm execution. The read cycle data is presented on the **ipt_test_membist_data_out[n:0]** outputs. The internal address is not presented but can be determined from monitoring the cycle count in the algorithm sequence. BIST operation in bitmap mode is the same as debug test mode.

Bitmap mode on RAM arrays requires for every read compare cycle in the test algorithm, the array data are output. To monitor BIST with automatic test equipment (ATE), a test pattern that mimics the bist algorithm can be used to identify individual failing bit cells. Data for all outputs must be presented. Where the number a data out pins is less the output width of the memory, multiple cycles or multiple passes of the algorithm must be used to output all of the data.

Bitmap mode for ROM arrays requires for every read cycle, the actual ROM data are output.

70.4 RAM Fault Model and Test Algorithm

70.4.1 RAM Fault Model

The fault model is based on the IFA-13 fault model (Dekker 1990). IFA-13 is an extension of March C- and provides fault coverage for all simple faults (a given fault does not influence the behavior of other faults). This fault model provides a high degree of fault coverage and can be implemented with a simple march algorithm.

Table 70-7. Unlinked Functional Faults

Fault Class	March Element(s) Required to Sensitize Fault
(AF) Address decoder faults where fault is only contained in the address decode logic.	^(rx, . . . , $\overline{wx}$) v($\overline{rx}$, . . . , wx)
(SAF) Stuck-at fault where the value of a cell is always 0 (SA0) or the value is always 1 (SA1)	(. . . , w0 , r0 , . . .) SA1 (. . . , w1 , r1 , . . .) SA0
(SOF) Stuck-open fault where a memory cell can not be accessed	(. . ., rx, . . . , $\overline{rx}$, . . .)
(TF) Transition fault where a cell fails to transition from 1 to 0 or from a 0 to 1	(. . . , w0 , r0 , . . .) (v/1)TF (. . . , w1 , r1 , . . .) (^/0)TF
(DRF) Data retention fault where a cell fails to retain state for time T	simple DRFs are detected by placing a delay between any two march elements ^(... , wx); delay; v(rx,...,$\overline{wx}$)
(CFin) Inversion coupling fault where a transition in the coupling cell causes an inversion in the coupled cell (CFid) Idempotent coupling fault where a transition in the coupling cell forces a 0 or 1 in the coupled cell. (CFst) State coupling fault where a coupled cell is forced to a value 0 or 1 when the coupling cell is in a certain state (CFdst-w) Disturb by write coupling fault where the coupled cell is disturbed (makes a transition) when the coupling cell undergoes a write operation	all coupling faults covered by including one of the march elements from Case A and one from Case B. Case A: ^(rx,...,$\overline{wx}$,$\overline{rx}$,...,wx);^(rx,...) v(rx,...,$\overline{wx}$,$\overline{rx}$,...,wx);v(rx,...) Case B: ^($\overline{rx}$,...,wx,rx,...,$\overline{wx}$);^($\overline{rx}$,...) v($\overline{rx}$,...,wx,rx,...,$\overline{wx}$);v($\overline{rx}$,...)

Word oriented memories require extensions to the march sequence to cover multi-bit data paths. The simplest method entails running the entire march algorithm multiple times with different data backgrounds on each execution. The number of data backgrounds is dependent on the width of the IO (Dekker 1988). For devices where the IO width is a power of 2, the equation is:

$$K = \log_2(m) + 1 \qquad \textit{Eqn. 70-1}$$

where m is the number of IO and K is the number of patterns required. For a 32 bit wide memory, 6 patterns are required (see Table 70-8 on page 11). This is not the most efficient method but is generally easy to implement in a simple BIST controller. A more efficient approach is to modify the march algorithm to cover multiple IO related faults. This may not be feasible without increasing the complexity of the BIST.

Table 70-8. Data Backgrounds Required for 32 I/O Memory

Data Background	Data Complement
0000 0000 0000 0000 0000 0000 0000 0000	1111 1111 1111 1111 1111 1111 1111 1111
0000 0000 0000 0000 1111 1111 1111 1111	1111 1111 1111 1111 0000 0000 0000 0000
0000 0000 1111 1111 0000 0000 1111 1111	1111 1111 0000 0000 1111 1111 0000 0000
0000 1111 0000 1111 0000 1111 0000 1111	1111 0000 1111 0000 1111 0000 1111 0000
0011 0011 0011 0011 0011 0011 0011 0011	1100 1100 1100 1100 1100 1100 1100 1100
0101 0101 0101 0101 0101 0101 0101 0101	1010 1010 1010 1010 1010 1010 1010 1010

Table 70-9. Data Backgrounds Required for 24 I/O Memory

Data Background	Data Complement
0000 0000 0000 0000 0000 0000	1111 1111 1111 1111 1111 1111
0000 0000 0000 1111 1111 1111	1111 1111 1111 0000 0000 0000
0000 1111 1111 0000 0000 1111	1111 0000 0000 1111 1111 0000
0000 1111 0000 1111 0000 1111	1111 0000 1111 0000 1111 0000
0011 0011 0011 0011 0011 0011	1100 1100 1100 1100 1100 1100
0101 0101 0101 0101 0101 0101	1010 1010 1010 1010 1010 1010

Table 70-10. Data Backgrounds Required for 16 I/O Memory

Data Background	Data Complement
0000 0000 0000 0000	1111 1111 1111 1111
0000 0000 1111 1111	1111 1111 0000 0000
0000 1111 0000 1111	1111 0000 1111 0000
0011 0011 0011 0011	1100 1100 1100 1100
0101 0101 0101 0101	1010 1010 1010 1010

70.4.2 RAM Test Algorithm

The test algorithm is derived from IFA-13 march (Dekker 1990). The march sequence involves sequential access through the memory address range applying a sequence of reads and writes to each address. The process repeats for each of the march sequences specified (six sequences are defined in the IFA-13 algorithm).

IFA-13 14n march algorithm (14 operations on each address)

```
{ v(WX̄); ^(RX̄,WX,RX); ^(RX,WX̄,RX̄); v(RX̄,WX,RX); v(RX,WX̄,RX̄); ^(RX̄) }
  M0      M1           M2            M3           M4            M5
```

IFA13 can be extended to include data retention faults by adding another sequence and inserting the delays. IFA-13 is then a 16N algorithm.

```
{ v(WX̄);^(RX̄,WX,RX);^(RX,WX̄,RX̄);v(RX̄,WX,RX);v(RX,WX̄,RX̄);delay;^(RX̄,WX);delay;v(RX); }
    M0       M1      M2            M3            M4            M5           M6
```

March C- 10n march algorithm.

```
{ v(WX̄); ^(RX̄,WX); ^(RX,WX̄); v(RX̄,WX); v(RX,WX̄); ^(RX̄) }
   M0      M1 M2     M3     M4 M5
```

Table 70-11. Fault Coverage for Specified March Algorithms

march test	length	AF	SAF	TF	CFst	CFin	CFid	CFdst-w	DRF	SOF
IFA-13	14n	+	+	+	+	+	+	+		+
IFA-13 with data retention	16n+2D	+	+	+	+	+	+	+	+	+
March C-	10n	+	+	+	+	+	+	+		

70.4.2.1 Algorithm For Late Write Buffered RAM

Any march algorithm can be extended with additional operations without reduction in fault coverage. Extending the IFA-13 algorithm to make all sequences the same number of operations per address and to force writes through the late write buffer to the array, yields the following algorithm. This algorithm incorporates write recovery where every bit cell is written with true and complement data during every write sequence. This algorithm has order 24N.

```
M0 - A[min..max]^ (WX,WX,WX̄,WX̄);
M1 - A[min..max]^ (RX̄,WX̄,WX,WX);
M2 - A[min..max]^ (RX,WX,WX̄,WX̄);
M3 - A[max..min]v (RX̄,WX̄,WX,WX);
M4 - A[max..min]v (RX,WX,WX̄,WX̄);
M5 - A[max..min]v (RX̄,WX̄,WX,WX);
```

The algorithm is repeated for each data background listed in tables 2, 3, and 4 depending on targeted memory IO width (32 IO RAM requires 6 backgrounds so six repetitions of the algorithm are required).

RAMs designed with a late write buffer require multiple write cycles to propagate the write through the buffer to the array. Writing to an address and immediately reading the contents stored in the array requires preforming a write to a different address to flush the late write buffer. Any read to the address of the last write will access data stored in the late write buffer bypassing access to the array. A write operation to a new address flushes the buffer and enables read from the array.

The extended algorithm sequence requires four operations on each address. Addresses are never toggled at full bus frequency. To provide full frequency read and write coverage to all addresses an additional sequence should be applied.

```
M0 - A[min..max-1]^ a[n](WX);a[n+1](WX̄);
M1 - A[min..max-1]^ a[n](RX);a[n+1](RX̄);
M0 - A[max..min+1]v a[n](WX);a[n-1](WX̄);
M1 - A[max..min+1]v a[n](RX);a[n-1](RX̄);
```

This algorithm should be applied with the data background 0x5555 and the complement 0xAAAA. This combination will write and read a logical checkerboard (alternating zeros and ones on the same bitline) to the memory array and a new address is selected every cycle. Depending on actual array implementation, this method does not guarantee true checkerboard data pattern (every adjacent bitcell is at the opposite data state). More knowledge of the actual layout and addressing scheme is required to create a true checkerboard.

70.4.3 Data Retention

The desired method for implementing data retention requires setting a specific state in all memory cells, lowering the voltage to the array and hold for a specified time, raising voltage and verifying previous cell state. The following steps implement a retention test.

1. write array with data under nominal operating conditions
2. put BIST is pause state and maintain array state
3. lower voltage to lowest safe voltage (all logic states maintained)
4. pause (100msec to 1sec or more, optimal time determined from silicon evaluation)
5. raise voltage to nominal operating conditions
6. read array
7. repeat sequence with complement data

Data retention should be performed with both data states (1 and 0) and could be implemented with the following algorithm.

```
data background 0x5555 and complement 0xAAAA

            M0 - A[min..max-1]^ a[n](WX);a[n+1](WX̄);
            pause
            M1 - A[min..max-1]^ a[n](RX);a[n+1](RX̄);
            M0 - A[max..min+1]v a[n](WX);a[n-1](WX̄);
            pause
            M1 - A[max..min+1]v a[n](RX);a[n-1](RX̄);
```

70.5 ROM Fault Model and Test Algorithm

70.5.1 ROM Fault Model

ROM requires a reduced functional fault model. Stuck-at, coupling, address decoder, and transition faults are covered by ROM BIST. The data in the ROM is fed into a multiple input shift register and compressed using polynomial division (Cycle Redundancy Checking). The final result in the MISR is compared to a known good value to verify the ROM contents are correct.

70.5.2 ROM Test Algorithm

Algorithm consists of accessing each memory location of ROM is sequential order starting at minimum address and incrementing to max address. Sequence is repeated addressing memory from max address to min address.

ROM MISR is initially seeded with a known fixed input. ROM signature comparison is performed externally after outputting MISR contents. MISR data is output on the **ipt_test_membist_data_out[n:0]** pins.

The test sequence is a simple up down scan read of the ROM array. For each read operation, the read value is input to the MISR.

```
M0 - A[min..max]^ (RX);
M1 - A[max..min]v (RX);
```

70.6 Integration of Multiple BIST Modules

System level integration of multiple BIST modules is possible with this BIST specification. Given the following assumptions, Figure 70-3 on page 70-15 depicts a possible system level integration of multiple BIST modules. This integration scheme can be used at the top level of the IC design and at lower sub-module levels. The same common interface is used for all BIST modules.

70.6.1 Assumptions and Guidelines for BIST Integration

1. *Individual selects (* **ipt_test_membist_sel[n:2]** *) to each array connected to a BIST module.*
2. Wire-ored mode selection to all BIST (**ipt_test_membist_sel[1:0]**).
3. Wire-ored invoke and reset to all BIST (**ipt_test_membist_invoke** and **ipt_test_membist_reset**).
4. Multiplex all data out to a common data out port (**ipt_test_membist_data_out[n:0]**).
5. AND of all done signals (**ipt_test_membist_done**); asserts when all BIST are done. Each BIST has a safe state for **ipt_test_membist_done** that is output when the BIST is not selected. Safe state for **ipt_test_membist_done** is one (see Figure 70-1 on page 70-2).
6. OR of all fail signals (**ipt_test_membist_fail**); asserts when any BIST fails. Each BIST has a safe state for **ipt_test_membist_fail** that is output when the BIST is not selected. Safe state for **ipt_test_membist_fail** is zero (see Figure 70-1 on page 70-2).
7. When multiple BIST are combined in a sub-module and the BIST share a common address or data bus, a scheduler must be used to prevent contention between BIST modules. Scheduling is also possible at the top level where individual array/BIST selects can be employed. Scheduling at the sub-module level ensures bus contention can not occur.

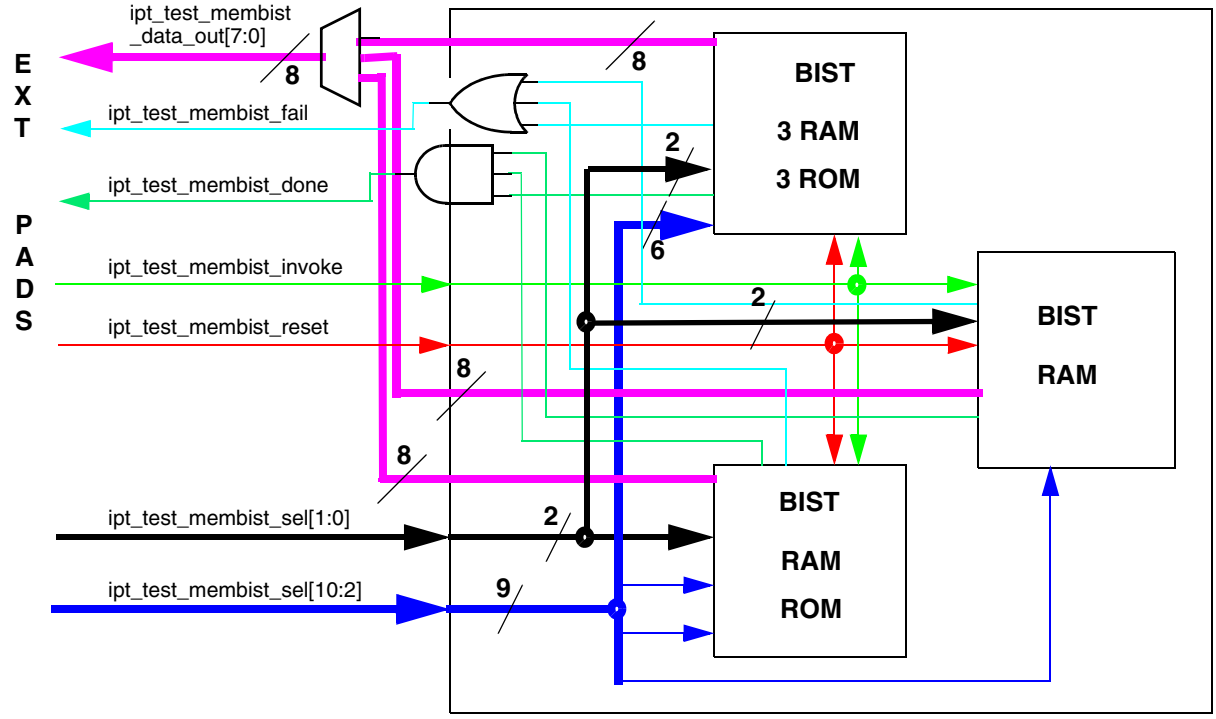


Figure 70-3. Example System Level Integration of Multiple BIST Modules

Chapter 71
System JTAG Controller (SJC)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
1	5/6/02	Chris Daniels	Updated for Neptune LTS BIST
1.1	05/07/03		Updated for LTE specification release.

71.1 Introduction

The Neptune IC has an integrated S-ONYXU DSP and ARM7TDMI-S MCU on the same die and includes two JTAG TAP controllers (the ARM MCU Multi-ICE™ TAP controller, and the ONYXU DSP OnCE™ TAP controller.) and some system logic. Neptune requires three modes:

1. DSP stand alone mode
2. ARM stand alone mode
3. Concatenation of MCU and DSP

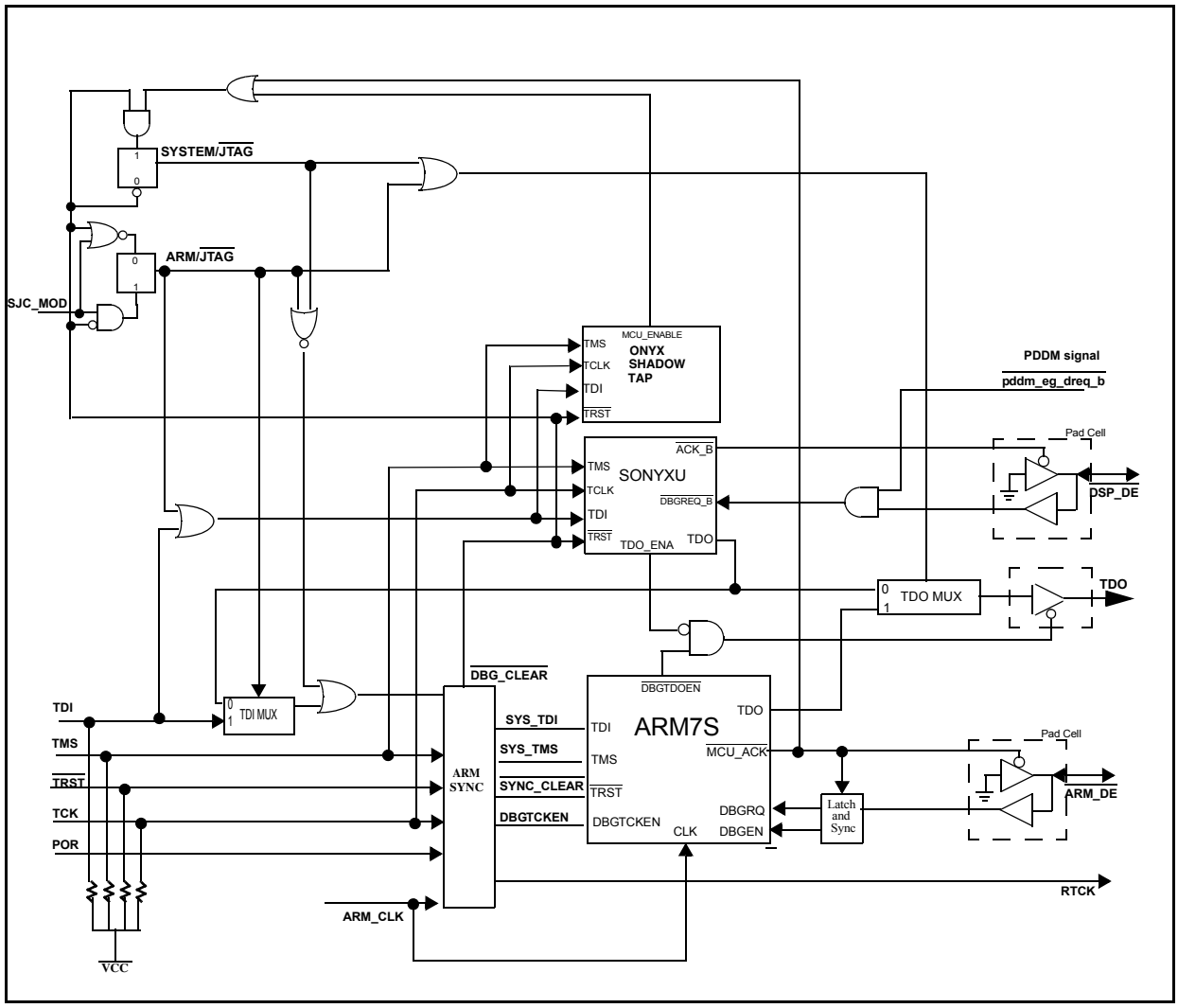


Figure 71-1. Neptune JTAG Block Diagram

The SJC also provides the following features:

1. Forcing a Debug Mode on both cores in case of a breakpoint occurrence in one of the cores, (e.g a breakpoint on the ONYXU DSP will cause a debug request on the ARM MCU) when enabled by the Dual Core control register.
2. Simultaneously releasing of the ARM core and the ONYXU core after both core's emulation modules had entered the Debug Mode.

The Onyx mode is entered by holding the SJC_MOD pin low during a rising edge of $\overline{\text{DBG\_CLEAR}}$. In Onyx, mode only the System ONYXU DSP TAP controller is active. The ARM TAP is removed from the scan chain by holding ARM TDI high, and disconnecting ARM TDO. In JTAG Onyx mode, the control logic monitors the commands shifted into the DSP TAP controller. If the control logic detects the MCU_ENABLE command shifted into the DSP TAP controller, the SYSTEM/$\overline{\text{JTAG}}$ signal goes high. In this state, the two tap controllers will be linked.

ARM mode is entered by holding the SJC_MOD pin high during a rising edge of $\overline{\text{DBG\_CLEAR}}$ (see Section 71.2.3, "ARM Synchronization,"). In ARM mode, all JTAG pins are connected to the ARM core through the JTAG synchronizer. The ONYXU DSP TAP is removed from the scan chain by holding Onyx TDI high, and disconnecting Onyx TDO. This allows any off the shelf ARM Multi-ICE™ device to function.

The test logic includes a test access port (TAP) consisting of four dedicated signal pins, a 16-state controller, and three test data registers.

71.2 Neptune JTAG OVERVIEW

71.2.1 PAD SIGNALS

71.2.1.1 Test Clock (TCK)

The test clock input (TCK) pin is used to synchronize the test logic.

NOTE:

TCK must be held at its current level until RTCK reaches that level. If no RTCK is available, TCK period must be eight times greater than the arm clock (A_CLK).

See Section 71.4, “Serial ARM-ONYX TDI-TDO Waveform,” .

71.2.1.2 Return Test Clock (RTCK)

This test clock output is used to confirm when TCK has traveled through the synchronization logic. It is also used to confirm when to transmit data and when to receive data.

71.2.1.3 Test Mode Select (TMS)

The test mode select input (TMS) pin is used to sequence the test controller’s state machine. TMS is captured during the rising edge of TCK. TMS includes an internal pull-up resistor.

71.2.1.4 Test Data Input (TDI)

Serial test instruction and data are received through the test data input (TDI) pin. TDI is captured during the rising edge of TCK. TDI includes an internal pull-up resistor.
The test data input for the ARM MCU and the ONYXU DSP is gated by the Neptune system logic.

71.2.1.5 Test Data Out (TDO)

Serial instruction and data is transmitted through the test data out (TDO) pin. TDO should be captured during the rising edge of TCK.

71.2.1.6 Test Reset ($\overline{\text{TRST}}$)

Asserting TRST will cause

- Reset of the Onyx and ARM TAP controllers to their TEST LOGIC RESET state
- Reset of the synchronization logic
- Re-sampling of GPIO to determine mode

The JTAG controllers will maintain reset two mcu clocks after the release of $\overline{\text{TRST}}$

71.2.1.7 SJC_MOD

Sampled after the release of $\overline{\text{TRST}}$. If $\overline{\text{TRST}}$ is not asserted in a POR event, SJC_MOD will be sampled after a POR. This signal determines which mode JTAG is initially in. 1= ARM Mode 0= Onyx mode.

71.2.2 SJC Pin List

Table 71-1 lists all the pins in the SJC module.

Table 71-1. SJC Pin List

Pin Name	Direction	Description	Active
Pad Signals			
sjc_pad_dbgtdoen_b	Output	Pad TDO enable	low
sjc_rtck	Output	return test clock	
sjc_tdo	Output	test data out	
gpio	Input	captured or rising edge of $\overline{\text{TRST}}$ 1 = ARM mode 0 = JTAG mode	
pad_a_dbgreq_b	Input	Debug request for the ARM from the PAD	low
pad_o_dbgreq_b	Input	Debug request for the ONYX from the PAD	low
tck	Input	test clock	
tdi	Input	test data in	
tms	Input	test mode select	
trst_b	Input	test reset	Low
ARM signals			
sjc_a_dbgen	Output	ARM debug enable	high
sjc_a_dbgrq	Output	ARM debug request	high
sjc_a_dbgtcken	Output	ARM test clock enable	high
sjc_a_tdi	Output	ARM test data in	
sjc_a_tms	Output	ARM test mode select	
sjc_a_trst_b	Output	ARM test reset	low
a_clk	Input	ARM clock	
a_dbgtdoen_b	Input	ARM TDO enable	low
a_mcu_ack_b	Input	ARM acknowledge of debug mode	low
a_tdo	Input	ARM test data out	
ONYX signals			
sjc_o_dbgreq_b	Output	ONYX debug request	low
sjc_o_tck	Output	ONYX test clock	
sjc_o_tdi	Output	ONYX test data in	

Table 71-1. SJC Pin List

Pin Name	Direction	Description	Active
sjc_o_tms	Output	ONYX test mode select	
sjc_o_trst_b	Output	ONYX test reset	low
o_ack_b	Input	ONYX acknowledge of debug mode	low
o_clk	Input	ONYX clock	
o_dbgtdoen	Input	ONYX TDO enable	low
o_tdo	Input	ONYX test data out	
PDDM signals			
pddm_eg_dreq_b	Input	PDDM ONYX debug request	low
system signals			
por_b	Input	power on reset	high
BIST signals			
sjc_bist_enable	Output	Enable DSP BIST clock	high
sjc_membist_invoke	Output	Start DSP BIST engine	high
sjc_membist_reset	Output	Reset DSP BIST engine	high
sjc_membist_release	Output	Release self pause	high
sjc_membist_retention_en	Output	Enable self pause	high
sjc_j_gp_regsel	Output	BIST ABCR register select	high
sjc_sjt_bist_shift_dr	Output	BIST shift data reg clk	-
sjc_sjt_bist_update_dr	Output	BIST update data reg clk	-
sjc_sab_tdi	Output	BIST data in	-
sjc_gpio_prod_cnfg	Output	GPIO configuration to BIST production mode	high
sjc_gpio_bit_map_cnfg	Output	GPIO configuration to BIST bit mapping mode	high
sjc_sio_acs	Output	BIST I/O serial access strobe	high
sab_tdo	Input	BIST data out	-
tcm_dsp_bist_bypass_mode	Input	TCM DSP BIST bypass request	high

Table 71-1. SJC Pin List

Pin Name	Direction	Description	Active
scan signals			
ipt_clk_se	Input	scan enable	high
ipt_scan_mode	Input	scan mode	high
ipt_test	Input	in test mode	high

71.2.3 ARM Synchronization

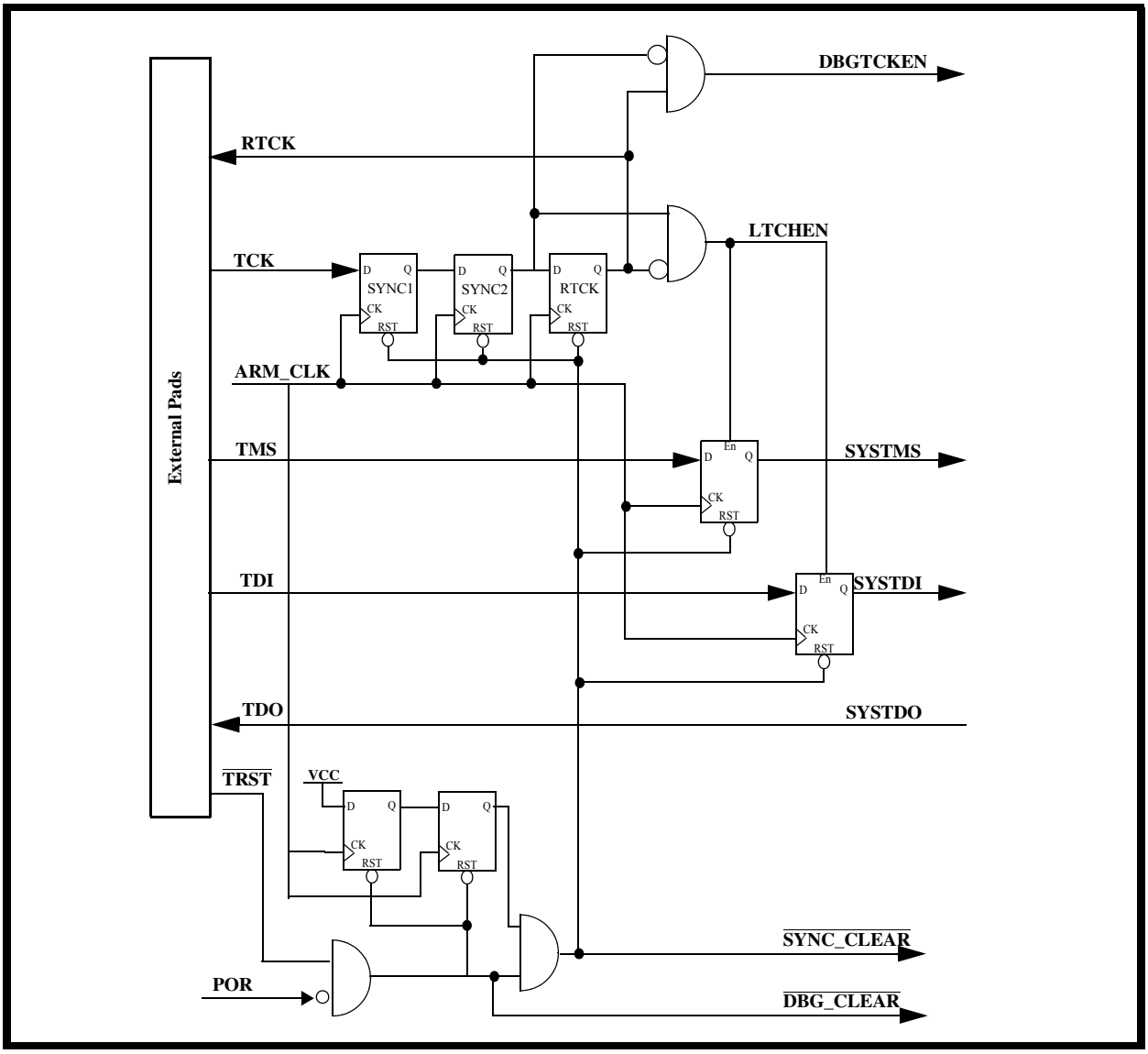


Figure 71-2. ARM7TDMI-S synchronization circuit

The ARM7TDMI-S has only one clock domain (CLK). In order to accommodate scan, TCK, TMS, and TDO is synchronized with CLK. This synchronization circuit will be placed in the SJC. All JTAG TAP will use RTCK to keep their tap synchronous with the ARM TAP. See Section 71.4, “Serial ARM-ONYX TDI-TDO Waveform,”

71.2.4 SJC Modes

Two modes are created based on the gpio signal SJC_MOD. These modes are used to maintain compatibility to ARM MCU Multi-ICE™ products, as well as maintain IEEE JTAG standards. SJC_MOD will be latched on the rising edge of $\overline{\text{TRST}}$ or POR. Changing modes will require re-assertion of $\overline{\text{TRST}}$ or POR

71.2.4.1 Onyx Mode (SJC_MOD = 0)

This mode will directly connect TMS, TDI and SYSTDO to the Onyx TAP controller. The ARM TAP will be forced to bypass mode by placing TDI high. This will provide a dedicated user-accessible test access port (TAP) that uses the same communication style as the IEEE 1149.1 Standard Test Access Port and Boundary Scan Architecture. In system mode the onyx TAP supports the following capabilities:

1. Bypass the Neptune for a given circuit-board test by effectively reducing the boundary scan register to a single cell (BYPASS).
2. Disable the output drive to pins during circuit-board testing (HIGHZ).
3. Query identification information (manufacturer, part number and version) from the Onyx (IDCODE).
4. Provide a means of serial chaining the ONYXU DSP TAP and the ARM MCU Multi-ICE™ circuits to control a target system (MCU_ENABLE).

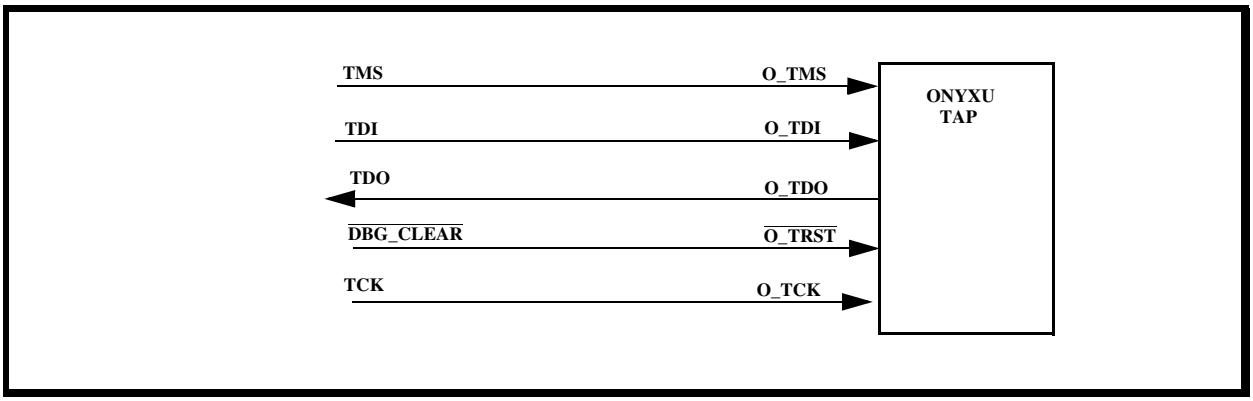


Figure 71-3. Onyx (ONYX configuration)

71.2.4.2 Enabling ARM-Onyx JTAG Concatenation

Placing the ARM and Onyx in a serial chain can only be done from Onyx mode (SJC_MOD = 0). Once in Onyx mode, the concatenating the ARM and ONYX can be completed by two methods:

1. Assertion of the MCU_DE line while the TAP controllers are in the Test-Logic-Reset state and the $\overline{\text{TRST}}$ input is negated. This will cause entry into Onyx mode when the ARM acknowledges the debug request.
2. Shifting the ENABLE_MCU command into the DSP TAP controller.(IR = 4'b0011)

The SJC logic will configure the TDI and TDO muxes, so that the two controllers are connected in a serial manner: with the DSP TAP controller closest to the TDI input and the MCU TAP controller closest to the TDO output pin. In this configuration, the MCU instruction register and data registers are concatenated with the DSP instruction register and data registers to form a combined IR, and a combined DR. The composition of the combined DR is a function of the respective instructions present in the two combined IR sections.

JTAG compliance may be re-established from the serial chain configuration by assertion of $\overline{\text{TRST}}$ with the $\overline{\text{MCU\_DE}}$ pin held negated.

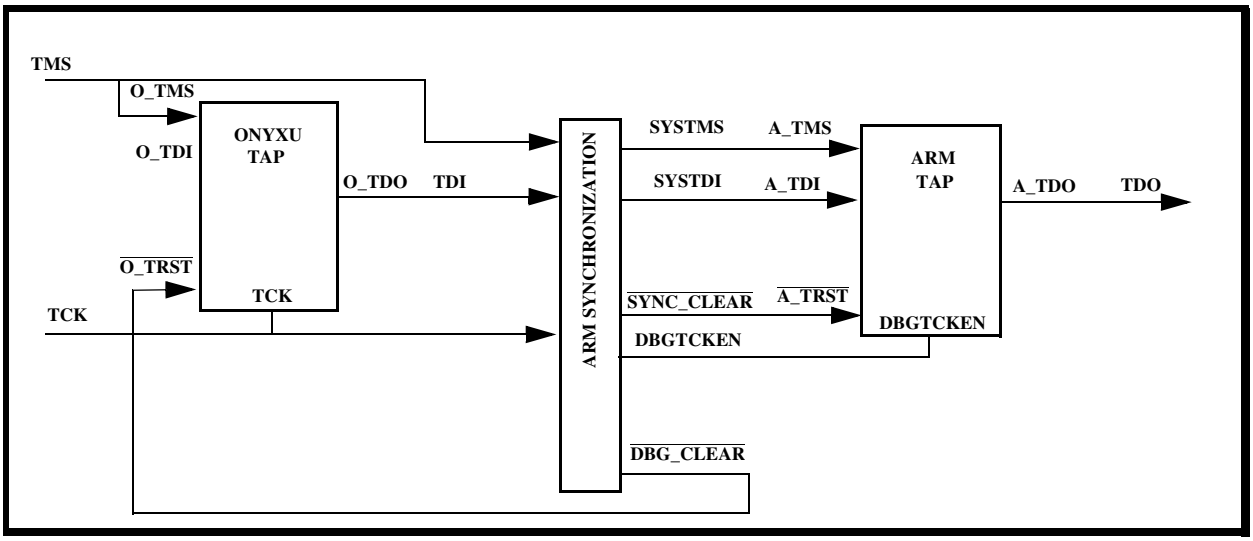


Figure 71-4. Onyx (Concatenation mode)

71.2.4.3 ARM Mode (SJC_MOD = 1)

This mode will directly connect SYSTMS, SYSTDI and SYSTDO directly to the ARM core. The Onyx TAP will be forced to bypass mode by forcing TDI high. This will allow ARM MCU Multi-ICE™ products to interface with the ARM core without extra programming. $\overline{\text{TRST}}$ must be asserted to leave this mode.

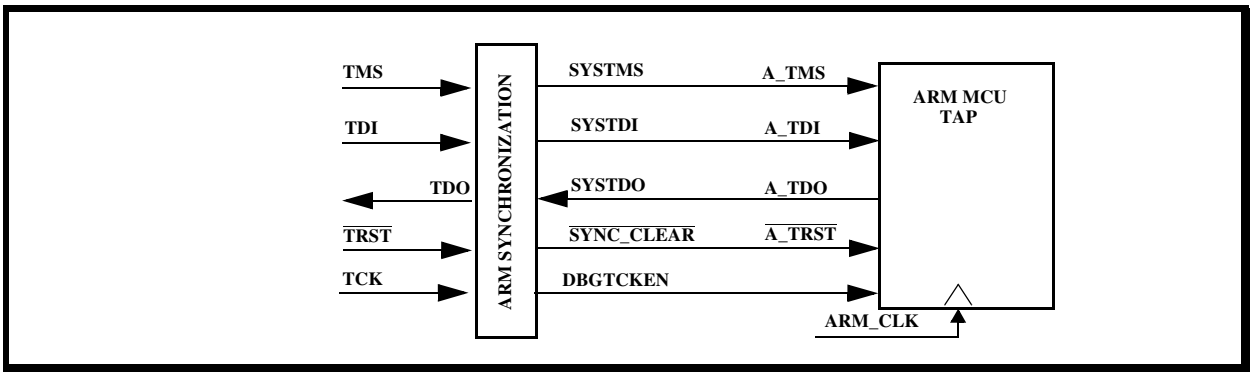


Figure 71-5. ARM Mode Configuration

71.2.5 DSP JTAG Overview

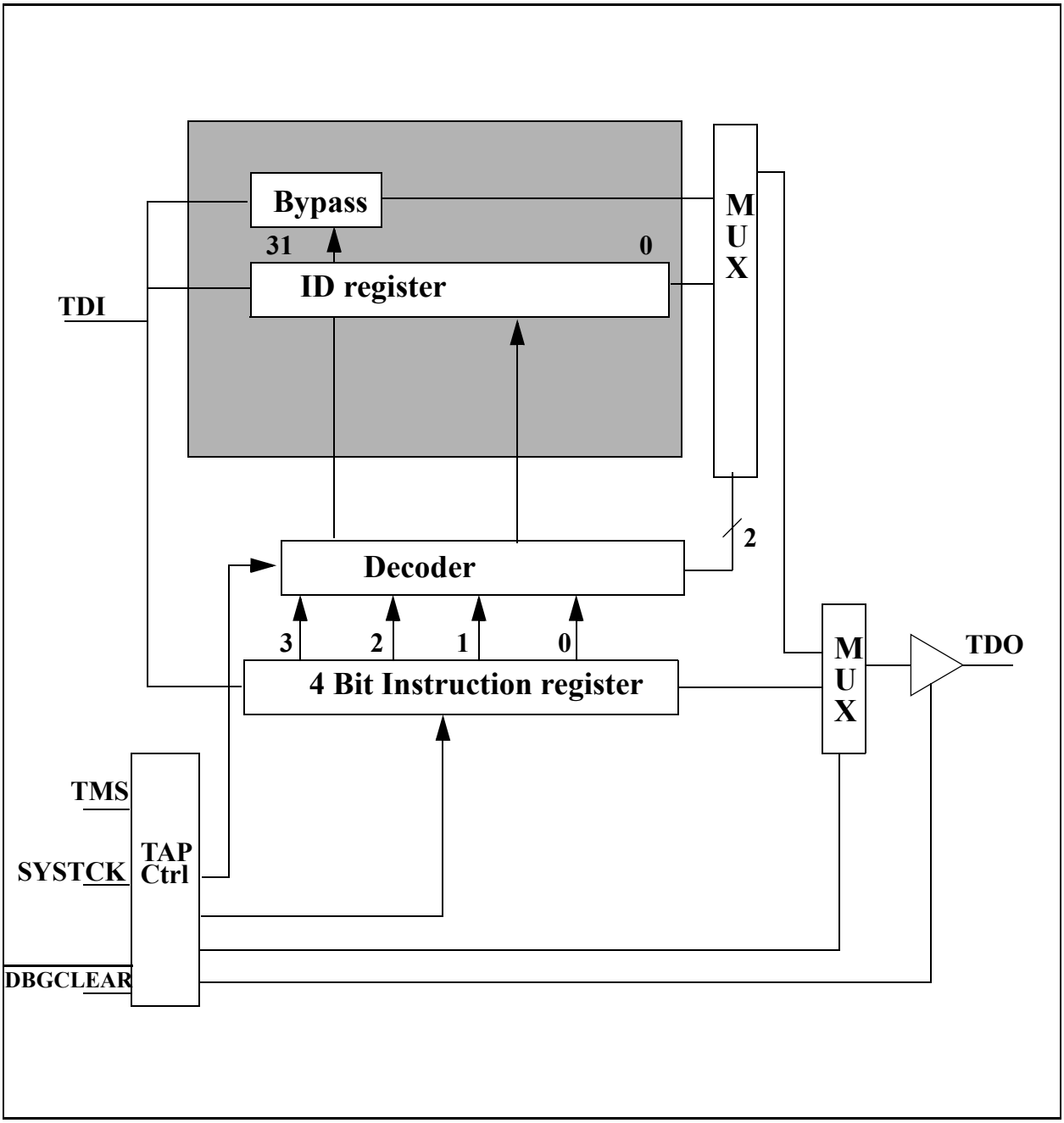


Figure 71-6. ONYXU JTAG Block Diagram

The DSP Core implementation includes a 1-bit bypass register, a 32-bit identification register and a boundary scan register. This implementation includes a dedicated TAP and five pins for the interface. TDI, TMS, and SYSTCK are from the ARM synchronization portion of the SJC.

71.2.5.1 TAP CONTROLLER

The TAP controller is responsible for interpreting the sequence of logical values on the TMS signal. It is a synchronous state machine that controls the operation of the JTAG logic. The state machine is shown in Figure 71-7 on page 14. The value shown adjacent to each arc represents the value of the SYSTMS signal sampled on the rising edge of SYSTCK signal. For a description of the TAP controller states, please refer to the IEEE 1149.1 document.

71.2.5.2 INSTRUCTION REGISTER

The ONYX JTAG supports the public instruction HIGHZ, and provides the capability for disabling all device output drivers. Another public instructions enables the Serial Chaining logic (MCU_ENABLE) The Neptune JTAG implementation includes a 4-bit instruction register without parity consisting of a shift register with four parallel outputs. Data is transferred from the shift register to the parallel outputs during the Update-IR controller state. The four bits are used to decode the eight unique instructions shown in Table 71-2 on page 15. All other encoding options are reserved for future enhancements and are decoded as BYPASS.

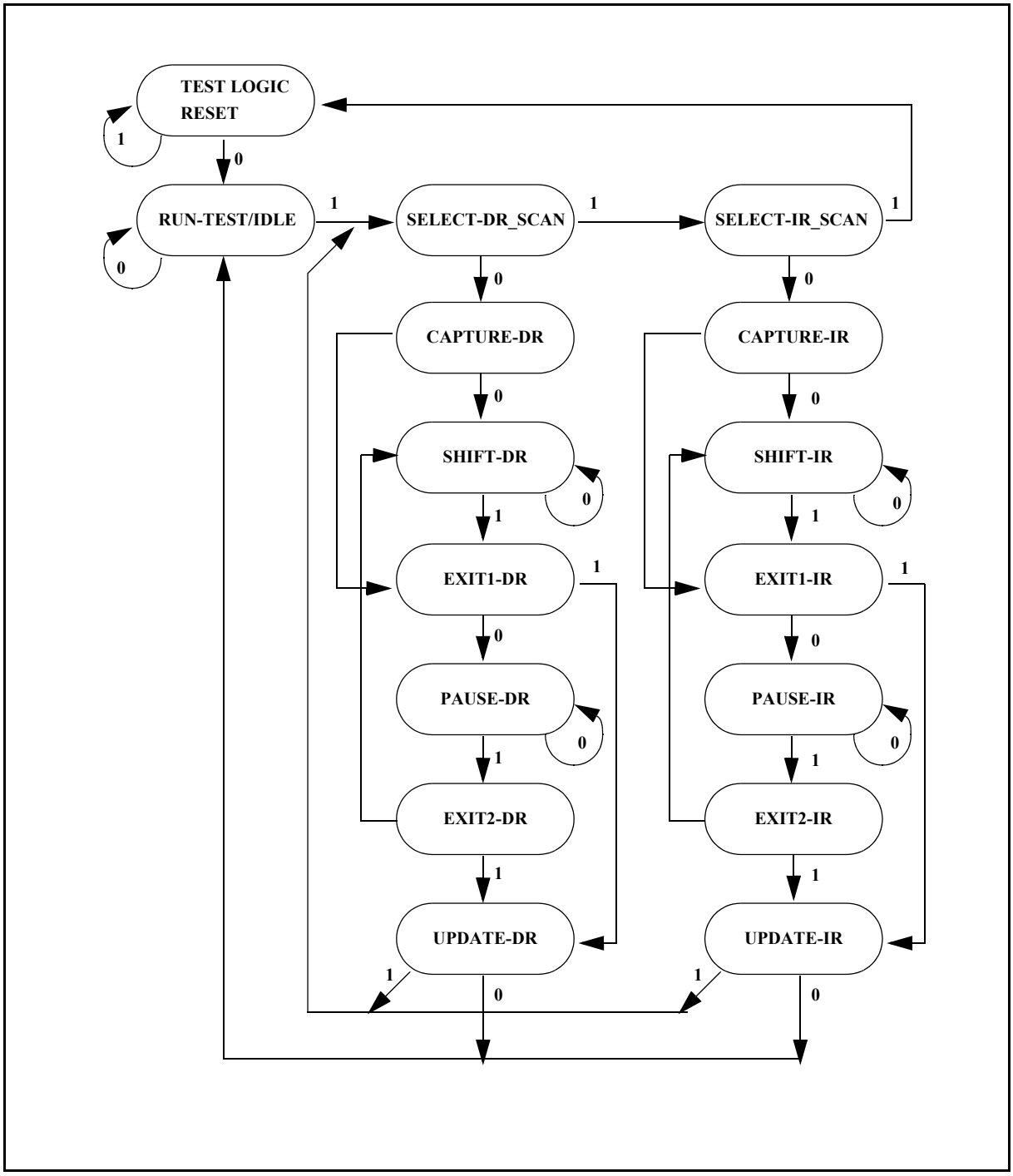


Figure 71-7. TAP Controller State Machine

The instruction register is reset to 0b0010 in the test-logic-reset controller state which is equivalent to the IDCODE instruction.
During the capture-IR controller state, the parallel inputs to the instruction shift register are loaded with 0b01 in the least significant bits as required by the standard. The two most significant bits are loaded with the values of the core status bits OS1 and OS0 from the OnCE controller.

Figure 71-8. Instruction Register Capture Value

3	2	1	0
OS1	OS0	0	1

Table 71-2. JTAG Instructions

Code				Instruction
B3	B2	B1	B0	
0	0	0	0	RESERVED
0	0	0	1	RESERVED
0	0	1	0	IDCODE
0	0	1	1	ENABLE_MCU
0	1	0	0	HIGHZ
0	1	0	1	RESERVED
0	1	1	0	ENABLE_DSP_ONCE
0	1	1	1	DSP_DEBUG_REQUEST
1	0	0	0	LOAD_ABISTCR
1	0	0	1	LOAD_JTCR
1	0	1	0	BYPASS
1	1	X	X	BYPASS

71.2.5.3 IDCODE

The IDCODE instruction selects the ID register, and the system logic controls the I/O pins. This instruction is provided as a public instruction to allow the manufacturer, part number and version of a component to be determined through the TAP. Figure 71-9 on page 16 shows the ID register configuration.

Figure 71-9. Identification Register Configuration

31 28	27 22	21 17	16 12	11 1	0
Version Information	Customer Part Number			Manufacturer Identity	1
	Design Center Number	Core Number	Chip Derivative Number		
0001	001000	11010	0010	00000001110	1

One application of the ID register is to distinguish the manufacturer(s) of components on a board when multiple sourcing is used. As more components emerge which conform to the IEEE 1149.1 standard, it is desirable to allow for a system diagnostic controller unit to blindly interrogate a board design in order to determine the type of each component in each location. This information is also available for factory process monitoring and for failure mode analysis of assembled boards.

Motorola's Manufacturer Identity is 0b00000001110. The Customer Part Number consists of two parts: Motorola Design Center Number (bits 27:22) and a sequence number (bits 21:12). The sequence number is divided into two parts: Core Number (bits 21:17) and Chip Derivative Number (bits 16:12).

Once the IDCODE instruction is decoded, it will select the ID register which is a 32 Bit data register. Since the bypass register loads a logic 0 at the start of a scan cycle, whereas the ID register loads a logic 1 into its least significant bit, examination of the first bit of data shifted out of a component during a test data scan sequence immediate following exit from Test-Logic-Reset controller state will show whether such a register is included in the design. When the IDCODE instruction is selected, the operation of the test logic has no effect on the operation of the on-chip system logic as required by the IEEE 1149.1 standard.

71.2.5.4 HIGHZ

When HIGHZ is invoked, all output drivers, including the two-state drivers, are turned off (i.e., high impedance). The instruction selects the bypass register. The HIGHZ instruction also asserts internal reset for the Neptune system logic to force a predictable internal state while performing external boundary scan operations. In this mode, all internal pull-up resistors on all the pins (except for the TMS, TDI and TRST~ pins) will be disabled.

71.2.5.5 ENABLE_DSP_ONCE

Provided as a public instruction to allow the user to perform system debug functions. When the ENABLE_DSP_ONCE instruction is decoded the ONYXU TDI and TDO (o_tdo and o_tdi) pins are connected directly to the DSP OnCE registers. The particular DSP OnCE register connected between ONYXU TDI and TDO at a given time is selected by the DSP OnCE controller depending on the DSP OnCE instruction being currently executed. All communication with the DSP OnCE controller is done through the Select-DR-Scan path of the ONYXU JTAG TAP Controller.

71.2.5.6 ENABLE_MCU

The ENABLE_MCU is provided as a public instruction to allow the user to perform system debug functions. When the command is applied, the ARM and ONYXU instruction registers and data registers are placed in a serial chain. This will allow access to both TAP controllers.

71.2.5.7 DSP_DEBUG_REQUEST

The DSP_DEBUG_REQUEST is provided as a public instruction to allow the user to generate a debug request signal to the DSP Core. When the DSP_DEBUG_REQUEST instruction is decoded the onyx TDI and TDO pins are connected to the Instruction Registers. Due to the fact that in the capture-IR state of the TAP the OnCE status bits are captured in the Instruction shift register, the external JTAG controller must continue to shift-in the DSP_DEBUG_REQUEST instruction while polling the status bits that are shifted-out until the Debug Mode of operation is entered (acknowledged by the combination 11 on OS1-OS0). After the acknowledgment of the Debug Mode is received, the external JTAG controller must issue the ENABLE_DSP_ONCE instruction to allow the user to perform system debug functions.

71.2.5.8 LOAD_ABISTCR

The LOAD_ABISTCR instruction connects TDI to the DSP ABIST ABISTCR. Data will be shifted in during the SHIFT_DR state and will be updated on the UPDATE_DR state. TDO will shift out the previous value of ABISTCR during the SHIFT_DR state. Refer to the *ABIST Functional Specification* for more information on the ABISTCR.

NOTE:

LOAD_ABISTCR is only available in Onyx mode. Placing LOAD_ABISTCR into the IR during any other mode will result in the same as BYPASS.

71.2.5.9 LOAD_JTCR

The LOAD_JTCR instruction connects TDI and TDO to the JTAG Test Control Register (JTCR). Data will be shifted in during the SHFT_DR state and will be updated on the UPDATE_DR state. TDO will shift out the previous JTCR values.

NOTE:

LOAD_JTCR is only available in Onyx mode. Placing LOAD_JTCR into the IR during any other mode will result in the same as BYPASS.

71.2.5.10 BYPASS

The BYPASS instruction selects the single Bit bypass register as shown in Figure on page 18, and the system logic controls the I/O pins. This creates a shift-register path from SYSTDI to the bypass register and, finally, to SYSTDO, circumventing the TBD Bit boundary scan register. This instruction is used to enhance test efficiency when a component other than the Neptune becomes the device under test.

When the bypass register is selected by the current instruction, the shift-register stage is set to a logic zero on the rising edge in the Capture-DR controller state. Therefore, the first bit to be shifted out after selecting the bypass register will always be a logic zero.

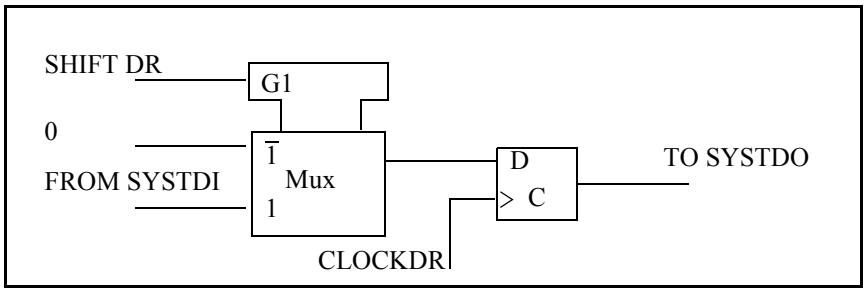


Figure 71-10. Bypass Register

71.2.6 JTAG Test Control Register (JTCR)

The JTCR was created to support the Neptune LTE DSP ABIST methodology. The JTCR is a internal data register to the SJC which can be accessed when the LOAD_JTCR instruction is located in the shift register. The JTCR can only be accessed in Onyx mode. Each bit will be updated on the UPDATE_DR state. JTCR resets on a TRST, POR, or entering the TEST LOGIC RESET state.

JTAG Test Control Register(JTCR)

IR =LOAD_JTCR

7	6	5	4	3	2	1	0
		sjc_sio_acs	sjc_gpio_bit_map_cnfg	sjc_gpio_prod_cnfg	dsp_bist en	dsp_bist_invoke	dsp bist reset	dsp bist release	dsp bist ret en
		rw	rw	rw	rw	rw	rw	rw	rw
		0	0	0	0	0	0	0	0

Table 71-3. JTCR Description

Name	Description	Settings
sjc_sio_acs Bit 6	BIST I/O serial access strobe **Note:** This bit is or'd with tcm_dsp_bist_bypass_mode	0 = disabled 1 = enabled
sjc_gpio_bit_map_cnfg Bit 6	GPIO configuration to BIST bit mapping mode	0 = disabled 1 = enabled
sjc_gpio_prod_cnfg Bit 5	GPIO configuration to BIST production mode	0 = disabled 1 = enabled
dsp_membist_en Bit 4	enable the dsp bist clock	0 = disabled 1 = enabled
dsp_membist_invoke Bit 3	start the dsp bist engine	0 = disabled 1 = enabled
dsp_membist_reset Bit 2	reset the dsp bist engine - must reset after enabled	0 = reset inactive 1 = reset active
dsp_membist_release Bit 1	release dsp self pause sequence in data retention algorithm	0 = disabled 1 = enabled
dsp_membist_retention_en Bit 0	enable dsp self pause sequence for data retention	0 = disabled 1 = enabled

71.2.7 Debug Mode

ARM can enter debug mode through three methods

1. Causing a breakpoint through the AWPT
2. Placing 33 bit in ARM JTAG scan chain1 high
3. Asserting $\overline{\text{ARM\_DE}}$ for at least four mcu clock cycles

Entering debug mode will cause the open drain pad $\overline{\text{ARM\_DE}}$ to assert for 3 ARM clock cycles.This will also enable ARM-ONYX serial chain mode if Onyx mode is currently selected.

ONYX can enter debug mode through three methods

1. Placing DSP_DEBUG_REQUEST in the DSP JTAG instruction register
2. Asserting $\overline{\text{DSP\_DE}}$ for at least six ONYX clock cycles
3. PDDM asserting $\overline{\text{pddm\_eg\_dreq}}$ for at least six ONYX clock cycles

Entering debug mode will cause the open drain pad $\overline{\text{DSP\_DE}}$ to assert for 3 ONYX clock cycles

To cause the dsp and mcu enter debug mode simultaneously, it is necessary to physically connect the two pads together. Since the two core (ONYX and ARM) clock run at different frequencies, it is necessary to place latching logic, so that when ONYX enters debug mode ARM can enter debug mode as well.

To leave debug mode see Chapter 71.3.2, “Example of Releasing the MCU and DSP from Debug Modes.”

The ARM core debug enable (A_DGBEN) will assert when one of the two conditions are met:

1. Debug mode for the ARM has been requested (A_DBGRQ asserts)
2. The TAP state machines have left the state TEST_LOGIC_RESET

After assertion A_DBGEN will stay asserted until $\overline{\text{SYNC\_CLEAR}}$ is asserted.

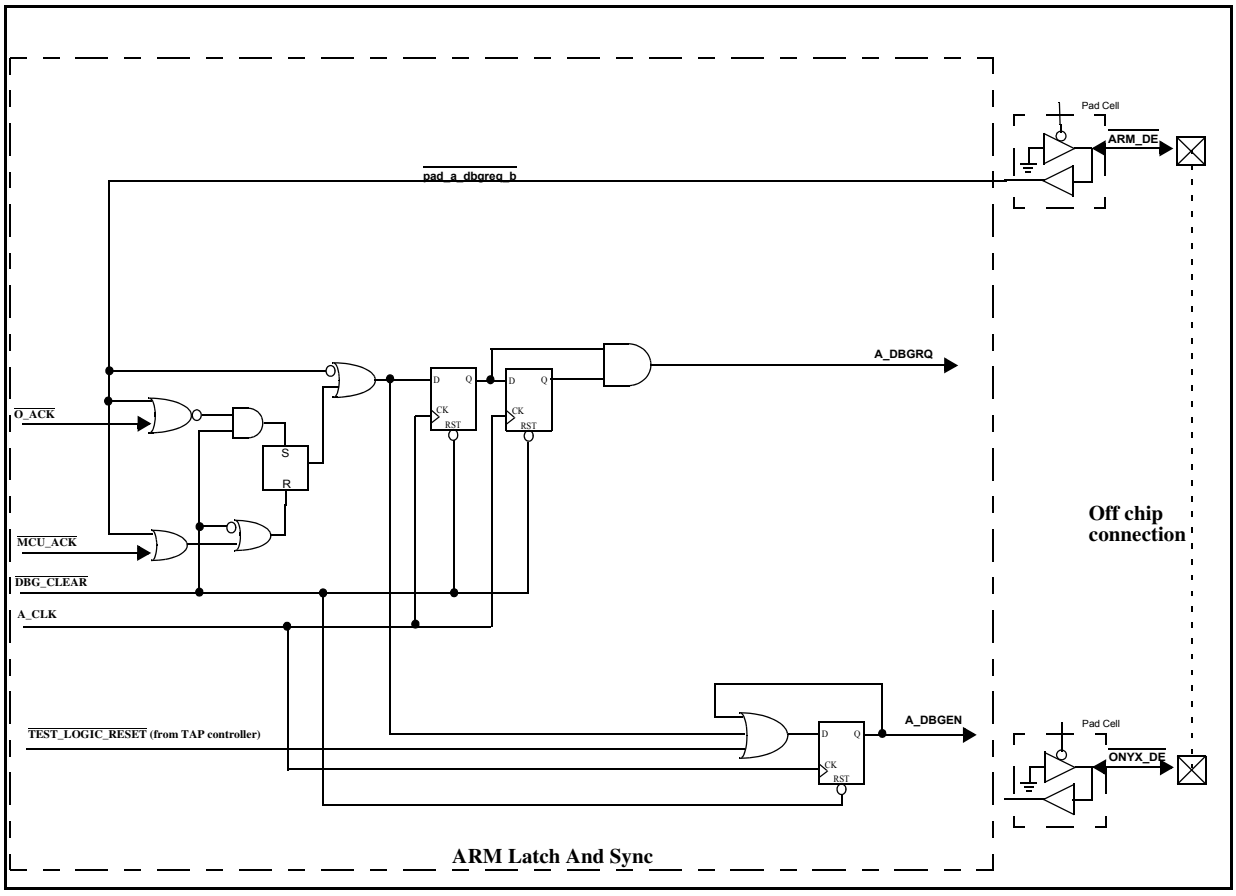


Figure 71-11. ARM and ONYX Latch and sync

71.3 Examples of JTAG instructions

71.3.1 Example of Entering ARM-Onyx serial chain via JTAG Control

Table 71-4. TMS Sequencing for entering ARM-Onyx serial chain via JTAG Control

step	TMS	ARM JTAG	Note
a	1	Test-Logic-Reset	
b	0	Run-Test/Idle	
c	1	Select-DR-Scan	
d	1	Select-IR-Scan	
e	0	Capture-IR	capture DSP core status bits
f	0	Shift-IR	the 4 bits of the JTAG ENABLE_MCU instruction (0b0011) are shifted into the DSP instruction register
g	0	Shift-IR	
h	0	Shift-IR	
i	0	Shift-IR	
j	1	Exit1-IR	At this point, the IR section of the DSP is ready to be loaded. The MCU tap controller shadow logic is ready to set the SYSTEM/$\overline{\text{JTAG}}$ signal.
k	1	Update-IR	Serial mode is enabled
l	0	Run-Test/Idle	Serial mode is enabled

NOTE:

When serial mode is enabled, the JTAG instruction register (IR) becomes the concatenation for the DSP's IR (4-bits) and the MCU's IR (4-bits). Further shifts into the JTAG instruction register should be 8-bits in length.

71.3.2 Example of Releasing the MCU and DSP from Debug Modes

This assumed that you are in a serial chain mode and all internal states have been restored both cores

Table 71-5. TMS Sequencing for Simultaneous Release of ARM ONYX Debug Mode via JTAG Control

step	TMS	JTAG	Note
a	1	Test-Logic-Reset	
b	0	Run-Test/Idle	
c	1	Select-DR-Scan	
d	1	Select-IR-Scan	
e	0	Capture-IR	
f	0	Shift-IR	
g-j	0	Shift-IR	the 4 bits of the SCAN_N (b0010)are shifted into the DSP + MCU instruction register
k-m	0	Shift-IR	the remaining 4 bits of the DPS instruction BYPASS(b1111) are shifted into the combined DSP + MCU IR's
n	1	Exit1-IR	
o	1	Update-IR	
p	1	Select-DR-Scan	
q	0	Capture-DR	
r	0	Shift-DR	
s-v	0	Shift-DR	4 bits (b0001) to the scan selection register, to select the scan chain 1 are shifted in
w	1	Exit1-DR	A single bit of data is placed in TDI for the DSP bypass register
v	1	Update-DR	
w	0	Run-Test/Idle	
x	1	Select-DR-Scan	
y	1	Select-IR-Scan	
z	0	Capture-IR	
aa	0	Shift-IR	
ab-ae	0	Shift-IR	the 4 bits of the INTEST(b1100) are shifted into the DSP + MCU instruction register

Table 71-5. TMS Sequencing for Simultaneous Release of ARM ONYX Debug Mode via JTAG Control

step	TMS	JTAG	Note
af-ai	0	Shift-IR	the remaining 4 bits of the DSP instruction BYPASS(b1111) are shifted into the combined DSP + MCU IR's
aj	1	Exit1-IR	
ak	1	Update-IR	
al	1	Select-DR-Scan	
am	0	Capture-DR	
an	0	Shift-DR	
ao-bu	0	Shift-DR	33 bits (b0,he1a00000) to chain 1 are shifted in. Unlike normal JTAG, the order that the bits are shifted in is MSB to LSB. This represents the command MOV R0, R0 with the breakpoint not set. For more information refer to ARM7DTMI-S Technical Reference Manual, Appendix D (Debug in Depth)
bv	1	Exit1-DR	A single bit of data is placed in TDI for the DSP bypass register
bw	1	Update-DR	
bx	1	Select-DR-Scan	
by	0	Capture-DR	
bz	0	Shift-DR	
ca-dg	0	Shift-DR	33 bits (b1,he1a00000) to chain 1 are shifted in. Unlike normal JTAG, the order that the bits are shifted in is MSB to LSB. This represents the command MOV R0, R0 with the breakpoint set. This will allow the next assembler command to function in system mode. For more information refer to ARM7DTMI-S Technical Reference Manual, Appendix D (Debug in Depth)
dh	1	Exit1-DR	A single bit of data is placed in TDI for the DSP bypass register
di	1	Update-DR	
dj	1	Select-DR-Scan	
dk	0	Capture-DR	
dl	0	Shift-DR	

Table 71-5. TMS Sequencing for Simultaneous Release of ARM ONYX Debug Mode via JTAG Control

step	TMS	JTAG	Note
dm-es	0	Shift-DR	33 bits (b0,heafffffa) to chain 1 are shifted in. Unlike normal JTAG, the order that the bits are shifted in is MSB to LSB. This represents the command B -6 with the breakpoint not set. The value of branch varies with the number of instructions that were executed with the ARM in debug mode. For more information refer to ARM7DTMI-S Technical Reference Manual, Appendix D (Debug in Depth)
et	1	Exit1-DR	A single bit of data is placed in TDI for the DSP bypass register
eu	1	Select-DR-Scan	
ev	1	Select-IR-Scan	
ew	0	Capture-IR	
ex	0	Shift-IR	
ey-fb	0	Shift-IR	the 4 bits of the RESTART(b0100) are shifted into the DSP + MCU instruction register
fc-ff	0	Shift-IR	the remaining 4 bits of the DSP instruction ENABLE_ONCE(b0110) are shifted into the combined DSP + MCU IR's
fg	1	Exit1-IR	
fh	1	Update-IR	
fi	1	Select-DR-Scan	
fj	0	Capture-DR	
fk	0	Shift-DR	
fl	0	Shift-DR	A single bit of data is placed in TDI for the MCU bypass register
fm-ft	0	Shift-DR	Bits Read + Go + Exit + No Register Selected Are placed in the DSP OnCE module (b0,b1,b1,b11111)
fu	1	Exit1-DR	
fv	0	Run-Test/Idle	Both Cores Exit Debug mode

71.4 Serial ARM-ONYX TDI-TDO Waveform

This timing diagram shows the flow form the TDI pin to the TDO pin. The assumptions are that both cores are in bypass mode and both TAP controllers are in the SHIFT-DR state.

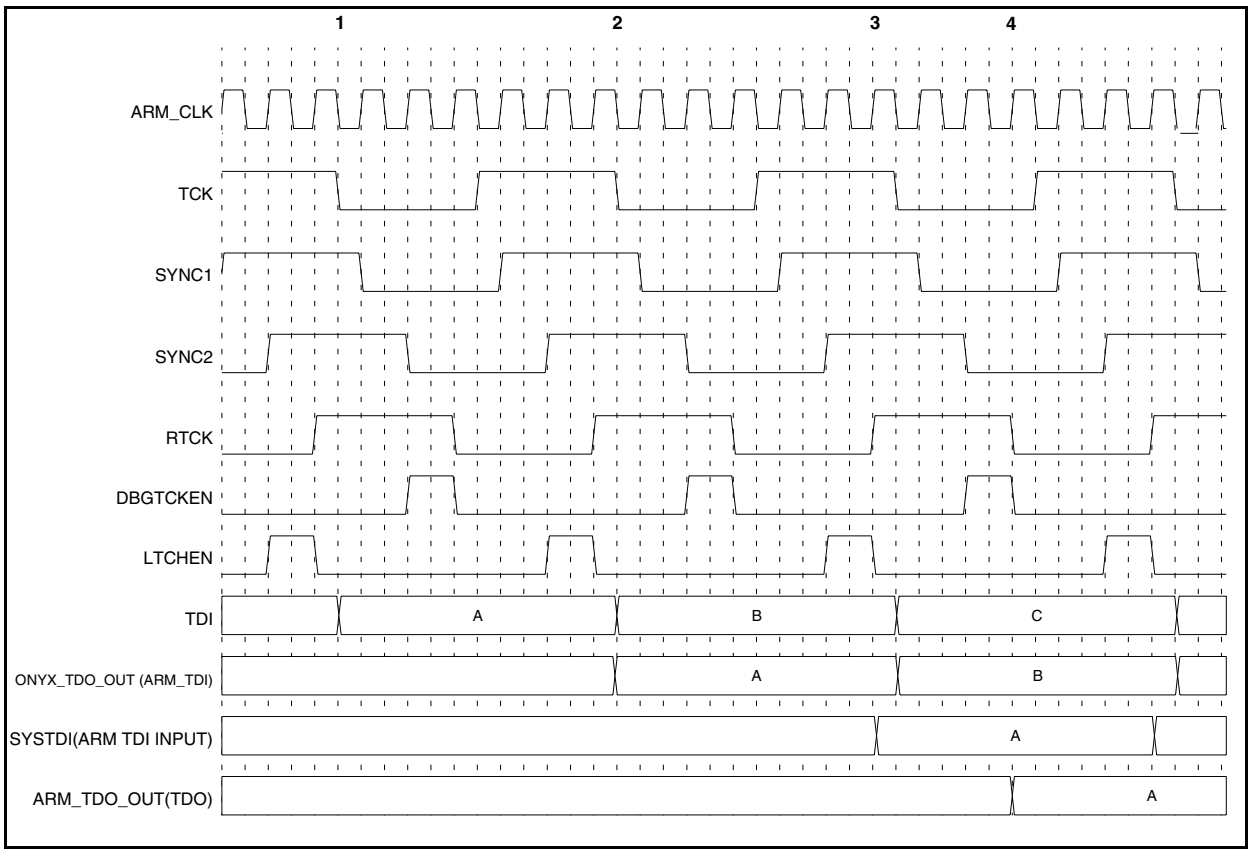


Figure 71-12. ARM-ONYX TDI-TDO Waveform

1. Data A is transmitted from the TDI pin on the falling edge of TCK
2. ONYX transmits data A to ARM TDI on a falling edge of TCK
3. Data A is synchronized and send to ARM TDI
4. ARM transmits data A to TDO on a falling edge of RTCK

71.5 Neptune JTAG Restrictions

The DSP Core features a low-power stop mode, which is invoked using an instruction called STOP. The interaction of the JTAG interface with low-power stop mode is as follows:

1. The TAP controller must be in the test-logic-reset state to either enter or remain in the low-power stop mode. Leaving the TAP controller test-logic-reset state negates the ability to achieve low-power, but does not otherwise affect device functionality.
2. The TCK input is not blocked in low-power stop mode. To consume minimal power, the TCK input should be externally connected to VCC or ground.
3. The TMS and TDI pins include on-chip pull-up resistors. In low-power stop mode, these two pins should remain either unconnected or connected to VCC to achieve minimal power consumption.
4. Since during STOP state all DSP Core clocks are disabled, the JTAG interface provides the means of polling the DSP Core status (sampled in the capture-IR state). The JTAG interface provides the software means of entering the Debug Mode by executing the DSP_DEBUG_REQUEST instruction.

71.6 Issues

The frequency of the DSP_CLK must be greater than the frequency of the MCU_CLK in order to have the latch and sync debug circuitry working properly.

Chapter 72
Analog Test Module (ANATEST)

Revision History Table

Version	Date	Author	Changes
0.0	03/08/02	Min Liao	Initial Release
1.0	03/25/02	Min Liao	1. Update spec with LTS terminology 2. Update figure 64-1, mux2 “0”->“1” 3. 64.1.2 level shifter: nominal voltage 1.5->1.575v, 2.5->2.775v
2.0	06/16/02	Min Liao	Major description update, added more functional description.
2.1	07/22/02	Min Liao	Added BIST issue for digital muxes
2.2	05/07/03		Updated for LTE specification release.

72.1 Overview

The primary purpose of Anatest is to bring out the signals from the analog blocks to two dedicated analog output pads for observability, debug and testing. This module is active during both the test mode and functional mode. The debugging capability can be also made available in non-test mode by driving the input data and controls from MQSPI block or TCM block.

As a secondary and independent purpose, various configuration bits may be selected and sent to the analog modules.

As shown in figure 64-1, this module makes use of the following sub-modules

1) an analog signal mux (mux1), output 1 differential mode signal pair to MUX2, among them 24 are from 12 analog modules, and 1 from saturation detection module since it only has 1 “det_test_mode_p”

2) an buffer selection block (mux2). It’s controlled by two encoded select lines, results in the following possible outputs:

a) unbuffered

b) buffered using tri-state buffer (Digital Buffer)

c) buffered using an opamp (Analog Buffer)

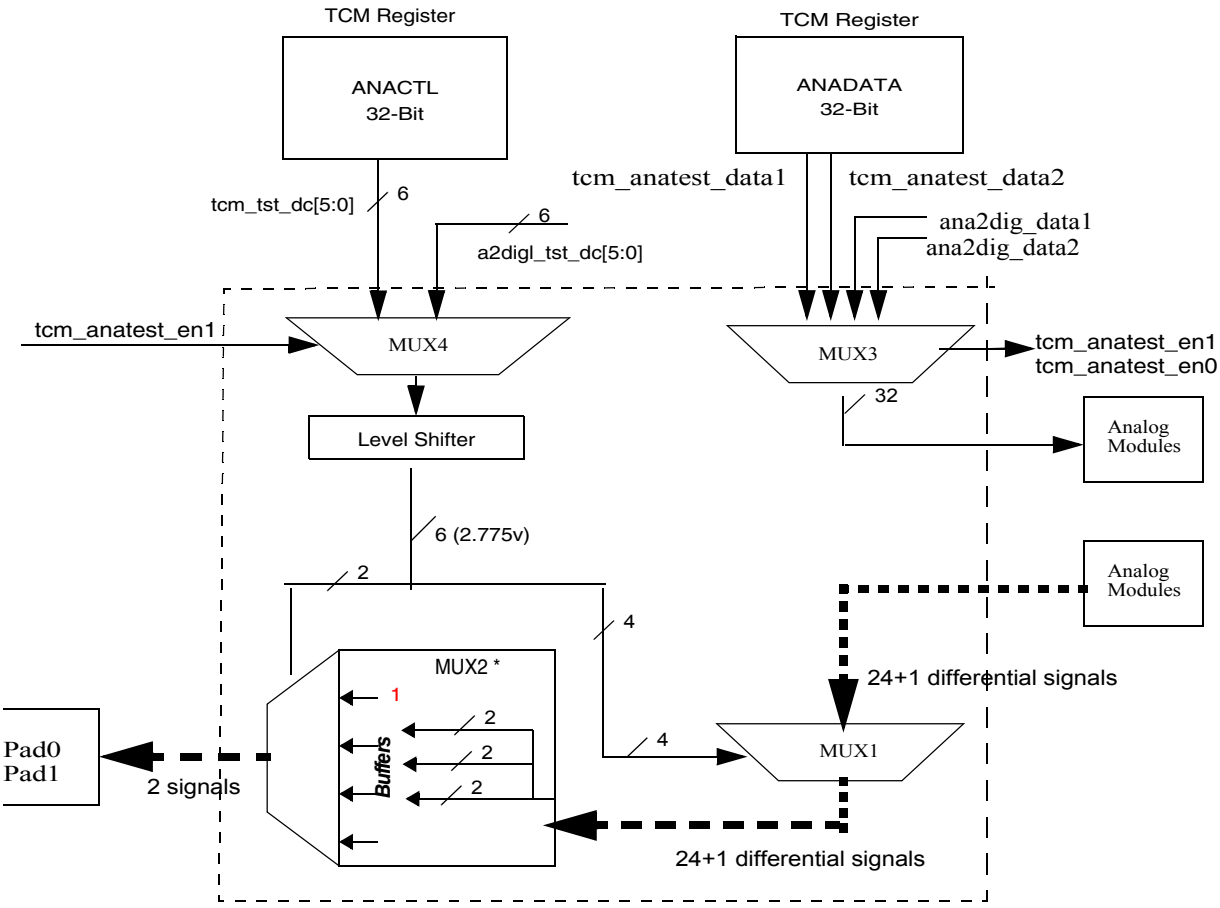
d) analog buffer with LVDD (1.575v) input

3) an digital mux which route configuration bit to analog modules (mux3)

4) selection line mux (mux4), generates the selection lines for mux1, mux2.

5) a (0, 1.575V) to (0, 2.775V) logic level shifter

6) two data registers (tcm_anatest_data1, tcm_anatest_data2) reside inside of TCM



* MUX2 chooses among internally buffered versions of one input

Figure 72-1. Anatest Block Diagram

72.2 Analog Signal Selection and Routing (Primary Function)

12 Analog differential mode signal pairs and 1 saturation detection module output are driven to the input of MUX1 from various analog modules on the chip. MUX1 then selects one signal pair among these. Because the signals are differential mode there are two wires associated with each --by convention we refer to one wire as positive (P), and the other negative(N). The output from MUX1 goes to a block called MUX2 which outputs the signal pairs to the pads. In MUX2, the “11” selection is conveniently used to determine the offset value generated between LVDD and the output pads. LVDD is acting as a reference voltage and whole purpose is to test the opamp.

72.3 Digital Signal Selection and Routing (Secondary Function)

As a secondary function, the ANATEST block uses MUX3 to select configuration bits to be routed to the analog blocks. These bits come from either a TCM register or from A2DGL depending upon the MUX3 select line.

The MUX3 and MUX4 allow A2DIGL module send some configuration bits to analog modules. tcm_anatest_en1 and tcm_anatest_en0 are the two control bits which are used to select from where the data come. tcm_anatest_en1 is module select and tcm_anatest_en0 is data select.

Table 72-1.
Table 72-2.

Data Selected	tcm_anatest_en1 (module)	tcm_anatest_en0 (data)
ana_data1 (TCM)	0	0
ana_data2 (TCM)	0	1
ana2dig_data1 (A2DIGL)	1	0
ana2dig_data2 (A2DIGL)	1	1

In Figure64-1, analog signals are shown in bold dashed lines. The various external analog modules are lumped together and named “Analog Modules'. These analog modules could be different modules from those we mentioned in 64.2 or there could be some overlap.

ana2dig_data1 and ana2dig_data2 are buses from individual inputs from A2DIGL.

ana2dig_data1 = {a2digl_tx_syn, a2digl_pac, a2digl_cmon_camp, a2digl_tunec, a2digl_voc,a2digl_det_test_mode,a2digl_rxafe}

ana2dig_data2 = {4’b0, a2digl_reg,a2digl_gpadc}

72.4 Analog Muxes

Mux1 selects (12:1) one output from 12 Analog blocks. Each block has two outputs, a positive and a negative. These go to the MUX2 before they are brought out to the pads.

The MUX2 is controlled by two control bits which allow the output:

1. to be unbuffered (Unbuffered)
2. to be buffered using tri-state buffer (Digital buffer)
3. to be buffered using an opamp (Analog buffer)
4. analog buffer with LVDD (1.575V) input

The four Muxes and level shifter will physically reside as close to the analog blocks as possible.

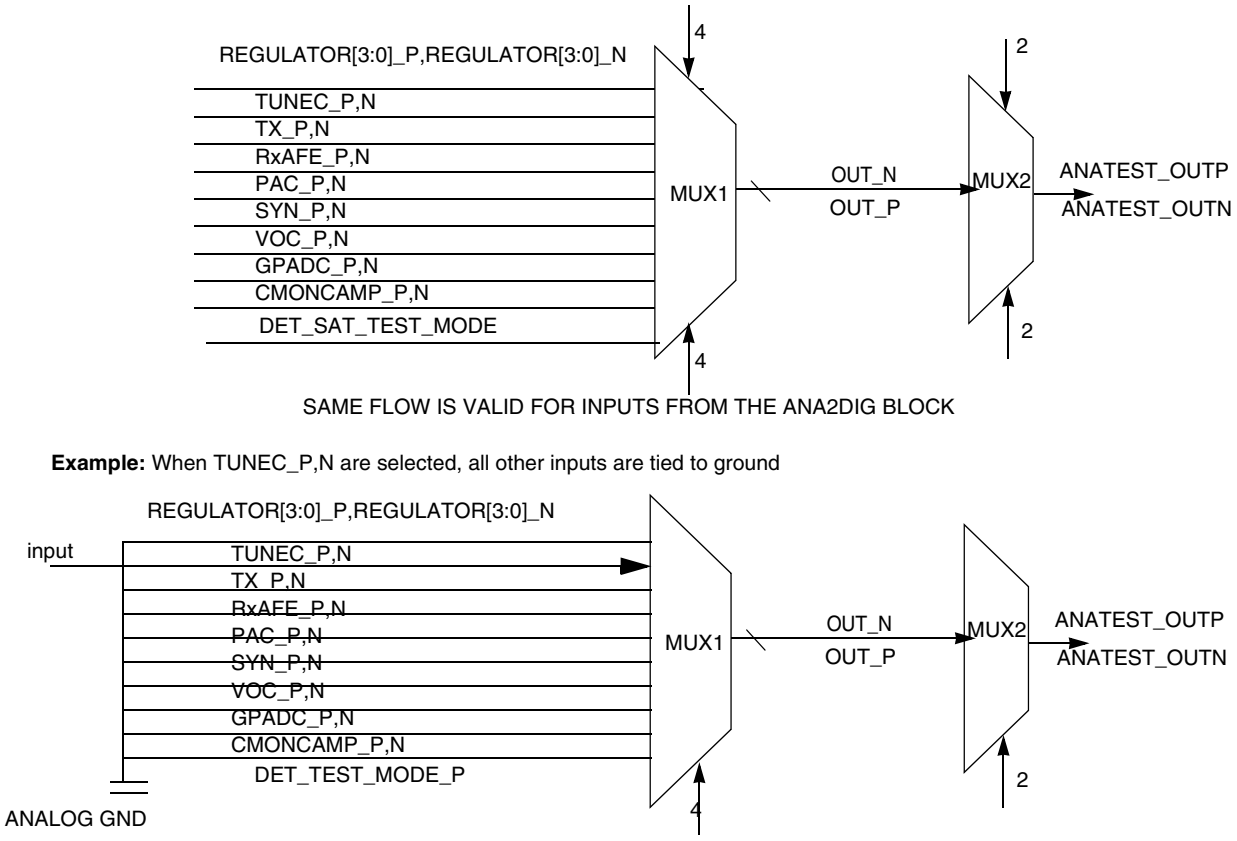
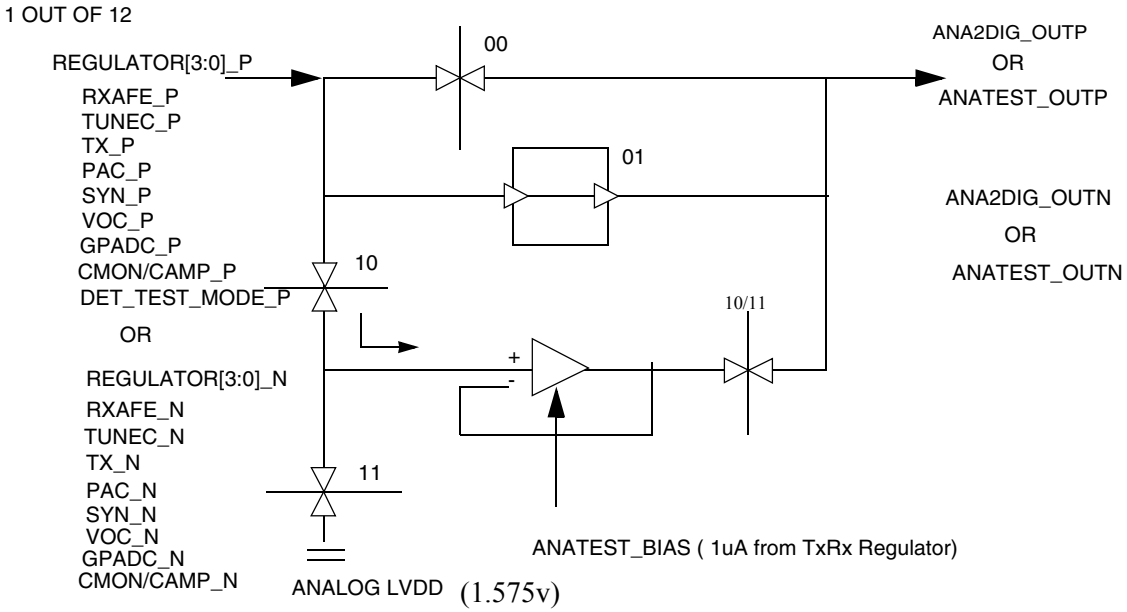


Figure 72-2. MUX Diagram



Note: Unselected inputs should be grounded.

Figure 72-3. MUX2

72.4.1 Analog Mux1

Table 72-3. Analog Mux1

TST_DC3	TST_DC2	TST_DC1	TST_DC0	Test_output Select
0	0	0	0	RX_AFE_P,N
0	0	0	1	TUNEC_P,N
0	0	1	0	TX_P,N
0	0	1	1	PAC_P,N
0	1	0	0	SYN_P,N
0	1	0	1	VOC_P,N
0	1	1	0	GPADC_P,N
0	1	1	1	CMONCAMP_P,N
1	0	0	0	REGUL_TXRX_TEST_OUTP,N
1	0	0	1	UNUSED
1	0	1	0	REGUL_CODEC_TEST_OUTP,N
1	0	1	1	REGUL_SYNTH_TEST_OUTP,N
1	1	0	0	REGUL_CORE_TEST_OUTP,N
1	1	0	1	DET_TEST_MODE_P
1	1	1	0	UNUSED
1	1	1	1	UNUSED

72.4.2 ANALOG Mux2

Table 72-4. Analog Mux2

TST_DC5	TST_DC4	Output_type Select
0	0	Unbuffered
0	1	Digital Buffer
1	0	Analog Buffer
1	1	analog buffer with LVDD (1.575V) input

72.5 Digital Muxes

Anatest gets two select lines tcm_anatest_en1 and tcm_anatest_en0 from TCM. The logic are implemented inside of TCM.

tcm_anatest_en1 = tcm_bist_mode_en | tm_anatest_en1

tcm_anatest_en0 = a2di_acc_ref 0 | tm_anatest_en0

72.5.1 Functional Mode

In functional mode, tcm_bist_mode_en is low, and tcm_anatest_en1 is totally controlled by tm_anatest_en1

tcm_anatest_en0 is controlled by a2di_acc_ref 0 ORed with tm_anatest_en0

72.5.2 BIST Mode

In TCM, tcm_bist_mode_en is ORed of all bist modes. So, as long as the chip gets into BIST mode

1. tcm_bist_mode_en =1

2. all registers are cleared, tm_anatest_en0 is low

In this case, tcm_anatest_en1 always stay high and A2DIGL is selected by requirement.

tcm_anatest_en0 can select DATA1 or DATA2 by programming a2di_acc_ref0 which comes from SPI group 10 a2di_sdata[64].

72.6 Level Shifter

Analog Mux is designed with Analog library that uses a 2.775v supply. Control signals coming from the control register are part of a digital block and use 1.575v supply.

A level shifter block will shift the ‘High’ level of 6 signals coming into the analog block from 1.575v to 2.775v.

The level shifter will physically reside as close to the analog blocks as possible.

72.7 Register Summary

These registers are inside TCM

KEY:

Always Reads One	Always Reads Zero	Read/ Write Bit	Read-Only Bit	Write-Only Bit	Write 1 to Clear	Self-Clear Bit	N/A
1	0	bit	bit	bit	bit	0	
					W1C	bit	

Table 72-5. Anatest Control Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ANA_CNTL ($2484_4010)								unused									
	R	unused	tm_anatest_en 11	tm_anatest_en 0	unused							TST_DC[5:0]					
	W																

Table 72-6. Anatest Data1 Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ANA_DATA1 ($$2484_4014)	R	TX_SYN[31:26]						PAC[25:21]					CMON/CAMP[20:16]				
	W																
	R	TUNEC[15:8]								VOC[7:6]		DET__TEST_MODE_P	RXAFE[4:0]				
	W																

Table 72-7. Anatest Data2 Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ANA_DATA2 ($$2484_4018)	R	unused				unused											
	W																
	R	unused				REG[7:0]								GPADC[3:0]			
	W																

72.8 Detailed Register Descriptions

ANACTL is a 32 bit control Read/Write register, generates the control bits for MUX1 and MUX2. This register physically resides in the Test Control Module (TCM). The control signals are level shifted before connecting to the two muxes.

ANA_DATA1 and ANA_DATA2 are 32 bit R/W registers. The bits are input bits, programmed in (by TCM in the master mode) to allow testing certain functionality of each block. The inputs are level shifted inside each block.

Unselected inputs to MUX1 are grounded from the corresponding analog modules.

The following figures and associated text give detailed descriptions of the various Anatest registers. The following definitions serve as a key for these figures:

- **Grey bit**: unimplemented bit. Always reads as zero; Writing has no effect.
- **TYPE**: Type of register bit. Defines register bit's behavior. Possible values:
  - **r**: read only. Writing this bit has no effect.
  - **w**: write only.
  - **rw**: Standard read/write bit. Only software can change bit's value (other than a hardware reset).
  - **rwm**: A read/write bit that may be modified by a hardware in some fashion other than reset.
  - **w1c**: A status bit that can be read, and it cleared by writing a logic 1.
  - **slfclr**: Self clearing bit. Writing a one has some effect on module, but it always reads as 0.
- **RESET**: Gives the reset value of the bit. Possible values:
  - **0**: Will reset to a logic 0
  - **1**: Will reset to a logic 1
  - **?**: The reset state is unknown.
  - **u**: Unaffected by reset

ANA_CNTL	Control Register															$2484_4010
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
	unused															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	unused	tm_anatest_en11	tm_anatest_en0	unused							TST_DC[5:0]					
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 72-8. ANA_CTL Description

Name	Description	Settings
Bit 31-15	Unused	N/A
TM_ANATEST_EN1 Bit 14	**Anatest Enable —** Select between TCM and A2DIGL	0=select TCM 1=select A2DIGL
TM_ANATEST_EN0 Bit 13	**Anatest Enable --** Select between data1 and data2	0=select DATA1 1=select DATA2
Bits12-6	**Unused**	0=disable 1=enable
TST_DC[5:4] Bits5-4	**Anatest Control MUX2 —** Select one set of signals from either TCM or A2DIGL.	See the MUX2 table.
TST_DC[3:0] Bits 3-0	**Anatest Control MUX1—** Select one out of twelve outputs from analog modules	See the MUX1 table.

ANA_DATA1	Data Register															$2484_4014
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
	TX_SYN[31:26]						PAC[25:21]					CMON/CAMP[20:16]				
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	TUNEC[15:8]								VOC[7:6]		DET	RXAFE[4:0]				
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0

Table 72-9. Data Register Descriptions

Name	Description	Settings
RXAFE[4:0] Bits 4-0	**Receiver Analog Front End —** Input bits.	0 = disable 1 = asserted
DET_TEST_MODE_P Bit 5	**Saturation detection bit**	0 = disable 1 = asserted
VOC[7:6] Bits 7-6	**Voice Codec & Interface —** Input bits.	0 = disable 1 = asserted
TUNEC[15:8] Bits 15-8	**Tuning Circuit —** Input bits.	0 = disable 1 = asserted
CMON/CAMP[20:16] Bits 24-20	**System Clock Monitor/Clock Amplifier —** Input bits.	0 = disable 1 = asserted
PAC[25-21] Bits 25-21	**Power Amplifier Control DAC/ADC —** Input bits.	0 = disable 1 = asserted
TX_SYN[31-26] Bits 31-26	**Transmitter & Interface —** Input bits.	0 = disable 1 = asserted

ANA_DATA2	Data Register															$2484_4018
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT16
	unused															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	unused				REG[7:0]								GPADC[3:0]			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 72-10. Data Register Descriptions
Table 72-4.

Name	Description	Settings
Bits [31:16]	unused	N/A
Bits [15:12]	unused	N/A
REG[7:0] Bits 11-4	**REG -** Input bits.	0 = disable 1 = asserted
GPADC[3:0] Bits 3-0	**GPADC**-Inputbits.	0 = disable 1 = asserted

72.9 Pin List

Table 72-5. ANATEST Pin List

Pin Name	Direction	Description
tcm_tst_dc[5:0]	input	control bits from TCM
tcm_anatest_data1[31:0	input	input from TCM data register1
tcm_anatest_data2[15:0]	input	input from TCM data register2
tcm_anatest_en1	input	enable bit1 from TCM
tcm_anatest_en0	input	enable bit0 from TCM
a2digl_tst_dc[5:0]	input	control bits from a2digl
a2digl_det_test_mode	input	saturation detection bit.
a2digl_rxafe[4:0]	input	input from a2digl
a2digl_voc[1:0]	input	input from a2digl
a2digl_tunec[7:0]	input	input from a2digl.
a2digl_cmon_camp[4:0]	input	input from a2digl.
a2digl_pac[4:0]	input	input from a2digl
a2digl_reg[7:0]	input	input from a2digl.
a2digl_gpadc[3:0]	input	input from a2digl.
a2digl_tx_syn[5:0]	input	input from a2digl.
rxafe_p	input	test bit from rxafe.
rxafe_n	input	test bit from rxafe.
tunec_p	input	test bit from tunec
tunec_n	input	test bit from tunec
pac_p	input	test bit from pac
pac_n	input	test bit from pac
syn_p	input	test bit from syn
syn_n	input	test bit from syn
voc_p	input	test bit from voc
voc_n	input	test bit from voc
tx_p	input	test bit from tx
tx_n	input	test bit from tx
gpadc_p	input	test bit from gpadc

Table 72-5. ANATEST Pin List

Pin Name	Direction	Description
gpadc_n	input	test bit from gpadc
cmon_camp_p	input	test bit from cmon_camp
cmon_camp_n	input	test bit from cmon_camp
regul_txrx_test_outp	input	test bit from regulator_txrx
regul_txrx_test_outn	input	test bit from regulator_txrx
regul_codec_test_outp	input	test bit from regul_codec
regul_codec_test_outn	input	test bit from regul_codec
regul_synth_test_outp	input	test bit from regulator_synth
regul_synth_test_outn	input	test bit from regulator_synth
regul_core_test_outp	input	test bit from regulator_core
regul_core_test_outn	input	test bit from regulator_core
det_test_mode_p	input	test bit from saturation detection module
regul_txrx_anatest_ibias	input	bias input for analog circuits
scan_mode	input	scan mode
scan_en	input	scan_en
test_reset	input	test_reset
test_clk	input	test_clk
anatest_sdi_1	input	scan data input
anatest_sdo_1	output	scan data output
anatest_out0	output	output to pad->anatest_n
anatest_out1	output	output to pad->anatest_p
anatest_reg[7:0]	output	output to regulator
anatest_gpadc[3:0]	output	output to gpadc
anatest_det_test_mode	output	output to saturation detection module
anatest_rxafe[4:0]	output	output to rxafe
anatest_voc[1:0]	output	output to voc
anatest_tunec[7:0]	output	output to tunec
anatest_cmoncamp[4:0]	output	output to cmon_camp
anatest_pac[4:0]	output	output to pac

Table 72-5. ANATEST Pin List

Pin Name	Direction	Description
anatest_tx_syn[5:0]	output	output to tx

In Figure64-1, ana2dig_data1 and ana2dig_data2 are buses from individual inputs from ana2dig block. ana2dig_data1 = {a2digl_tx_syn, a2digl_pac, a2digl_cmon_camp, a2digl_tunec, a2digl_voc,a2digl_det_test_mode,a2digl_rxafe} ana2dig_data2 = {4'b0, a2digl_reg,a2digl_gpadc}

Chapter 73
Input/Output Pad (PADIO)

Revision History Table

Revision	Date	Author	Changes
0	03/25/02	Geoffrey Hall	Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	08/01/02	Mark Babcock, Shannon Osgood	Updated per DDTS# DSPhl14078. Update table 73-4.
0.2	05/07/03		Updated for LTE specification release.

73.1 Introduction

This section describes the requirements of the I/O pads for the Neptune IC products processed in HiP7 technology. The overall goal is the definition of a minimum number of pad variations used on the Neptune product derivatives.

The description of the requirements is summarized as follows:

- Functional requirements
- Power rails, options, and control signals
- Switching and electrical characteristics
- Layout considerations

73.2 Functional Requirements

The features of the I/O pad module include the following:

- Integration-level selectable CMOS or open drain output type
- Three integration-level selectable output drive strengths - standard, high, and super high drive strength, nominally 4 mA, 8 mA, 16 mA, more or less, depending on operating conditions
- A maximum drive signal overrides 4 mA or 8 mA selections boosting output to 16 mA
- Integration-level selectable pull-up / pull-down weak MOSFETs approximately equivalent to pull up resistors of 22kOhms, 47kOhms, or 69kOhms or pull down resistor of 69kOhms
- Integration-level pull enable signal to disable the pull-up/pull-down capability
- Integration-level selectable active high or active low sense of the output enable signal
- Pad-keeper structure at the input path
- Integration-level selectable input or output functionality
- Integration-level selectable CMOS input or Schmitt trigger input
- Level shifter to convert level of signals from core (1.5 to 1.65 V) to output digital level (1.8 to 3.0 V)
- ESD protection

The definition of integration-level selectable is either via-metal interconnection at the top level chip to VDD/VSS or under software/hardware control. Neptune has requirements for GPIO and dedicated I/O to be configurable by application software and/or hardware. The remaining requirements are for testing purposes.

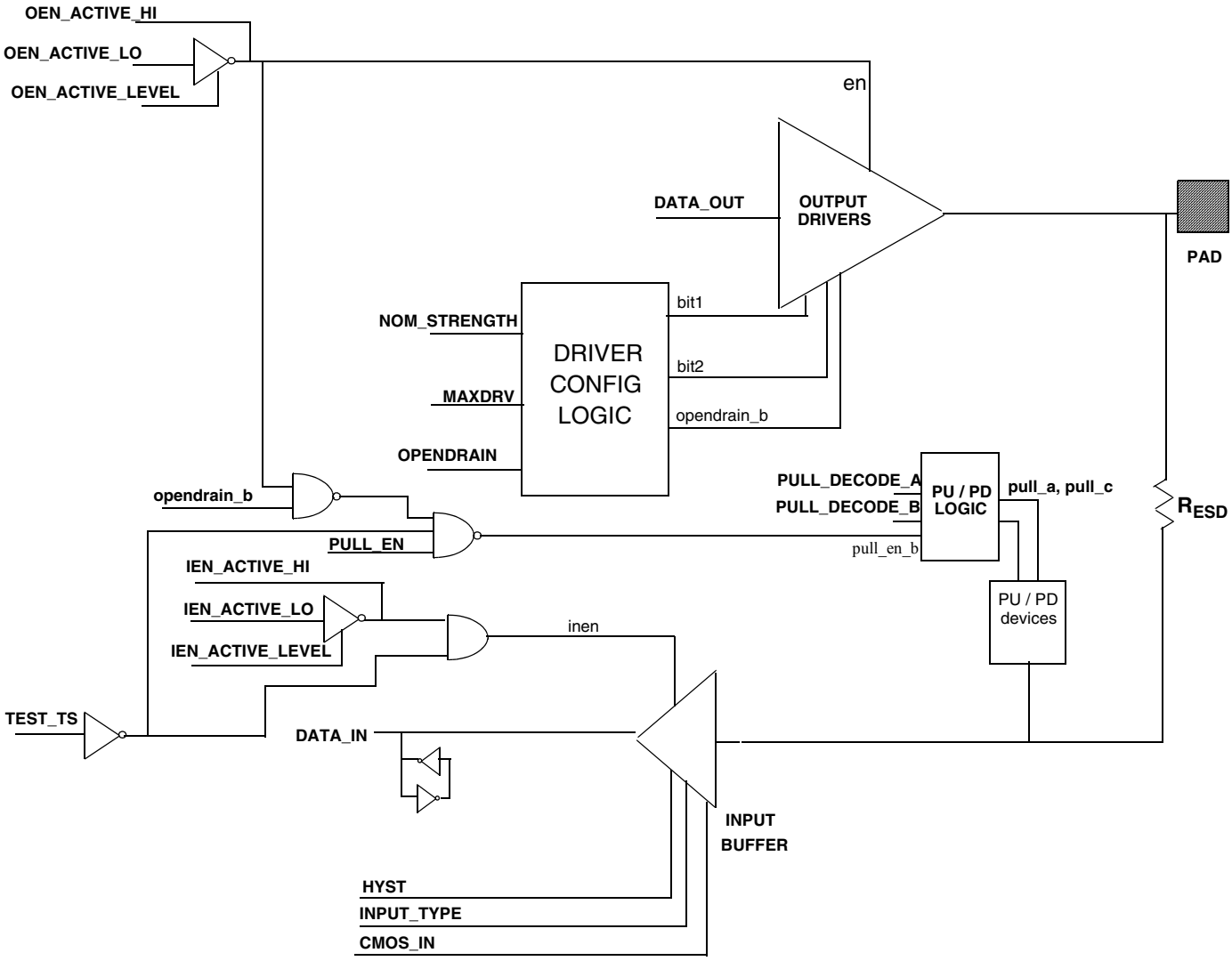


Figure 73-1. Schematic Diagram of the I/O Cell

73.3 PADIO Module Pin List

Table 73-1 lists the power rails and control signals associated with the proposed I/O buffer module.

Table 73-1. PADIO Module Pin List

Signal	Direction	Description
V_{DD}	input	core power rail 1.5 - 1.65 V
VSS	input	core ground
V_{DDS}	input	power rail for output driver 1.8 - 3.0 V (except RTC output driver supply is 1.5 - 1.65 V)
OVSS	input	ground for output driver
OEN_ACTIVE_LEVEL	input	integration-level option to select either active high or active low OEN OEN_ACTIVE_LEVEL=1: OEN is active low OEN_ACTIVE_LEVEL=0: OEN is active high See Section 73.6, “Chip Integration-Level Connections.”
OEN_ACTIVE_LO	input	integration-level connection for active low OEN If the module ouput enable is active high, OEN_ACTIVE_LO needs to be tied to ground. In that case the OEN_ACTIVE_HI connection is used as the output enable control signal to the pad cell. See Section 73.6, “Chip Integration-Level Connections.”
OEN_ACTIVE_HI	input	integration-level connection for active high OEN If the module output enable is active low, OEN_ACTIVE_HI needs to be left unconnected. In that case the OEN_ACTIVE_LO connection is used as the output enable control signal to the pad cell. See Section 73.6, “Chip Integration-Level Connections.”
IEN_ACTIVE_LEVEL	input	integration-level option to select either active high or active low IEN IEN_ACTIVE_LEVEL=1: IEN is active low IEN_ACTIVE_LEVEL=0: IEN is active high See Section 73.6, “Chip Integration-Level Connections.”
IEN_ACTIVE_LO	input	integration-level connection for active low IEN If the module input enable is active high, IEN_ACTIVE_LO needs to be tied to ground. In that case the IEN_ACTIVE_HI connection is used as the input enable control signal to the pad cell. See Section 73.6, “Chip Integration-Level Connections.”
IEN_ACTIVE_HI	input	integration-level connection for active high IEN If the module input enable is active low, IEN_ACTIVE_HI needs to be left unconnected. In that case the IEN_ACTIVE_LO connection is used as the input enable control signal to the pad cell. See Section 73.6, “Chip Integration-Level Connections.”
PULL_EN	input	signal to enable the PU/PD capability PULL_EN=0: PU/PD capability disabled PULL_EN=1: PU/PD capability enabled

Table 73-1. PADIO Module Pin List (Continued)

Signal	Direction	Description
PULL_DECODE_A PULL_DECODE_B	input	integration-level option bits to select PU or PD, and to control the PU strength A,B=0,0: 69KOhm PD A,B=0,1: 69KOhm PU A,B=1,0: 47KOhm PU A,B=1,1: 22KOhm PU For these two option bits ONLY, a logic '1' is VDDS, rather than VDD.
OPENDRAIN	input	signal to select between open drain or CMOS output OPENDRAIN=1: output is open drain OPENDRAIN=0: output is CMOS
NOM_STRENGTH	input	integration-level option to select standard or high pad drive strength NOM_STRENGTH=1: high drive strength NOM_STRENGTH=0: standard drive strength
MAXDRV	input	signal that selects test drive strength at every pad when in TEST mode MAXDRV=1: super drive strength MAXDRV=0: strength is nominal (standard or high, determined by NOM_STRENGTH)
PAD	input/output	I/O to the external world
DATA_OUT	input	data coming from the core into the pad
DATA_IN	output	data coming out of the pad into the core
ESD_RC	input	Signal for ESD trigger
TEST_TS	input	signal to control the input enable and PU/PD signals for leakage testing
HYST	input	integration-level connection to select Schmitt trigger input See Section 73.6, "Chip Integration-Level Connections."
CMOS_IN	input	integration-level connection to select CMOS input See Section 73.6, "Chip Integration-Level Connections."
INPUT_TYPE	input	integration-level connection to select CMOS or Schmitt trigger input INPUT_TYPE connected to CMOS_IN: CMOS input INPUT_TYPE connected to HYST: Schmitt trigger input See Section 73.6, "Chip Integration-Level Connections."

73.4 Electrical Requirements

This section defines the electrical requirements for the Neptune I/O pads. Temperatures used in these simulations, in degrees C: COLD=-20, ROOM=25, HOT=110

73.4.1 DC Characteristics

Table 73-2. Recommended Operating Conditions

Characteristic	Symbol	MIN	NOM	MAX	Unit
Core Supply Voltage (Output of Core Regulator)	V_{DD}	1.5	1.575	1.65	V
Supply Voltage (Level Shift I/O)[1]	V_{DDS}	1.8		3.0	V
Supply Voltage (RTC logic & I/O) [2]	LV_{DD}	1.5	1.575	1.65	V
Supply Voltage (FVDD Group logic & I/O)[3]	FV_{DD}	1.5	1.575	1.65	V
High-Level DC input voltage [4]	V_{IH}	$0.7V_{DD}$		V_{DDS}	V
Low-Level DC input voltage[4]	V_{IL}	0		$0.3V_{DD}$	V
Input Transition (t_r and t_f) time (10% to 90%)	t_t	0		25	ns
Operating ambient temperature range	T_A	-20		70	^0C
Junction temperature range	T_J	-20		110	^0C

1. AC performance of super high drive strength pads are spec'ed for MAX Output Supply Voltage (VDDS) limited to 1.93 V.
2. RTC output pad AC performance derated for voltages below 1.8 V.
3. FVDD group output pad AC performance derated for voltages below 1.8 V.
4. For level shifting inputs V_{IL} V_{IH} are determined relative to V_{DD}.

Table 73-3. Electrical Characteristics over Industrial Operating Conditions

Characteristic	Symbol	Test Conditions	MIN	NOM	MAX	Units
High-level output voltage[1] and [2]	V_{OH}	I_{OH} = max rated I_{OH} = 100 µA	V_{DDS}-0.5 V_{DDS}-0.2			V
Low-level output voltage[1] and [2]	V_{OL}	I_{OL} = max rated I_{OL} = 100 µA			0.5 0.2	V
Low-level input current	I_{IL}	$V_I = V_{SS}$			+/- 1	uA
High-level input current	I_{IH}	$V_I = V_{DD}$			+/- 1	uA
3-state output Hi-Z current	I_{OZ}	$V_I = V_{DD}$ or V_{SS}			+/- 20	uA
Hysteresis	V_{HYS}		0.25			V
Hysteresis input trip point rising	V_{TP+}		$0.5V_{DD}$			V
Hysteresis input trip point falling	V_{TP-}				$0.5V_{DD}$	V

1. For level shifting outputs V_{OL} and V_{OH} are determined relative to V_{DDS}.
2. V_{OL} and V_{OH} are modeled with a constant current load on the output driver and measuring the DC pad voltage while driving high and driving low. Max rated I_{OH} and I_{OL} are 3 mA, 6 mA, and 12 mA for the standard high, and super high drive strength pads, respectively.

73.4.2 AC Characteristics

Table 73-4. General Operating Specifications

Characteristic	Symbol	Test Condition	MIN	NOM	MAX	Units
Output Pad Propagation delay (super high Drive)[1] and [2]	Pad_{propO}	25pF 50pF			5 7.6	ns
Output Pad Propagation delay (high Drive)[1]	Pad_{propO}	25pF 50pF			7.6 10	ns
Output Pad Propagation delay (standard Drive)[1]	Pad_{propO}	25pF 50pF			10 16	ns
Input Pad Propagation delay	Pad_{propI}	2pF			1	ns
I/O Pad Rise / Fall Times (super high Drive) [2]	Pad_{rf}	25pF 50pF	1.0 1.5		3.5 5.0	ns
I/O Pad Rise / Fall Times (high Drive)	Pad_{rf}	25pF 50pF	1.5 2.4		5.0 8.0	ns
I/O Pad Rise / Fall Times (standard Drive)	Pad_{rf}	25pF 50pF	2.5 4.0		8.0 14	ns
I/O Pad Slew Rate (super high Drive) [2]	Pad_{slew}	25pF 50pF	0.4 0.25		1.2 0.75	V/ns

1. Pad Propagation delays are spec'ed from 50% to V_{OL} or V_{OH}.
2. AC performance of super high drive strength pads are spec'ed for Max Output Supply Voltage (V_{DDS}) limited to 1.93 V.

73.5 Layout Considerations

The HiP7 pad cells will meet the 63um wire pitch rules.

73.6 Chip Integration-Level Connections

73.6.1 Enable Selectors

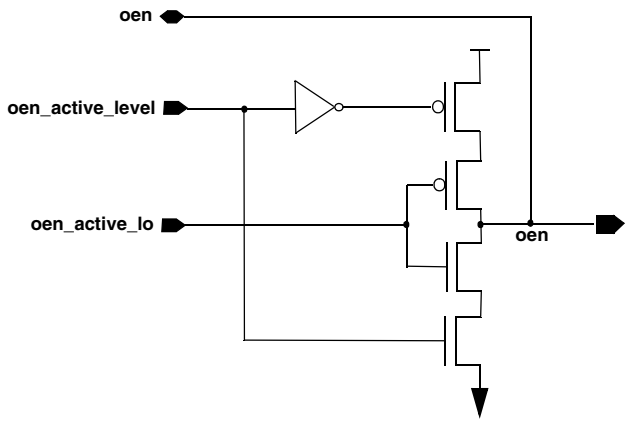


Figure 73-2. Output Enable Selector Circuit.

NOTE:

The following connections are made between the pad top-level schematic and this circuit: OEN_ACTIVE_LEVEL to oen_active_level, OEN_ACTIVE_HI to oen, and OEN_ACTIVE_LO to oen_active_lo

The connections to this circuit at chip integration level are determined on whether the output enable signal is active high or active low. The output enable signal from the module should be connected to OEN_ACTIVE_HI or OEN_ACTIVE_LO, depending on whether the output enable signal from the module is active high or active low, respectively. The OEN_ACTIVE_LEVEL signal is tied to either VSS or VDD, depending on whether the output enable signal from the module is active high or active low, respectively. If the output enable is active high, the OEN_ACTIVE_LO input should be connected to VSS.

The input enable selector circuit is identical to the output enable selector circuit, with the ien, ien_active_lo, and ien_active_level signals.

Figure 73-3 chip integration level connections to the output enable selector circuit for an active high output enable signal from the core module.

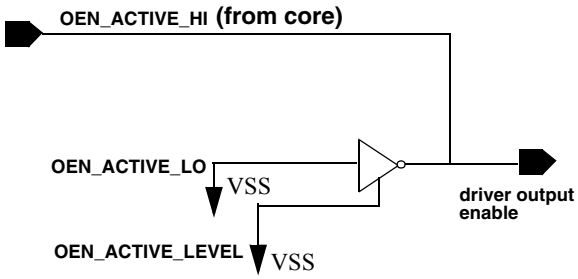


Figure 73-3. Output Enable Selector Circuit for an Active High Output Enable

NOTE:

An active high output enable signal from the module should be connected to the OEN_ACTIVE_HI pin. The OEN_ACTIVE_LO and OEN_ACTIVE_LEVEL pins should be tied to VSS.

Figure 73-4 shows chip integration level connections to the output enable selector circuit for an active low output enable signal from the core module.

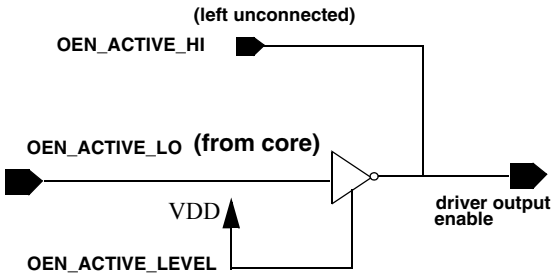


Figure 73-4. Output Enable Selector Circuit for an Active Low Output Enable

NOTE:

An active low output enable signal from the module should be connected to the OEN_ACTIVE_LO pin. The OEN_ACTIVE_HI pin should be left unconnected and OEN_ACTIVE_LEVEL should be tied to VDD.

73.6.2 Input Type Selector

Figure 73-5 is a detailed schematic of the input type selector circuit.

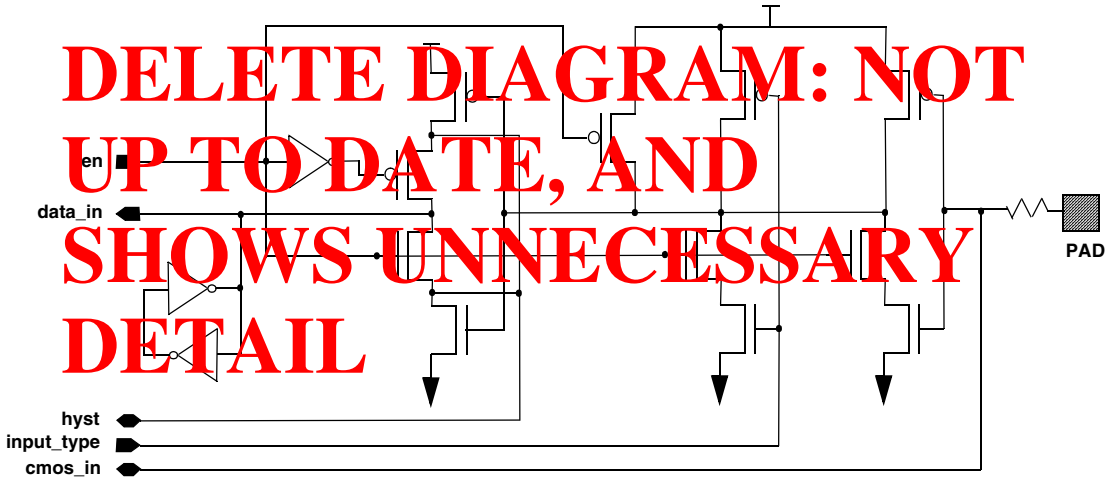


Figure 73-5. Input Type Selector Circuit

The connections to this circuit at chip integration level are determined on whether the input type is CMOS or hysteresis. If the input is CMOS, INPUT_TYPE should be connected to CMOS_IN and the HYST pin should be left unconnected. If the input is hysteresis type, INPUT_TYPE should be connected to HYST and the CMOS_IN pin should be left unconnected.

Figure 73-6 is a block of the input type selector circuit.

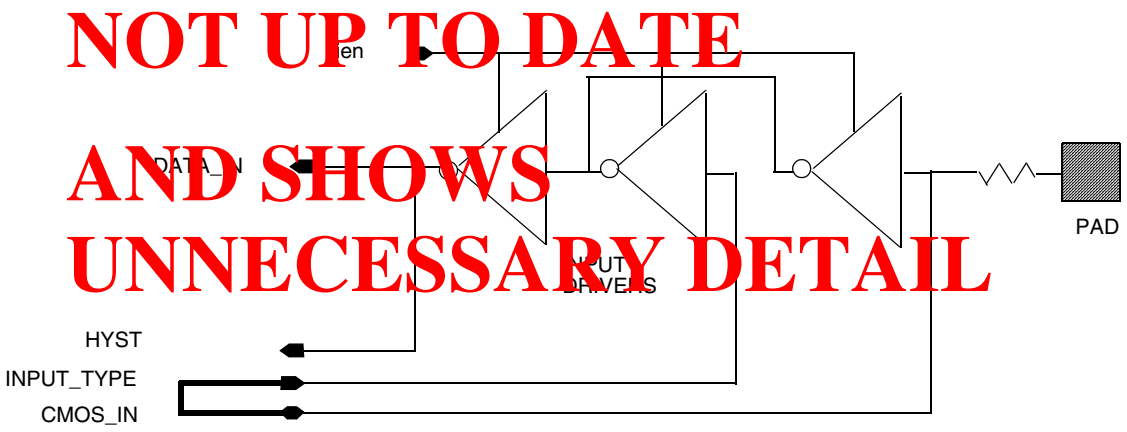


Figure 73-6. Chip Integration Level Connections to the Input Type Selector Circuit for CMOS Input.

NOTE:

INPUT_TYPE pin should be connected to CMOS_IN and the HYST pin should be left unconnected.

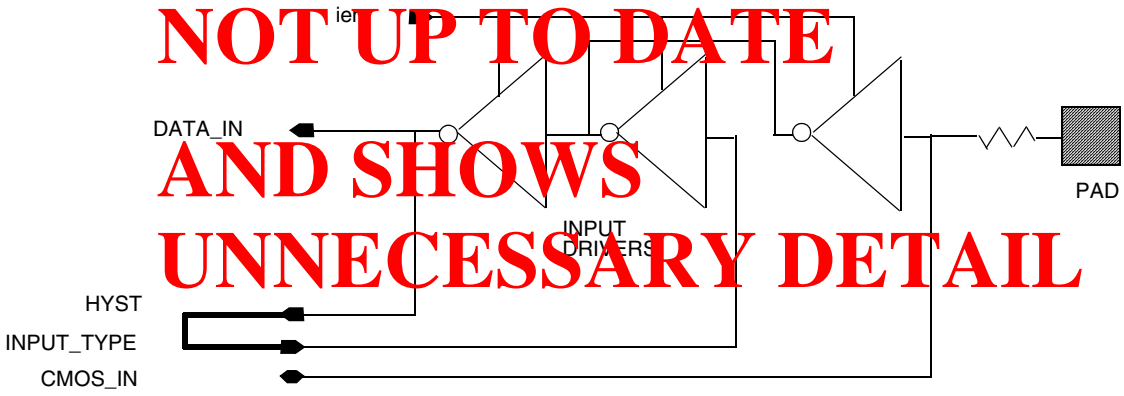


Figure 73-7. Chip Integration Level Connections to the Input Type Selector Circuit for Hysteresis Input.

NOTE:

INPUT_TYPE pin should be connected to HYST and the CMOS_IN pin should be left unconnected.

Chapter 74
Standard ESD Protection Networks (PADRING)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	5/03/02	Geoffrey Hall Shannon Osgood	Replaced all figures and added section 7.3 per Geoffrey Hall
0.2	05/07/03		Updated for LTE specification release.

74.1 Introduction

This chapter is a description of the ESD protection network for Neptune LTE.

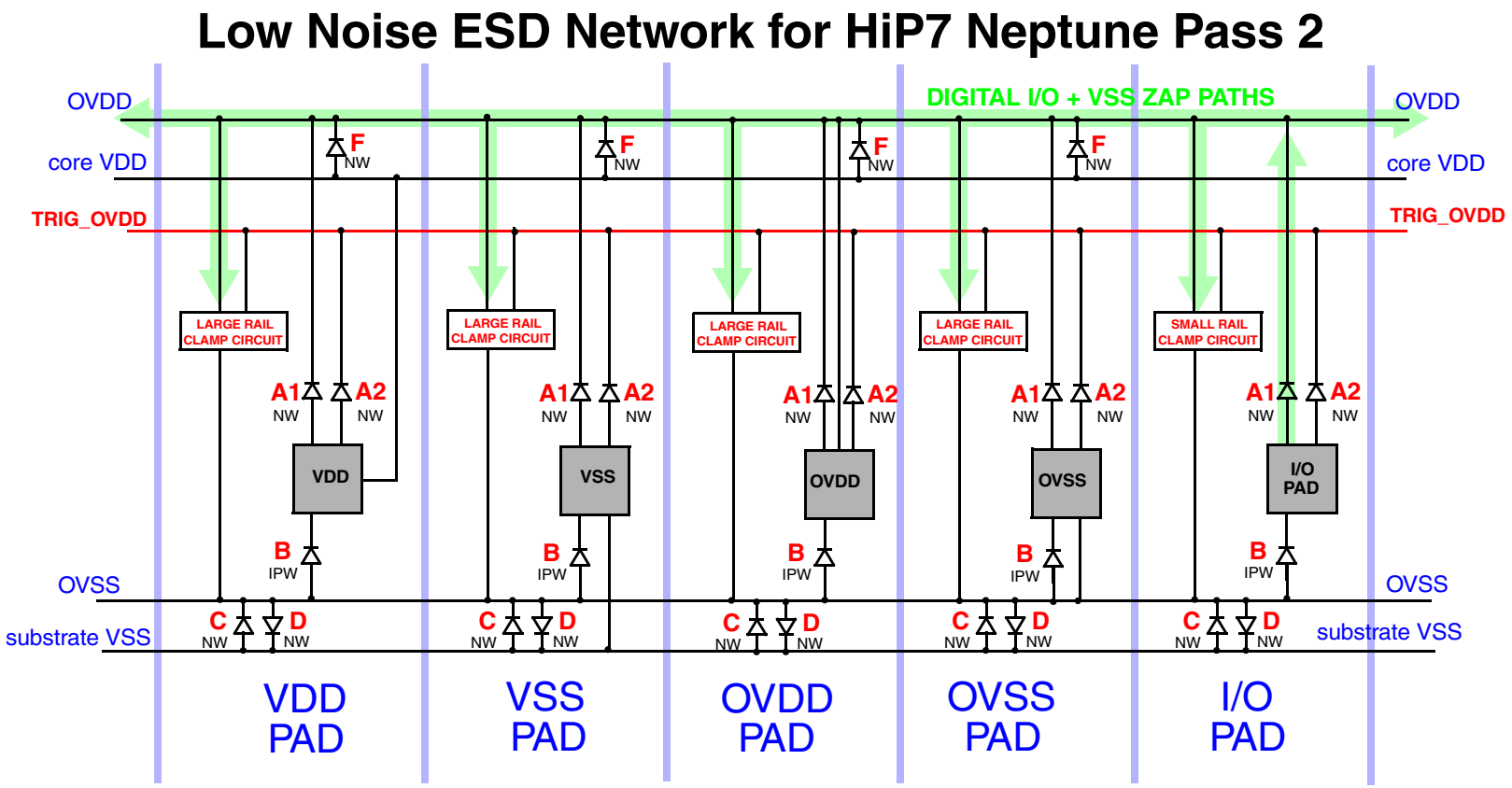


Figure 74-1. Low Noise ESD NEtwork for HiP7 Neptune Pass 2

ESD Network Changes from Pass1:

- Switched to boosted rail clamp network.
- Boosted clamps utilize A2 diode and small TRIG_OVDD bus to power trigger circuits. Allows a 50% clamp size reduction.
- ESD ground rail changed from VSS to OVSS. Only VSS rail rings the chip. C and D diodes protect OVSS to VSS.

Changes to Reduce Substrate Noise:

- No NWELL ballast resistor in output buffer.
- NMOS output buffer in IPW.
- B diode in IPW.
- Analog VSS two ESD diodes from noisy OVSS.

Low Noise ESD Network for HiP7 Neptune Pass 2

Each local OVDD bus segment serves as the positive ESD rail.

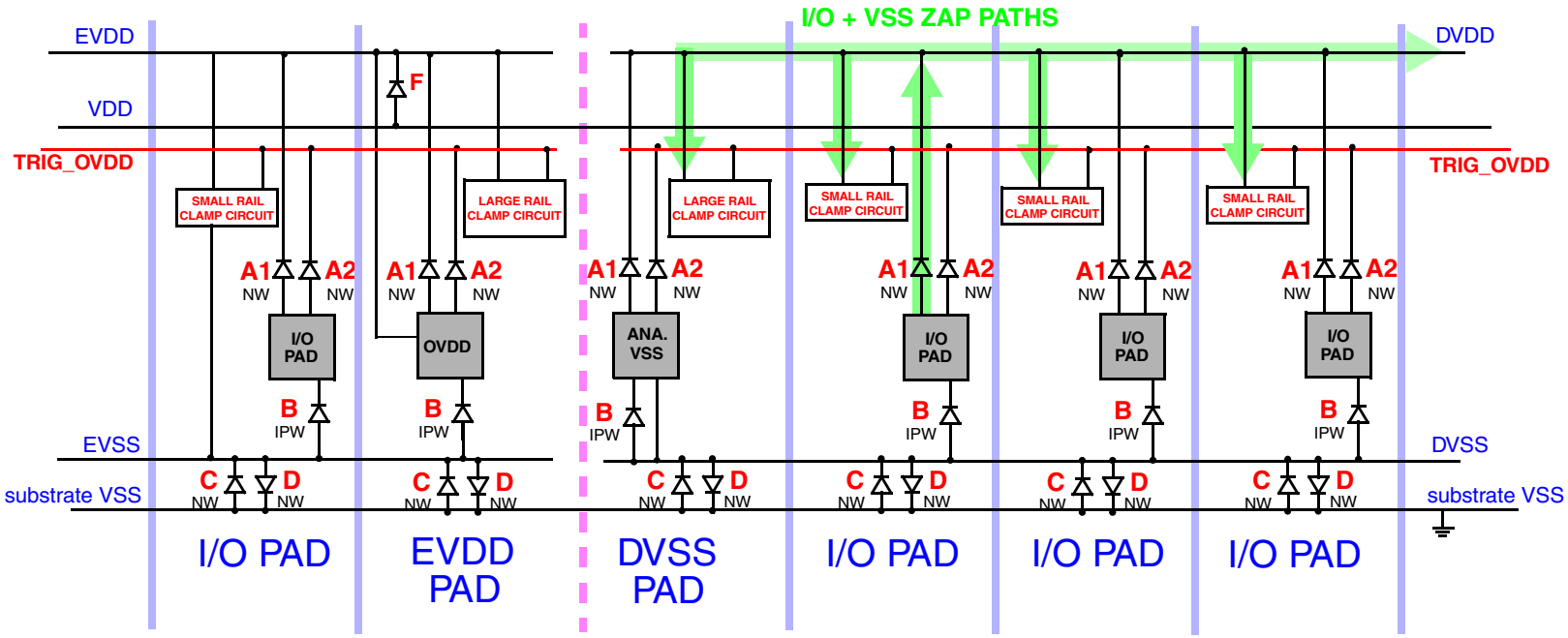


Figure 74-2. Low Noise ESD Network for HiP7 Neptune Pass 2

ESD Network Requirements:

- All rail clamp circuits are placed between the local OVDD bus segment and local OVSS.
- There are 14 independent OVDD supplies and 7 independent OVSS supplies on Neptune.
- The OVDD bus and the TRIG_OVDD bus are broken at the boundary of each segment.

ESD Network Requirements:

- Each OVDD bus segment must be terminated on both ends with a large rail clamp circuit in order to pass ESD.
- Large rail clamp circuits are placed only in power pads (OVDD, OVSS, VDD,VSS, etc.).
- Therefore, power pads must be placed on both ends of each OVDD bus segment.

HiP7 Neptune Pass 2 ESD Network for Analog Segments

Analog power and ground are simply treated as isolated OVDD and OVSS busses.

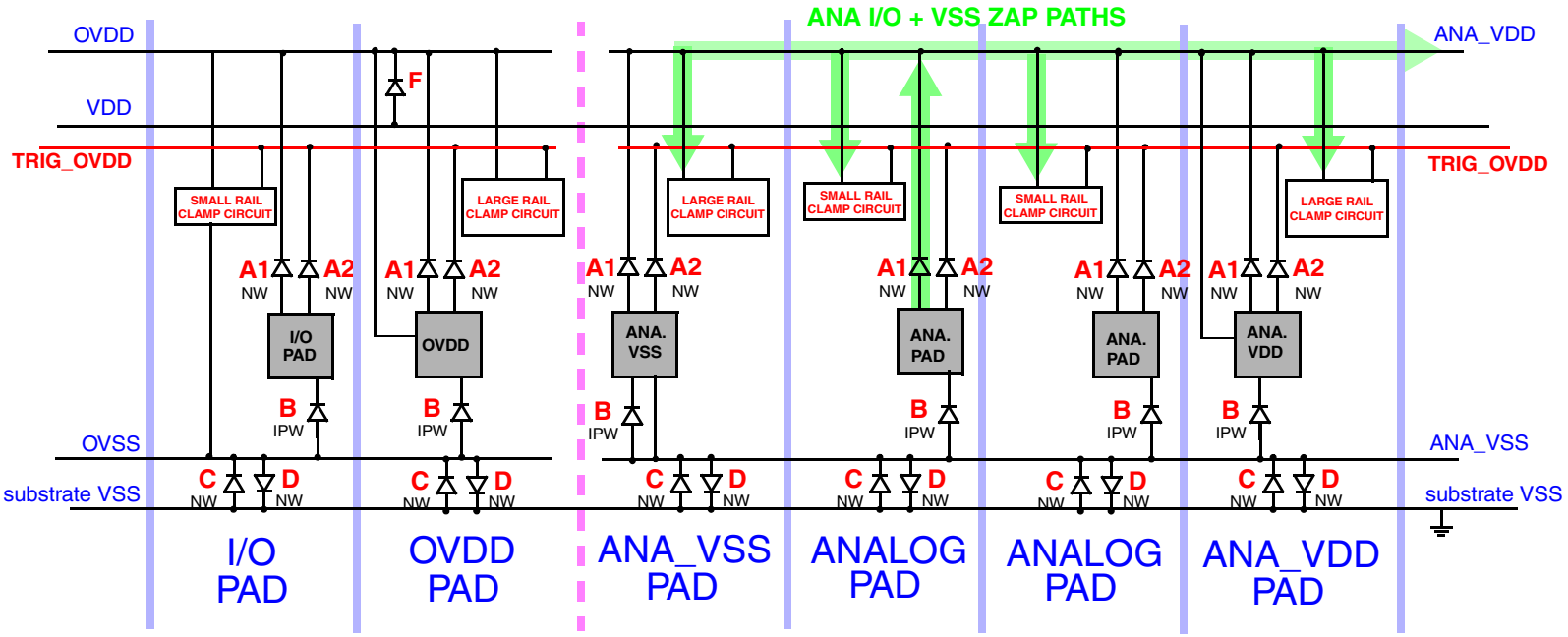


Figure 74-3. HiP7 Neptune Pass 2 ESD Network for Analog Segments

Advantages of this approach:

- Maximizes reuse of one optimized pad bus floorplan.
- May reuse layout of OVDD & OVSS pads for AVDD & AVSS.
- Minimizes the number of rails routed through the pad ring.
- Low resistance analog power and ground rails.
- Analog VDD rail is isolated from Noisy OVDD rail.

Analog/RF Application:

- Diodes A1 and B may be downsized if needed for high freq. operation -- with an ESD performance penalty.
- This was done on the GHz synthesizer input pins VCO_1 and VCO_2.

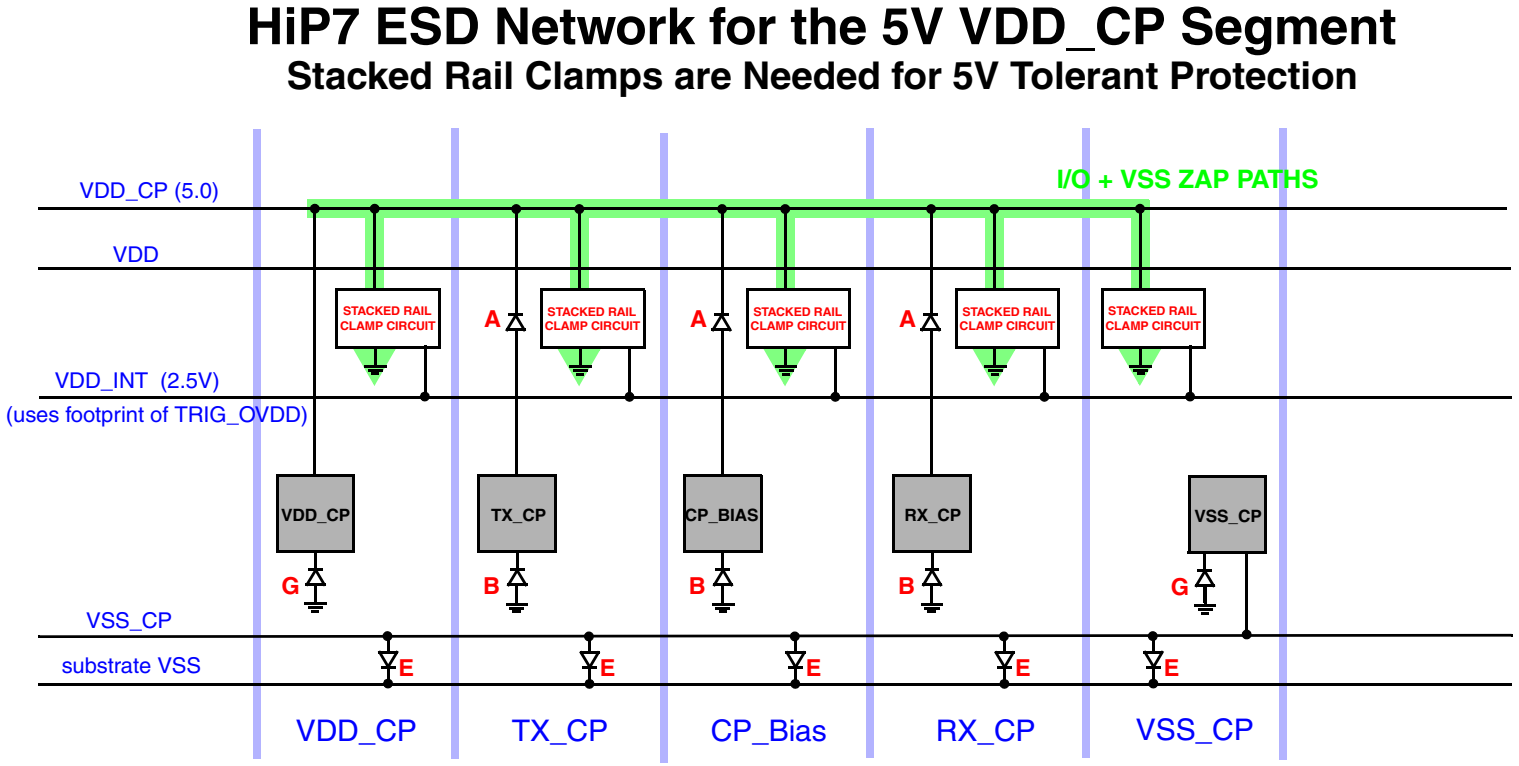


Figure 74-4. HiP7 ESD Network for the 5V VDD_CP Segment

Advantages of this approach:

- Maximizes reuse of one optimized pad bus floorplan.
- Very similar layout for all five 5V pads.
- Low resistance VDD_CP and VSS_CP rails.

5V Charge Pump Application:

- VDD_INT must be available to protect DGO MOSFET gates. A bias circuit is used on Neptune.

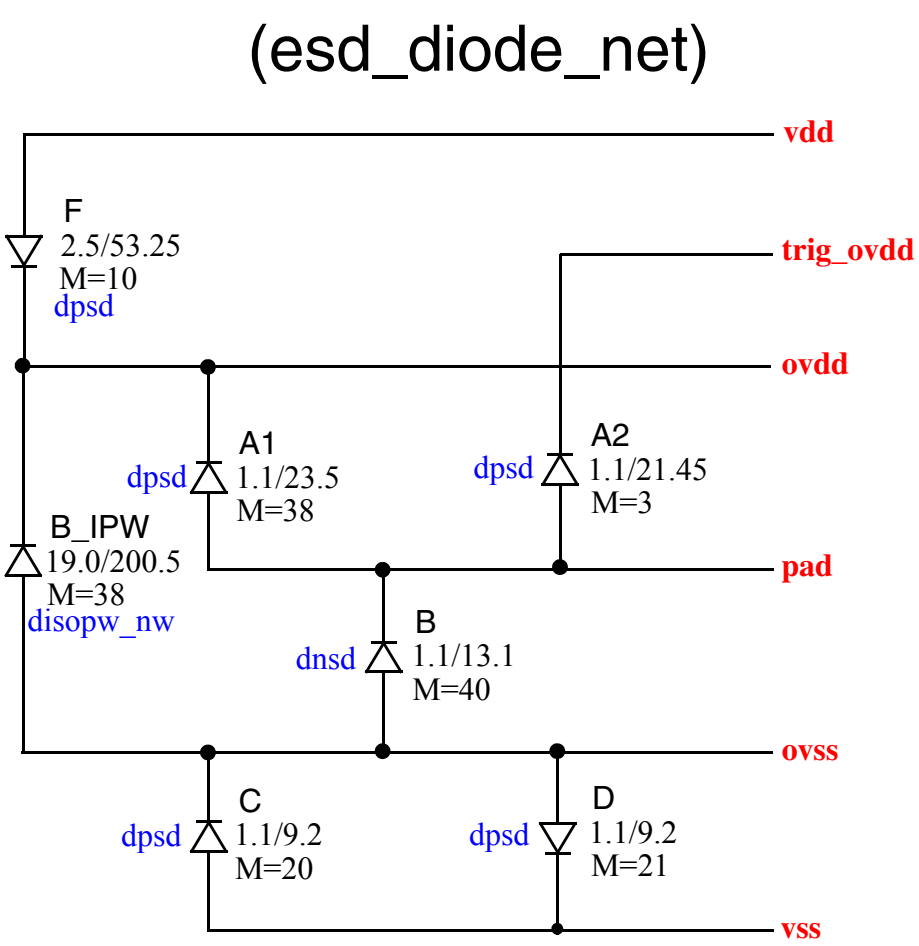


Table 74-1. Sizes for Neptune 2 ESD Network Components

Network Element	Critical Sizing Parameter	Design/ Final
Diode A1	NWELL Diode P+ Active Per.	1869.6/560.9
Diode A2	NWELL Diode P+ Active Per.	135.3/40.6
Diode B	IPW Diode N+ Active Per.	1136/340.8
Diode C	NWELL Diode P+ Acive Per.	412/123.6
Diode D	NWELL Diode P+ Acive Per.	432.6/129.8
Diode F	NWELL Diode P+ Acive Per.	1115/334.5
Large Rail Clamp	Primary DGO NMOS W	6128/1838.4
Small Rail Clamp	Primary DGO NMOS W	1100/330
5V Stacked Rail Clamp	Primary DGO NMOS W (Upper and Lower NMOS)	4830/1449

Figure 74-5. Large Rail Clamp Circuit

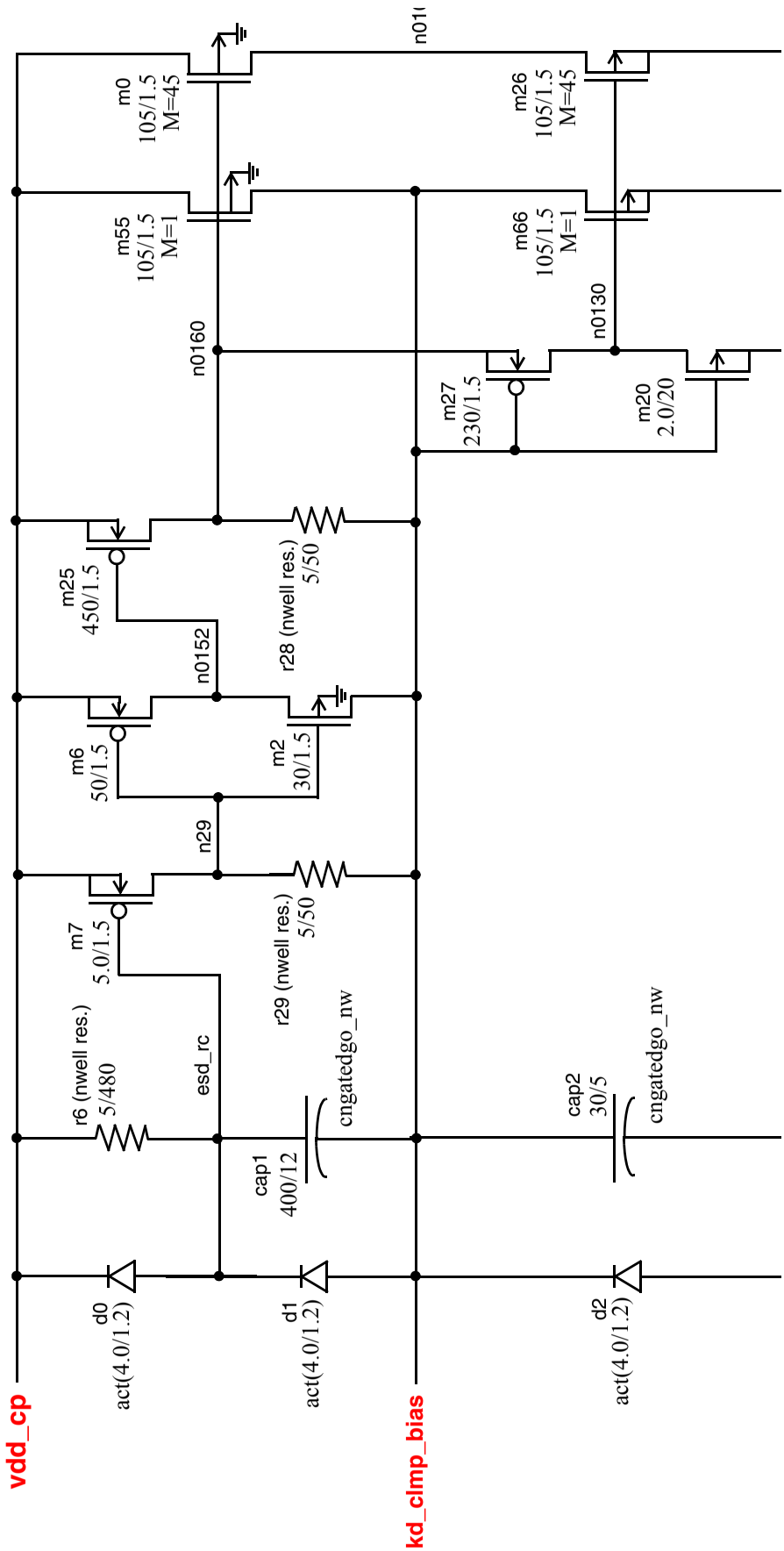


Figure 74-6. HiP7 Stacked DGO 5V Rail Clamp Circuit

SMALL RAIL CLAMP CIRCUIT

(esd_sml_clmp)

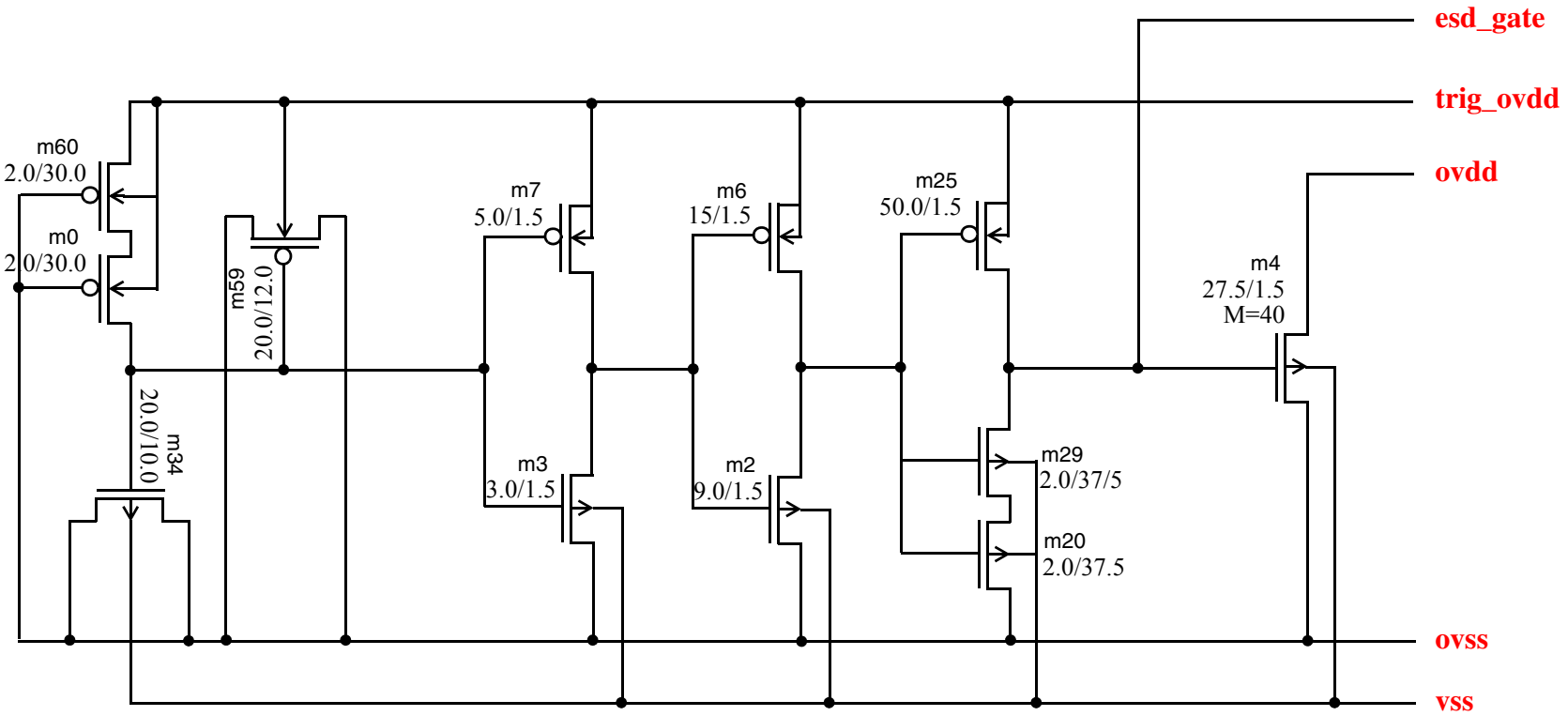


Figure 74-7. Small Rail Clamp Circuit

LARGE RAIL CLAMP CIRCUIT

(esd_lrg_clmp)

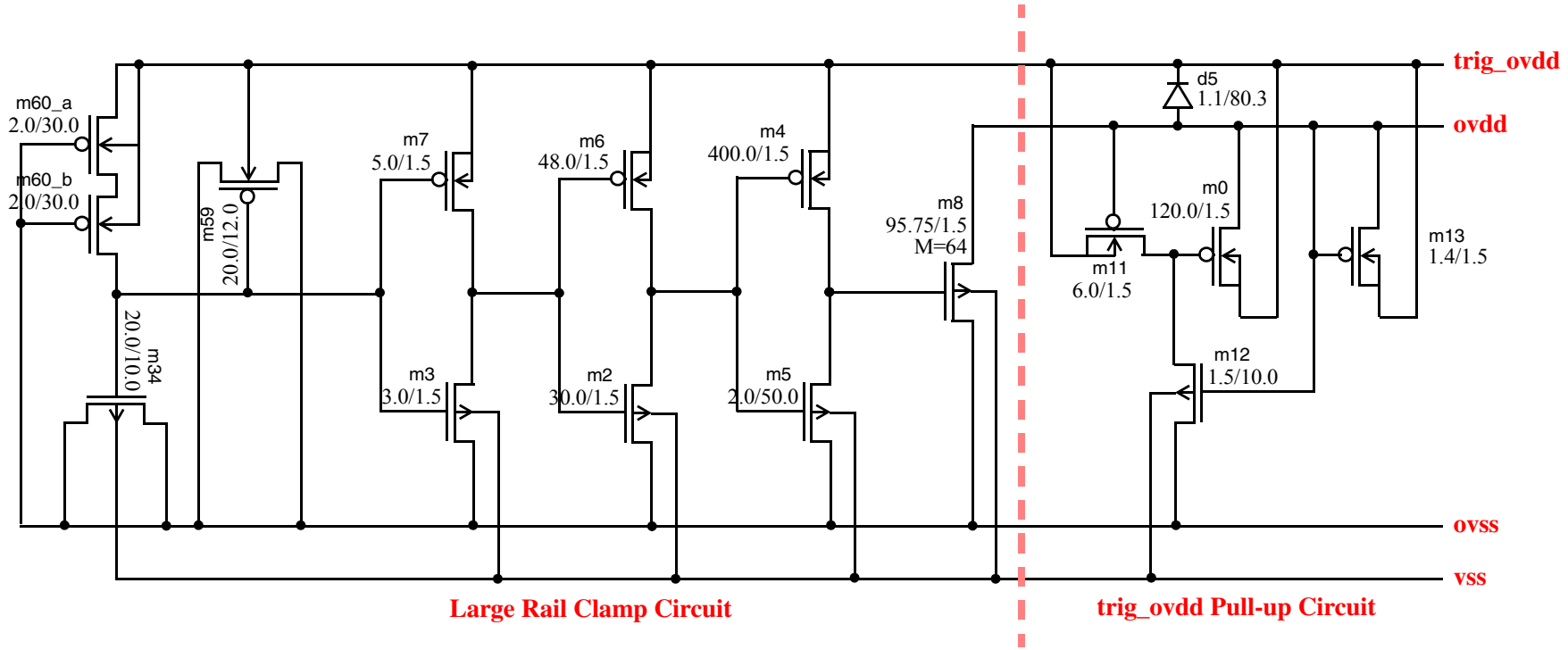


Figure 74-8. Large Rail Clamp Circuit

DIODE NETWORKS
(esd_diode_net)

For analog and power-down pad groups, the F diode is omitted.

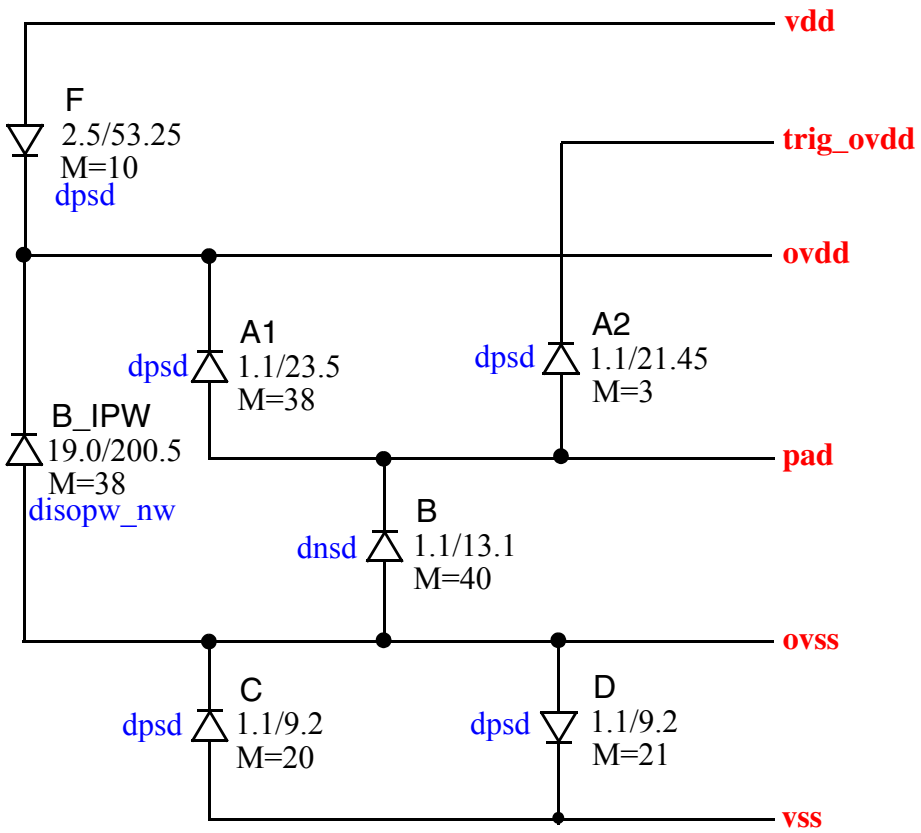


Figure 74-9. Diode Networks

74.2 Pad Drive Strength

74.2.1 User Drive Strength Control

The application software has control over the drive strength of digital pads listed in Table 6-15 on page 6-72. The drive strength is programmable in groups of pads. These groups are controlled by the listed bits in the ARM External Interface Module (AEIM) on page 48-1.

These pads may have two or three possible drive strengths that are available for selection. Write to the control bits listed in Table 6-15 on page 6-72 to select the desired drive strength.

This functionality is accomplished by connecting the listed control bits to the NOM_STRENGTH and MAXDRV ports of each I/O pad.

74.2.2 Test Drive Strength Control

Pads which are not listed in Table 6-15 on page 6-72 are not connected to bits in the ARM External Interface Module (AEIM) on page 48-1. These pads have NOM_STRENGTH and MAXDRV ports connected to the boost_test[1:0] signals of Miscellaneous Test Control Register on page 60-26.

Pads which are listed in Table 6-15 on page 6-72 but do not have an entry in the “16mA drive group” column have only the NOM_STRENGTH port connected to bits in the ARM External Interface Module (AEIM) on page 48-1. These pads have the MAXDRV port connected to the boost_test[1] signals of TCM Miscellaneous Test Control Register on page 60-26.

Therefore to enable drive strengths other than the minimum in test mode, both the AEIM Configuration Register and the TCM Miscellaneous Test Control Register may need to be written.

74.3 Pad Cells

Table 74-2. Pad Cells

Neptune Pad Type	Description
std_io_lca	Standard digital I/O pad cell described in padio chapter.
sby_io_lca	Custom digital I/O pad cell for RTC pads. From std_io_lca with all circuitry powered by OVDD. No connections to VDD. VDD buss passes through with no connection.
std_ana_lca	Standard analog pad cell with ESD protection.
rf_ana_lca	Custom analog pad cell for high frequency pins. From std_ana_lca with reduced ESD protection diode size.
hv_ana_lca	Custom analog pad cell for high voltage (~5V) outputs.
vd_io_lca	Supply for local high voltage (~3V) I/O. Pad Connected to OVDD
vs_io_lca	Pad Connected to OVSS
vd_ana_lca	From vd_io_lca with f diode removed
vs_ana_lca	From vs_io_lca with f diode removed

Table 74-2. Pad Cells

Neptune Pad Type	Description
vd_core_lca	Pad Connected to VDD
vs_core_lca	Pad Connected to VSS
vd_reg_lca	
vd_hv_lca	5V capable supply for VCO control outputs
vs_hv_lca	5V capable ground for VCO control outputs

74.4 Power Supplies

74.4.1 Power Down Modes

Table 74-3. Power Down Modes

Neptune Pin	Description	Neptune Pad Type	Source	Mode 1	Mode 2	Mode 3	Mode 3	Mode 4	Mode 5
				Every-thing On	SIM Off	Nep-tune Analog Off	Sea-weed Analog Off	Analog & SIM Off	Every-thing Off
QVDD	Core Regulator Input	vd_io_lca	Seaweed: DIG_REG_OUT	On	On	On	On	On	Off
REG_BYP_CORE	Core Regulator Output	vd_core_lca	Neptune:CORE_REG	On	On	On	On	On	Off
VDD	Core Regulator Output	vd_core_lca	Neptune:CORE_REG	On	On	On	On	On	Off
AVDD	EIM Pins	vd_io_lca	Seaweed: DIG_REG_OUT	On	On	On	On	On	Off
BVDD	Seaweed Data Pins	vd_io_lca	Seaweed: IO_REG_OUT	On	On	On	On	On	Off
CVDD	LCD/Test Pins	vd_io_lca	Seaweed: IO_REG_OUT	On	On	On	On	On	Off
DVDD	Seaweed Control Pins	vd_io_lca	Seaweed: IO_REG_OUT	On	On	On	On	On	Off
EVDD	Algae Control & Data Pins	vd_io_lca	Seaweed: IO_REG_OUT	On	On	On	On	On	Off
JVDD	Keypad Pins - also GPADC analog circuit & Master Bias Genera-tor	vd_io_lca	Seaweed: IO_REG_OUT	On	On	On	On	On	Off
FVDD	SIM Pins	vd_io_lca	Seaweed: SIM_REG_OUT	On	Off	On	On	Off	Off

Table 74-3. Power Down Modes

Neptune Pin	Description	Neptune Pad Type	Source	Mode 1	Mode 2	Mode 3	Mode 3	Mode 4	Mode 5
LVDD	RTC Circuit & Pins & analog reference	vd_io_lca	Seaweed: REF_REG	On	On	On	On	On	On
VDDA	Crystal Oscillator Circuit & Pins	vd_ana_lca	Seaweed: IO_REG_OUT	On	On	On	On	On	Off
VDD_CODEC	Codec Regulator Input	vd_ana_lca	Seaweed: AUD_REG_OUT	On	On	On	Off	Off	Off
REG_BYP_CODEC	Codec Regulator Output	vd_reg_lca	Neptune: CODEC_REG	On	On	Off	Off	Off	Off
VDD_CP	Charge Pump Supply	vd_hv_lca	Seaweed: VM_REG_OUT						
VDD_TXRX	TXRX Regulator Input	vd_ana_lca	Seaweed: RF_REG_OUT	On	On	On	Off	Off	Off
REG_BYP_TXRX	TXRX Regulator Output	vd_reg_lca	Neptune: TXRX_REG	On	On	Off	Off	Off	Off
REG_BYP_PRE	Prescaler Regulator Output	vd_ana_lca	Neptune: PRE_REG	On	On	Off	Off	Off	Off

Chapter 75
Neptune LTE Electrical Requirements (ELEC)

Revision History

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	7/19/02	Mark Babcock	Update Power Consumption numbers. Add max Tj. Update FVDD per DSPhl13819. Update VDD_CP per DSPhl13821.
0.2	07/23/02	Mark Babcock	Updated Table 75-1 supply voltage.
0.3	08/26/02	Mark Babcock, Anand Gupta, Kamlesh Kumar	Updated per DDTS# DSPhl15148, DSPhl14662, and DSPhl15246. Added DMAC reference, and updated Table 75-1 JVDD spec.
0.4	12/19/02	Mark Babcock, Shannon Osgood	Updated per DDTS# DSPhl15183. Updated BBP and SAP timing tables and diagrams.
0.5	04/29/03	Mark Babcock, Shannon Osgood	Updated per DDTS# DSPhl15915. Updated minimum core VDD spec and minimum on QVDD.
0.6	05/07/03		Updated for LTE specification release.

This section defines the electrical characteristics for the Neptune LTE Program.

75.1 DC Operating Conditions

Table 75-1. DC Recommended Operating Conditions

Characteristics	Symbol	Min	Typ	Max	Units
Supply Voltage (Output of Core Regulator - internal node only)	V_{DD}	1.496	1.575 (±5%)	1.654	V
Supply Voltage (Input of Core Regulator)	QVDD	1.800	1.875 (±4%)	1.950	V
Supply Voltage (Level Shift I/O including special levels for SIM)	AVDD, BVDD, CVDD, DVDD, EVDD,	1.8		3.0	V
Supply for Keypad and Low Frequency Control Pins. Also powers analog portion of General Purpose ADC, ANATEST, and parts of TXRX Regulator and Core Regulator.	JVDD	2.68	2.775	2.875	V
Supply Voltage (Level Shift I/O including special levels for SIM)	FVDD	1.8		3.1	V
Supply Voltage (RTC logic and I/O)	LVDD	1.543	1.575 (±2%)	1.607	V
Supply Voltage (Clock Amplifier, Clock Monitor)	VDDA	2.68	2.775 (±3.6%)	2.875	V
Supply Voltage (Input of Regulators for Tx/Rx/CODEC)	VDD_TXRX, VDD_CODEC	2.68	2.775 (±3.6%)	2.875	V
Supply Voltage (Charge Pump)	VDD_CP	4.50	4.70	5.15	V
High-Level DC Input Voltage	V_{IH}	0.7* VDD		PVDD	V
Low-Level DC Input Voltage	V_{IL}	0.0		0.3* VDD	V
Input Transition Time (T_R/T_F)	T_T	0		25	ns
Operating Ambient Temperature	T_A	-20		70	^oC
Junction Temperature	T_J	-20		85	^oC

Table 75-2. DC Absolute Maximum Operating Conditions

Characteristics	Symbol	Min	Typ	Max	Units
Supply Voltage (Output of Core Regulator - internal node only)	V_{DD}	-0.5		2.0	V
Supply Voltage (Input of Core Regulator)	QVDD	-0.5		3.6	V

Table 75-2. DC Absolute Maximum Operating Conditions (Continued)

Characteristics	Symbol	Min	Typ	Max	Units
Supply Voltage (Level Shift I/O)	AVDD, BVDD, CVDD, DVDD, EVDD, FVDD, JVDD	-0.5		3.6	V
Supply Voltage (RTC logic and I/O)	LVDD	-0.5		2.0	V
Supply Voltage (Clock Amplifier, Clock Monitor)	VDDA	-0.5		3.6	V
Supply Voltage (Input of Regulators for Tx/Rx/CODEC)	VDD_TXRX, VDD_CODEC	-0.5		3.6	V
Supply Voltage (Charge Pump)	VDD_CP	-0.5		6.0	V
Input Voltage Range	V_I	-0.5		PVDD+0.3	V
Input Clamp Current (V_1<0 or V_1>QVDDH)	I_I		+/-20		mA
Output Clamp Current (V_1<0 or V_1>QVDDH)	I_O		+/-20		mA
Storage Temperature Range		-65		150	^oC

NOTE:

PVDD is intended to mean the supply of the level-shift I/O port under test.

NOTE:

Stresses beyond those listed under “absolute maximum ratings” may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under “recommended operating conditions” is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

Table 75-3. DC Electrical Characteristics

Characteristics	Symbol	Test Condition	Min	Max	Units
High Level Output	V_{OH}	I_{OH} = Max Rated	PVDD-0.5		V
High Level Output	V_{OH}	I_{OH} = 100uA	PVDD-0.2		V
Low Level Output	V_{OL}	I_{OL} = Max Rated		0.5	V
Low Level Output	V_{OL}	I_{OL} = 100uA		0.2	V
High Level Input Current	I_{IH}	V_I = PVSS		+/-1	uA

Table 75-3. DC Electrical Characteristics (Continued)

Characteristics	Symbol	Test Condition	Min	Max	Units
Low Level Input Current	I_{IL}	V_I = PVDD		+/-1	uA
Schmitt Trigger Vt+	V_{t+}		1.4		V
Schmitt Trigger Vt-	V_{t-}			0.8	V
High Level Output Current	I_{OH}	V_{OH}= PVDD-0.5V		-4, -8,-16	mA
Low Level Output Current	I_{OL}	V_{OL} = 0.5		4, 8,16	mA
Input Leakage - no PU/PD	I_{IN}	V_I = PVDD/V_{SS}		1	uA
Input Leakage - w/PU	I_{IN}	V_I = V_{SS}	-25	-50	uA
Input Leakage - w/PD	I_{IN}	V_I = PVDD	25	50	uA
High Level Input Current	I_{IH}	V_I = PVSS		+/-1	uA
Low Level Input Current	I_{IL}	V_I = PVDD		+/-1	uA
Tri-State Hi_Z Current	I_Z	V_I = PVDD		+/-20	uA
Tri-State Hi_Z Current	I_Z	V_I = PVSS		+/-20	uA

NOTE:

PVDD and PVSS are intended to be placeholders representing the supply of the level-shift I/O port under test.

NOTE:

Max Rated = 4mA, 8mA, or 16mA, defined by the table of I/O drive strength groups and the setting for each group.

Table 75-4. DSP Current Drain[1]

Characteristic	Symbol	Conditions	MIN	NOM	MAX[2]	Units
STOP Current	$Idd_{STOPdsp}$	VDD = 1.575V DSP in STOP XTAL = 26Mhz		2	4	mA
WAIT Current	$Idd_{WAITdsp}$	VDD = 1.575V DSP in WAIT, peripherals in default state XTAL = 26Mhz DSP CLK = 104MHz		12	18	mA
Active Current	Idd_{RUNdsp}	VDD = 1.575V DSP running tight software loop using DSP program and Data RAM		0.5	0.7	mA/Mhz

1. Numbers indicate DSP subsystem values only
2. At 70° C

Table 75-5. MCU Current Drain[1]

Characteristic	Symbol	Conditions	MIN	NOM	MAX[2]	Units
STOP Current	$Idd_{STOPmcu}$	VDD = 1.575V MCU in STOP XTAL = 26Mhz DPLLs OFF		2	4	mA
DOZE Current	$Idd_{DOZEmcu}$	VDD = 1.575V MCU in DOZE All peripherals in default state XTAL = 26Mhz MCU DPLL = 52MHz		5	7	mA
WAIT Current	$Idd_{WAITmcu}$	VDD = 1.575V MCU in WAIT All peripherals in disabled XTAL = 26Mhz MCU DPLL= 52MHz		6	9	mA
RUN Current	Idd_{RUNmcu}	VDD = 1.575V MCU running tight loop in internal RAM All peripherals disabled		0.4	0.6	mA/Mhz

1. Numbers indicate MCU subsystem values only.
2. At 70° C

Table 75-6. Deep Sleep Current

Characteristic	Symbol	Conditions[1]	MIN	NOM	MAX[2]	Units
Deep Sleep Current	Idd_{Deep}	26MHz off 32 KHz on DSM module on only DSP & MCU in STOP Only core regulator on			350	uA

1. Neptune should not be driving any external resistive nodes.
2. At 25° C

75.2 Power Up Sequence

This section is intended to define the power supply sequence for the Neptune LTE IC. Figure 75-1 illustrates the suggested power up sequence. Table 75-7 illustrates the recommended delay between the ramp of each supply group.

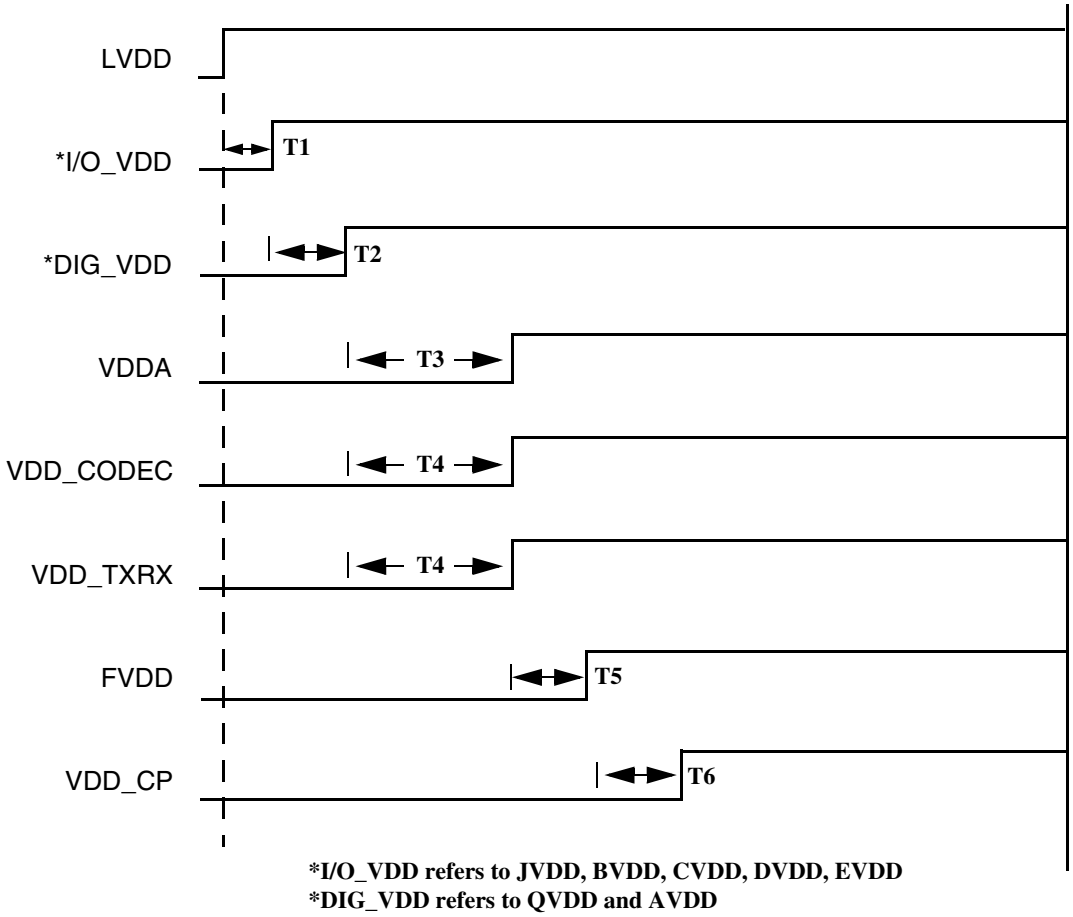


Figure 75-1. Power Up Sequence

NOTE:

The Neptune LTE IC must be held in reset prior to apply the LVDD supply. Reset must be maintained until power sequence is complete.

NOTE:

The sequence for DIG_VDD and I/O_VDD can be switched (i.e. DIG_VDD represented by T1 and I/O_VDD represented by T2). However, there are ESD diodes that could be forward biased in this case. During the time when DIG_VDD is supplied and I/O_VDD is not, these diodes could start to conduct current based on the potential difference of the two sets of rails.

Table 75-7. Power Sequence Delays

Sequence Delay	Definition
T1	3ms
T2	T1+4ms
T3	T2<T3≤T4
T4	T2+12ms
T5	T3+4ms
T6	T4+4ms

NOTE:

The suggested delay definitions are considered nominal. The sequence delay will not be characterized.

75.3 AC Characteristics

The AC Electrical Characteristics are given for VDD=1.575V and AVDD=1.8V at a capacitive load of 50pF.

75.3.1 Crystal Oscillator Characteristics

Please see Section 30.7.6, “Specifications,” on page 30-37 for Crystal Oscillator electrical characteristics.

75.3.2 Reference Digital Phase Locked Loop Characteristics

Table 75-8 and Table 75-8 list the electrical restrictions and valid REFDPLL settings.

Table 75-8. Electrical Restrictions

Signal	Clock Speed
2XPATREF	26MHz
REF_DPLL_OUT	260MHz or less
DSP_CLK	130MHz or less
MCU_CLK	52MHz or less

Table 75-9. Valid REFDPLL Settings & 2XPATREF, MCU, & DSP Divider

PDF	MFI	RDD	REF_DPLL_OUT (MHz)	MCU or DSP OutDiv	DSP Fop (MHz)	MCU Fop (MHz)
2	10	10	260	1	130	
2	10	10	260	2	65	
2	10	10	260	4	32.5	
2	10	10	260	8	16.25	
2	10	10	260	16	8.125	
2	10	10	260	32	4.0625	
2	10	10	260	64	2.0312	
2	10	10	260	128	1.0156	
2	10	10	260	5		52
2	10	10	260	6		43.3333
2	10	10	260	8		32.5
2	10	10	260	10		26
2	6	6	156	1	78	

Table 75-9. Valid REFDPLL Settings & 2XPATREF, MCU, & DSP Divider (Continued)

PDF	MFI	RDD	REF_DPLL_OUT (MHz)	MCU or DSP OutDiv	DSP Fop (MHz)	MCU Fop (MHz)
2	6	6	156	2	39	
2	6	6	156	4	19.5	
2	6	6	156	8	9.75	
2	6	6	156	16	4.875	
2	6	6	156	32	2.4375	
2	6	6	156	64	1.21875	
2	6	6	156	128	0.609375	
2	6	6	156	4		39
2	6	6	156	6		26
2	6	6	156	12		13
2	5	5	130	1	65	
2	5	5	130	2	32.5	
2	5	5	130	4	16.25	
2	5	5	130	8	8.125	
2	5	5	130	16	4.0625	
2	5	5	130	32	2.03125	
2	5	5	130	64	1.01563	
2	5	5	130	128	0.507813	
2	5	5	130	4		32.5
2	5	5	130	10		13
4	12	6	156	1	78	
4	12	6	156	2	39	
4	12	6	156	4	19.5	
4	12	6	156	8	9.75	
4	12	6	156	16	4.875	
4	12	6	156	32	2.4375	
4	12	6	156	64	1.21875	
4	12	6	156	128	0.609375	

Table 75-9. Valid REFDPLL Settings & 2XPATREF, MCU, & DSP Divider (Continued)

PDF	MFI	RDD	REF_DPLL_OUT (MHz)	MCU or DSP OutDiv	DSP Fop (MHz)	MCU Fop (MHz)
4	12	6	156	4		39
4	12	6	156	6		26
4	12	6	156	12		13
4	10	5	130	1	65	
4	10	5	130	2	32.5	
4	10	5	130	4	16.25	
4	10	5	130	8	8.125	
4	10	5	130	16	4.0625	
4	10	5	130	32	2.03125	
4	10	5	130	64	1.01563	
4	10	5	130	128	0.507813	
4	10	5	130	4		32.5
4	10	5	130	10		13
4	8	4	104	1	52	
4	8	4	104	2	26	
4	8	4	104	4	13	
4	8	4	104	8	6.5	
4	8	4	104	16	3.25	
4	8	4	104	32	1.625	
4	8	4	104	64	0.8125	
4	8	4	104	128	0.40625	
4	8	4	104	2		52
4	8	4	104	4		26
4	8	4	104	8		13
4	6	3	78	1	39	
4	6	3	78	2	19.5	
4	6	3	78	4	9.75	
4	6	3	78	8	4.875	

Table 75-9. Valid REFDPLL Settings & 2XPATREF, MCU, & DSP Divider (Continued)

PDF	MFI	RDD	REF_DPLL_OUT (MHz)	MCU or DSP OutDiv	DSP Fop (MHz)	MCU Fop (MHz)
4	6	3	78	16	2.4375	
4	6	3	78	32	1.21875	
4	6	3	78	64	0.609375	
4	6	3	78	128	0.304688	
4	6	3	78	2		39
4	6	3	78	6		13
8	12	3	78	1	39	
8	12	3	78	2	19.5	
8	12	3	78	4	9.75	
8	12	3	78	8	4.875	
8	12	3	78	16	2.4375	
8	12	3	78	32	1.21875	
8	12	3	78	64	0.609375	
8	12	3	78	128	0.304688	
8	12	3	78	2		39
8	12	3	78	6		13
8	8	2	52	1	26	
8	8	2	52	2	13	
8	8	2	52	4	6.5	
8	8	2	52	8	3.25	
8	8	2	52	16	1.625	
8	8	2	52	32	0.8125	
8	8	2	52	64	0.40625	
8	8	2	52	128	0.203125	
8	8	2	52	2		26
8	8	2	52	4		13
16	8	1	26	1	13	
16	8	1	26	2	6.5	

Table 75-9. Valid REFDPLL Settings & 2XPATREF, MCU, & DSP Divider (Continued)

PDF	MFI	RDD	REF_DPLL_OUT (MHz)	MCU or DSP OutDiv	DSP Fop (MHz)	MCU Fop (MHz)
16	8	1	26	4	3.25	
16	8	1	26	8	1.625	
16	8	1	26	16	0.8125	
16	8	1	26	32	0.40625	
16	8	1	26	64	0.203125	
16	8	1	26	128	0.101563	
16	8	1	26	2		13

Please see Section 34.2, “DPLL Specifications,” on page 34-3 for more Reference DPLL electrical characteristics.

75.3.3 DSP Internal Clocks

For each occurrence of T_{DH}, T_{DL}, T_{DC} or I_{DCYC} substitute with the numbers given in Table 75-10, (The used terms EfD and ET_{DC} are described at Table 75-11), the terms MF,DF,PDF stand for DSP PLL Multiplication Factor, Division Factor and Predivision Factor respectively and are described in Chapter 3:

Table 75-10. Internal DSP Clocks

Characteristics	Symbol	Expression
Internal DSP Operation Frequency With PLL Enabled	fD	(EfD•MF)/(PDF•DF)
Internal DSP Operation Frequency With PLL Disabled	fD	EfD/2
Internal DSPClock High Period - with PLL disabled - with PLL enabled and MF ≤ 4 - with PLL enabled and MF > 4	T_{DH}	ET_{DC} (Min) 0.49•ET_{DC}•PDF•DF/MF (Max) 0.51•ET_{DC}•PDF•DF/MF (Min) 0.47•ET_{DC}•PDF•DF/MF (Max) 0.53•ET_{DC}•PDF•DF/MF
Internal DSP Clock Low Period - with PLL disabled - with PLL enabled and MF ≤ 4 - with PLL enabled and MF > 4	T_{DL}	ET_{DC} (Min) 0.49•ET_{DC}•PDF•DF/MF (Max) 0.51•ET_{DC}•PDF•DF/MF (Min) 0.47•ET_{DC}•PDF•DF/MF (Max) 0.53•ET_{DC}•PDF•DF/MF
Internal DSP Clock Cycle Time With PLL Enabled	T_{DC}	ET_{DC}•PDF•DF/MF
Internal DSP Clock Cycle Time With PLL Disabled	T_{DC}	2 • ET_{DC}
Instruction DSP Cycle Time	I_{DCYC}	T_{DC}

Table 75-11. DSP Clocks

Characteristics	Symbol	Min	Max	Unit
Frequency Of The Internal DSP-REF Clock	EfD	0	130	MHz
Internal DSP-REF Clock Cycle Time with PLL disabled with PLL enabled	ET_{DC}	 7.7 7.7	 • 273100	ns

Table 75-12. MCU Clocks

Characteristics	Symbol	Min	Max	Unit
Frequency Of The Internal MCU-CLK Clock	fM	0	52	MHz
Internal MCU-CLK Clock Cycle Time	T_{MC}	19.8	•	ns

75.3.4 Clock Monitor Module Electrical Characteristics

Please see Table 24-1 on page 24-2 for Clock Monitor electrical characteristics.

75.3.5 Reset, Mode Select, and Interrupt Timing

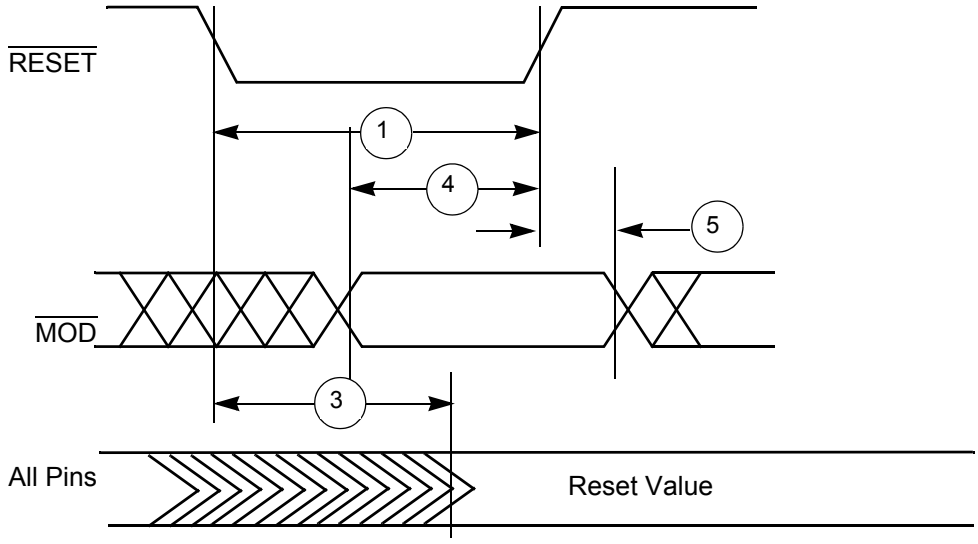


Figure 75-2. Reset Timing

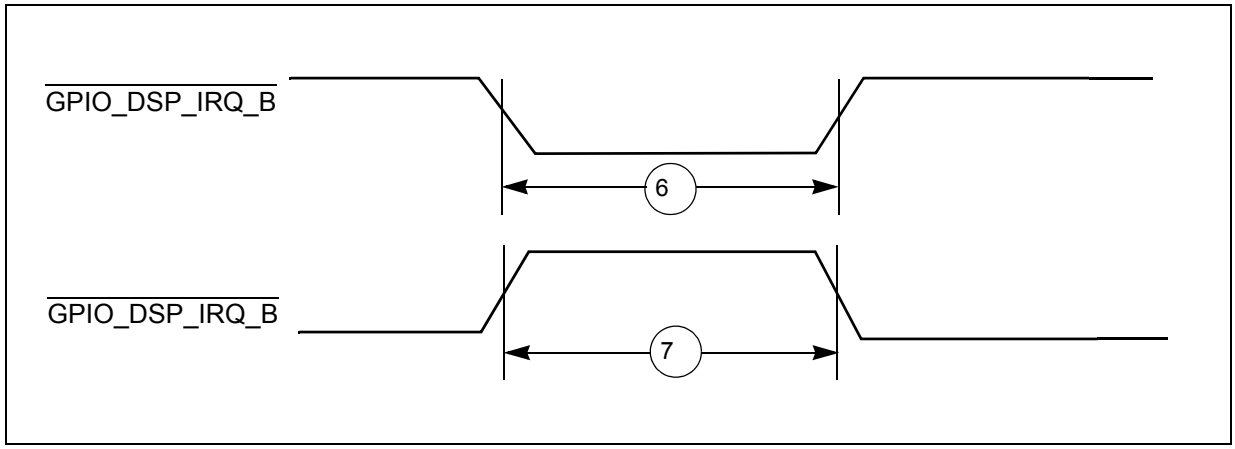


Figure 75-3. DSP External Interrupt Timing (Negative Edge-Triggered)

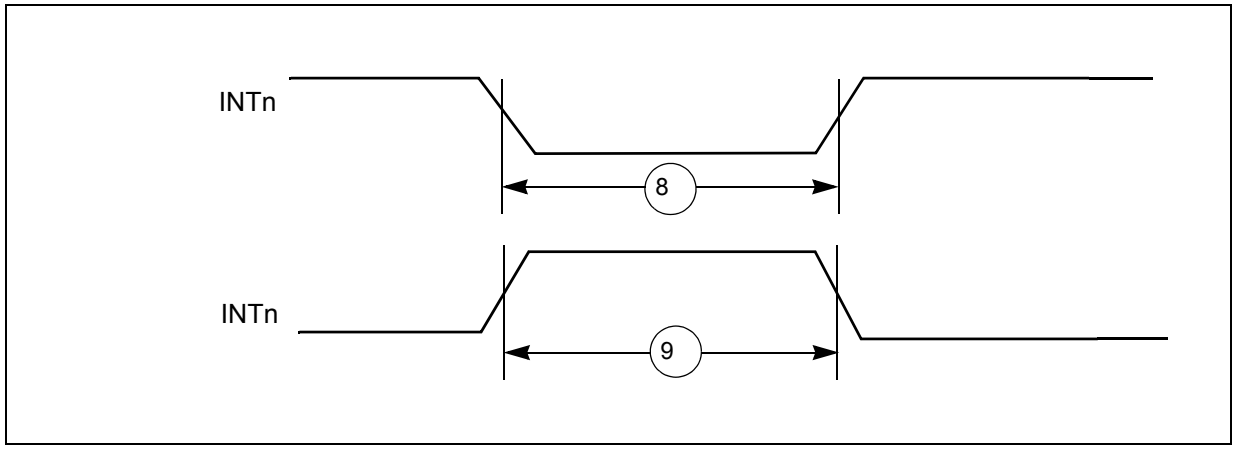


Figure 75-4. NT0-INT7 External Interrupt Timing

Table 75-13. Mode Select and Interrupt Timing Specifications

Num	Characteristics	Expression	@26MHz/32768Hz MCU 130MHz DSP		Unit
			Min	Max	
1	$\overline{\text{RESET\_IN}}$ Duration to be qualified as valid	$4*T_{CKIL}+0.05$	122.05	-	µs
2	$\overline{\text{RESET\_IN}}$ Duration to be qualified as not-valid	$2*T_{CKIL}-0.05$	-	60.98	µs
3	Delay from $\overline{\text{RESET\_IN}}$ Assertion to All Pins at Reset Value (periodically sampled and not 100% tested).	min: $4.5*T_{CKIL}$ max: $5.5*T_{CKIL}$	137.33	167.85	µs
4	MOD Select Setup Time		0	-	µs
5	MOD Select Hold Time	$4*T_{CKIL}+0.05$	122.05	-	µs
6	Minimum Edge-Triggered $\overline{\text{GPIO\_DSP\_IRQ\_B}}$ Assertion Width		T_{DC}		ns
7	Minimum Edge-Triggered $\overline{\text{GPIO\_DSP\_IRQ\_B}}$ Negation Width		T_{DC}		ns
8	Minimum Edge-Triggered INTn Width High		T_{MC}		ns
9	Minimum Edge-Triggered INTn Width Low		T_{MC}		ns

75.3.6 AEIM and Emulation Port Timing

Table 75-14. AEIM External Bus AC Timing Specifications Relative to M_CLK

Num	Characteristics	Expression	@26 MHz		Unit
			Min	Max	
11	M_CLK rise to aeim_addr_out Valid		--	5.5	ns
12	M_CLK rise to aeim_addr_out Invalid (hold)		1.5	--	ns
13	M_CLK rise to aeim_cs_b, aeim_cs5 Assertion		--	6	ns
14	M_CLK rise to aeim_cs_b, aeim_cs5 Negation (hold)		1.5	--	ns
15	M_CLK rise to aeim_eb_b Assertion (Write, EBASS=even)		1.5	6	ns
16	M_CLK fall to aeim_eb_b Assertion (Write, EBASS=odd)		1.5	6	ns
17	M_CLK rise to aeim_eb_b Negation (Write, EBNEG=even)		1.5	6	ns
18	M_CLK fall to aeim_eb_b Negation (Write, EBNEG=odd)		1.5	6	ns
19	M_CLK rise to aeim_eb_b Assertion (Read, EBM=0)		1.5	6	ns
20	M_CLK rise to aeim_eb_b Negation (Read, EBM=0)		1.5	6	ns
21	M_CLK rise to aeim_rw_b Assertion (Write, EBASS=even)		1.5	6	ns
22	M_CLK fall to aeim_rw_b Assertion (Write, EBASS=odd)		1.5	6	ns
23	M_CLK rise to aeim_rw_b Negation (Write, EBNEG=even)		1.5	6	ns
24	M_CLK fall to aeim_rw_b Negation (Write, EBNEG=odd)		1.5	6	ns
25	M_CLK rise to aeim_rw_b Assertion (Read)		1.5	6	ns
26	M_CLK rise to aeim_rw_b Negation (Read)		1.5	6	ns
27	M_CLK rise to aeim_oe_b Assertion (Read, OEASS=even)		1.5	6	ns
28	M_CLK fall to aeim_oe_b Assertion (Read, OEASS=odd)		1.5	6	ns
29	M_CLK rise to aeim_oe_b Negation (Read)		1.5	6	ns
30	M_CLK rise to aeim_lba_b Assertion ((BEN=0 & LBAEN=1) or BEN=1)		1.5	6	ns
31	M_CLK rise to aeim_lba_b Negation ((BEN=0 & LBAEN=1) or (Write & BEN=1))		1.5	6	ns
32	M_CLK rise to aeim_lba_b Negation (Read, BEN=1, IBDLY=odd)		1.5	6	ns
33	M_CLK fall to aeim_lba_b Negation (Read, BEN=1, IBDLY=even)		1.5	6	ns
34	M_CLK rise to aeim_bclk_out rise (Read, BEN=1, IBDLY=even)		1.5	6	ns

Table 75-14. AEIM External Bus AC Timing Specifications Relative to M_CLK (Continued)

Num	Characteristics	Expression	@26 MHz		Unit
			Min	Max	
35	M_CLK fall to aeim_bclk_out rise (Read, BEN=1, IBDLY=odd)		1.5	6	ns
36	M_CLK rise to aeim_bclk_out fall (Read, BEN=1, IBDLY=odd)		1.5	6	ns
37	M_CLK fall to aeim_bclk_out fall (Read, BEN=1, IBDLY=even)		1.5	6	ns
38	M_CLK rise to aeim_data_out		10.5	17.5	ns
39	aeim_data_in Valid to M_CLK rise (setup)		7.5	--	ns
40	M_CLK rise to aeim_data_in Invalid (hold)		1.5	--	ns
41	aeim_ecb_b Valid to M_CLK rise (setup)		16.6	--	ns
42	M_CLK rise to aeim_ecb_b Invalid (hold)		1.5	--	ns
43	M_CLK fall to aeim_wr_oe Assertion		1.5	6	ns
44	M_CLK rise to aeim_wr_oe Negation		1.5	6	ns
45	M_CLK rise to aeim_strobe Assertion		1.5	5.4	ns
46	M_CLK fall to aeim_strobe Negation		1.5	5.4	ns
47	M_CLK rise to aeim_show_rw_b, aeim_siz Valid		--	5.5	ns
48	M_CLK rise to aeim_show_rw_b, aeim_siz Invalid (hold)		1.5	--	ns
49	M_CLK rise to aeim_qstat Valid		--	15.5	ns
50	M_CLK rise to aeim_qstat Invalid (hold)		1.5	--	ns

Notes:

1. All timings are measured at the ARM7 Platform boundary with respect to the MCU Clock input to the ARM7 Platform, NOT at the chip boundary.
2. Min numbers are BCS, -40C, 1.65V with a loading of 2.5pF.
3. Max numbers are WCS, 105C, 1.35V with a loading of 2.5pF.
4. aeim_eb_b outputs are asserted for Reads if the EBM bit in the corresponding CS control register is clear.
5. aeim_rw_b output follows aeim_eb_b outputs timing for Writes.
6. aeim_wr_oe outputs are used to control Data pads output enable, and hence, are internal to Neptune chip.
7. aeim_strobe, aeim_show_rw_b, aeim_siz and aeim_qstat outputs are only used for emulation and real-time trace support.

Figure 75-5. AEIM External Bus Timing relative to M_CLK

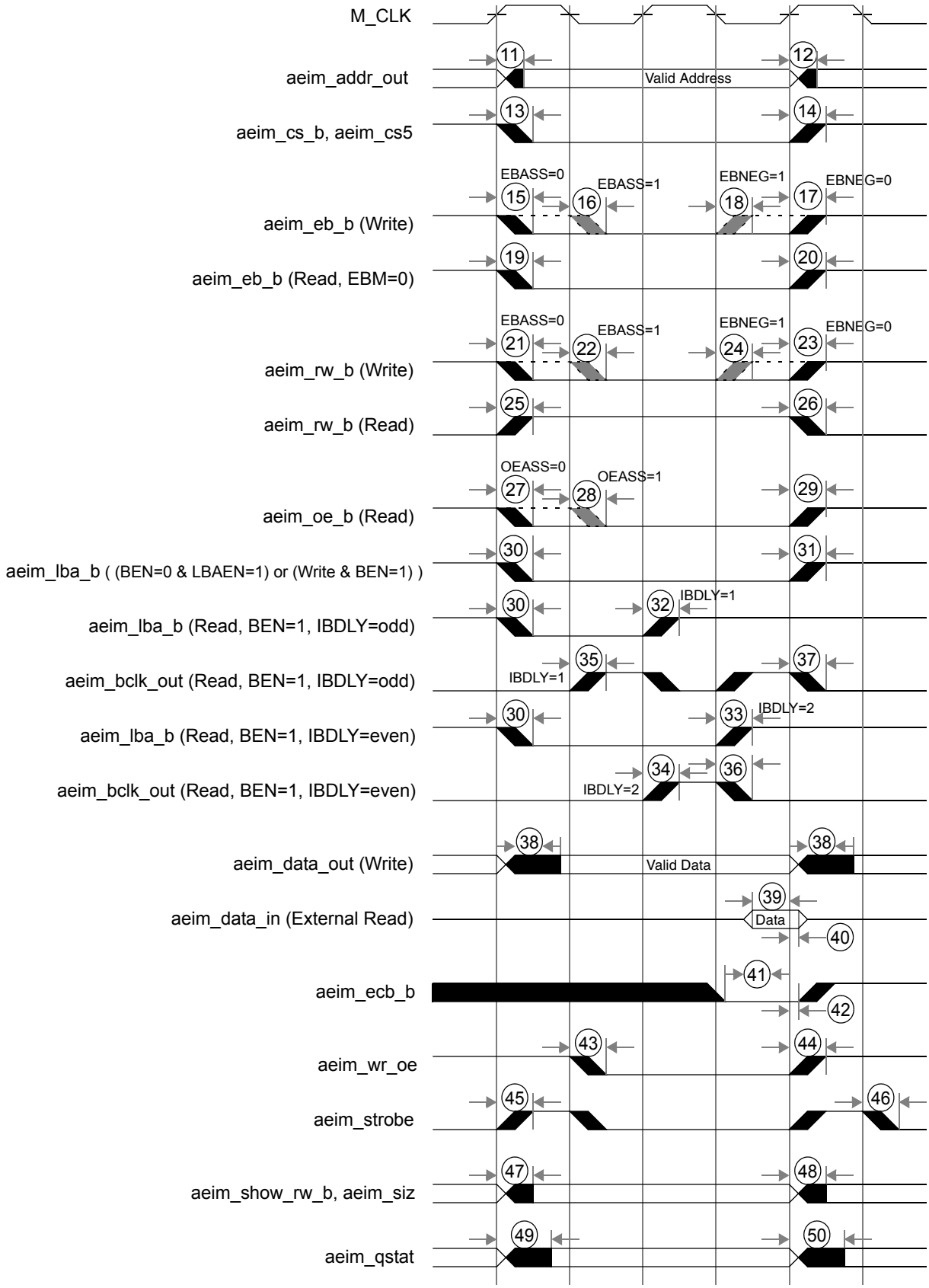

M_CLK
aeim_addr_out
Valid Address
aeim_cs_b, aeim_cs5
EBASS=0
EBASS=1
EBNEG=1
EBNEG=0
aeim_eb_b (Write)
aeim_eb_b (Read, EBM=0)
aeim_rw_b (Write)
aeim_rw_b (Read)
OEASS=0
OEASS=1
aeim_oe_b (Read)
aeim_lba_b ((BEN=0 & LBAEN=1) or (Write & BEN=1))
IBDLY=1
aeim_lba_b (Read, BEN=1, IBDLY=odd)
aeim_bclk_out (Read, BEN=1, IBDLY=odd)
IBDLY=2
aeim_lba_b (Read, BEN=1, IBDLY=even)
aeim_bclk_out (Read, BEN=1, IBDLY=even)
aeim_data_out (Write)
Valid Data
aeim_data_in (External Read)
Data
aeim_ecb_b
aeim_wr_oe
aeim_strobe
aeim_show_rw_b, aeim_siz
aeim_qstat

Table 75-15. AEIM External Bus AC Timing Specifications Relative to CKO

Num	Characteristics	Expression	@26 MHz		Unit
			Min	Max	
51	CKO rise to Address Valid		--	3	ns
52	CKO rise to Address Invalid (hold)		1.2	--	ns
53	CKO rise to CS Assertion		--	4.8	ns
54	CKO rise to CS Negation (hold)		1.2	--	ns
55	CKO rise to $\overline{EB}$ Assertion (Write, EBASS=even)		1.5	4	ns
56	CKO fall to $\overline{EB}$ Assertion (Write, EBASS=odd)		1.5	4	ns
57	CKO rise to $\overline{EB}$ Negation (Write, EBNEG=even)		1.5	4	ns
58	CKO fall to $\overline{EB}$ Negation (Write, EBNEG=odd)		1.5	4	ns
59	CKO rise to $\overline{EB}$ Assertion (Read, EBM=0)		1.5	4	ns
60	CKO rise to $\overline{EB}$ Negation (Read, EBM=0)		1.5	4	ns
61	CKO rise to R/$\overline{W}$ Assertion (Write, EBASS=even)		1.5	4	ns
62	CKO fall to R/$\overline{W}$ Assertion (Write, EBASS=odd)		1.5	4	ns
63	CKO rise to R/$\overline{W}$ Negation (Write, EBNEG=even)		1.5	4	ns
64	CKO fall to R/$\overline{W}$ Negation (Write, EBNEG=odd)		1.5	4	ns
65	CKO rise to R/$\overline{W}$ Assertion (Read)		1.5	4	ns
66	CKO rise to R/$\overline{W}$ Negation (Read)		1.5	4	ns
67	CKO rise to $\overline{OE}$ Assertion (Read, OEASS=even)		1.5	4	ns
68	CKO fall to $\overline{OE}$ Assertion (Read, OEASS=odd)		1.5	4	ns
69	CKO rise to $\overline{OE}$ Negation (Read)		1.5	4	ns
70	CKO rise to $\overline{LBA}$ Assertion ((BEN=0 & LBAEN=1) or BEN=1)		1.5	4	ns
71	CKO rise to $\overline{LBA}$ Negation ((BEN=0 & LBAEN=1) or (Write & BEN=1))		1.5	4	ns
72	CKO rise to $\overline{LBA}$ Negation (Read, BEN=1, IBDLY=odd)		1.5	4	ns
73	CKO fall to $\overline{LBA}$ Negation (Read, BEN=1, IBDLY=even)		1.5	4	ns
74	CKO rise to BURSTCLK rise (Read, BEN=1, IBDLY=even)		1.0	3	ns
75	CKO fall to BURSTCLK rise (Read, BEN=1, IBDLY=odd)		1.0	3	ns
76	CKO rise to BURSTCLK fall (Read, BEN=1, IBDLY=odd)		1.0	3	ns
77	CKO fall to BURSTCLK fall (Read, BEN=1, IBDLY=even)		1.0	3	ns

Table 75-15. AEIM External Bus AC Timing Specifications Relative to CKO (Continued)

Num	Characteristics	Expression	@26 MHz		Unit
			Min	Max	
78	CKO fall to Data-Out (Write, Internal Read in Show Cycles) [see Note 5]		1.0	3.3	ns
79	Data-In Valid to CKO rise (setup) (External Read)		15.0	--	ns
80	CKO rise to Data-In Invalid (hold) (External Read)		0	--	ns
81	ECB Valid to CKO rise (setup)		24.6	--	ns
82	CKO rise to ECB Invalid (hold)		0	--	ns
83	CKO rise to DB_STRB Assertion		1.5	3.5	ns
84	CKO fall to DB_STRB Negation		1.5	3.5	ns
85	CKO rise to DB_SHOW_RW, DB_SZ[1:0] Valid		--	2.5	ns
86	CKO rise to DB_SHOW_RW, DB_SZ[1:0] Invalid (hold)		1.2	--	ns
87	CKO rise to QSTAT[4:0] Valid		--	4.5	ns
88	CKO rise to QSTAT[4:0] Invalid (hold)		1.5	--	ns

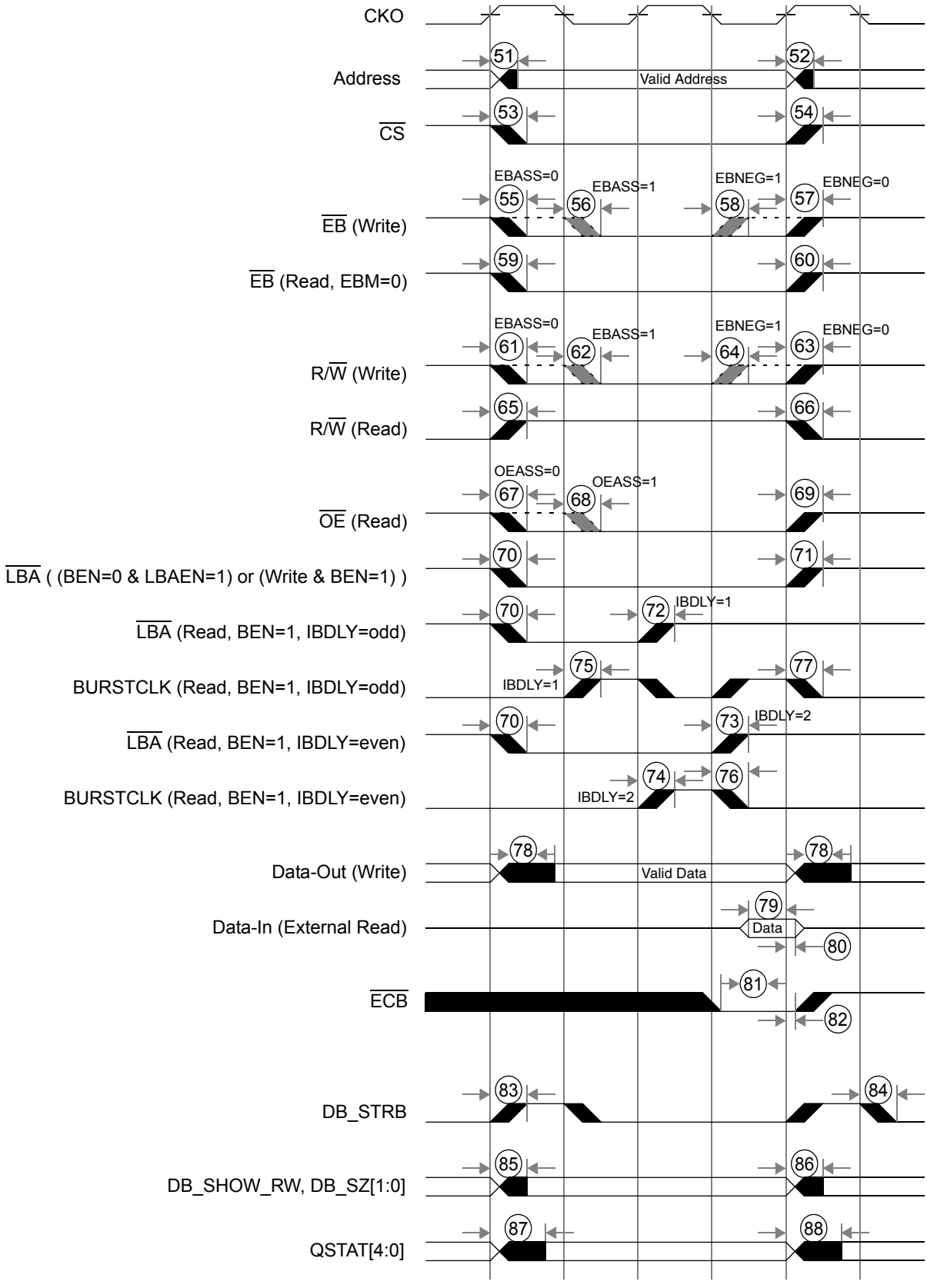


Figure 75-6. AEIM External Bus Timing Relative to CKO

NOTE:

1. All timings are measured at the Neptune IC boundary with respect to the CKO clock output.
2. Min numbers are BCS, -40C, 1.65V.
3. Max numbers are WCS, 105C, 1.35V.
4. Pad behavioral models are programmed for 4mA (or smallest) drive with 9ns WCS delay and 7ns BCS delay.
5. The pad behavioral models do NOT currently support pad output enable/disable delays. The data hold time depends upon pad output disable delay and external loading. So, the number given represents our estimate of 5ns pad output disable delay. Even if the delay is smaller, it would take 10KOhm of leakage for about 100ns to degrade the voltage level to fail V_{IH}/V_{IL} limits
6. $\overline{EB}$ outputs are asserted for Reads if the EBM bit in the corresponding CS control register is clear.
7. R/$\overline{W}$ output follows $\overline{EB}$ outputs timing for Writes.
8. aeim_wr_oe outputs are used to control Data pads output enable, and hence, are not shown as they are internal to Neptune chip.
9. DB_STRB, DB_SHOW_RW, DB_SZ[1:0] and QSTAT[4:0] outputs are only used for emulation and real-time trace support.

75.3.7 Subscriber Interface Module Timings

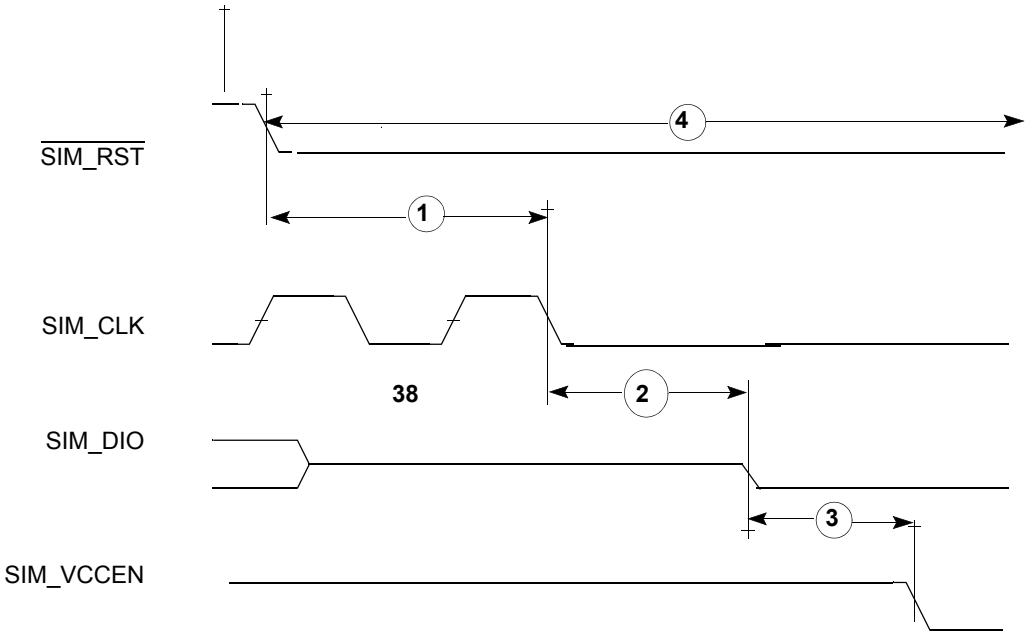
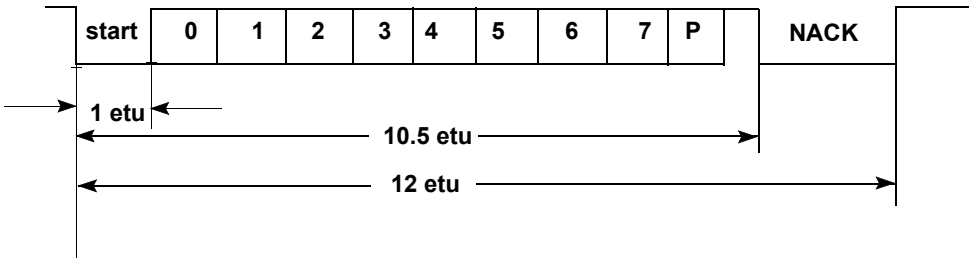


Figure 75-7. Subscriber Interface Module Power Down AC Timing



1 etu = 372/f or 64/f where f = 26Mhz/4 (for 3V) or 26Mhz/5 (for 1.8V).

Figure 75-8. Subscriber Interface Data Format

Table 75-16. Subscriber Interface Module Timings

Num	Characteristics	Expression	@26 MHz		Unit
			Min	Max	
1	$\overline{\text{SIM\_RST}}$ Low to SIM_CLK Low		1.18	200/f	us
2	SIM_CLK Deactivated to SIM_DIO Tristate to Low		1.18	200/f	us
3	SIM_DIO Low to SIM_VCCEN Low		1.18	200/f	us
4	$\overline{\text{SIM\_RST}}$ Low		40000/f	-	ns
	SENSE High to $\overline{\text{SIM\_RST}}$ Low		57	76	us

75.3.8 MQSPI Timing Specifications

Refer to Section 49.3.4.5, “SPI Receive Operation Timing Considerations,” on page 49-13 for design constraints on SPI round trip time.

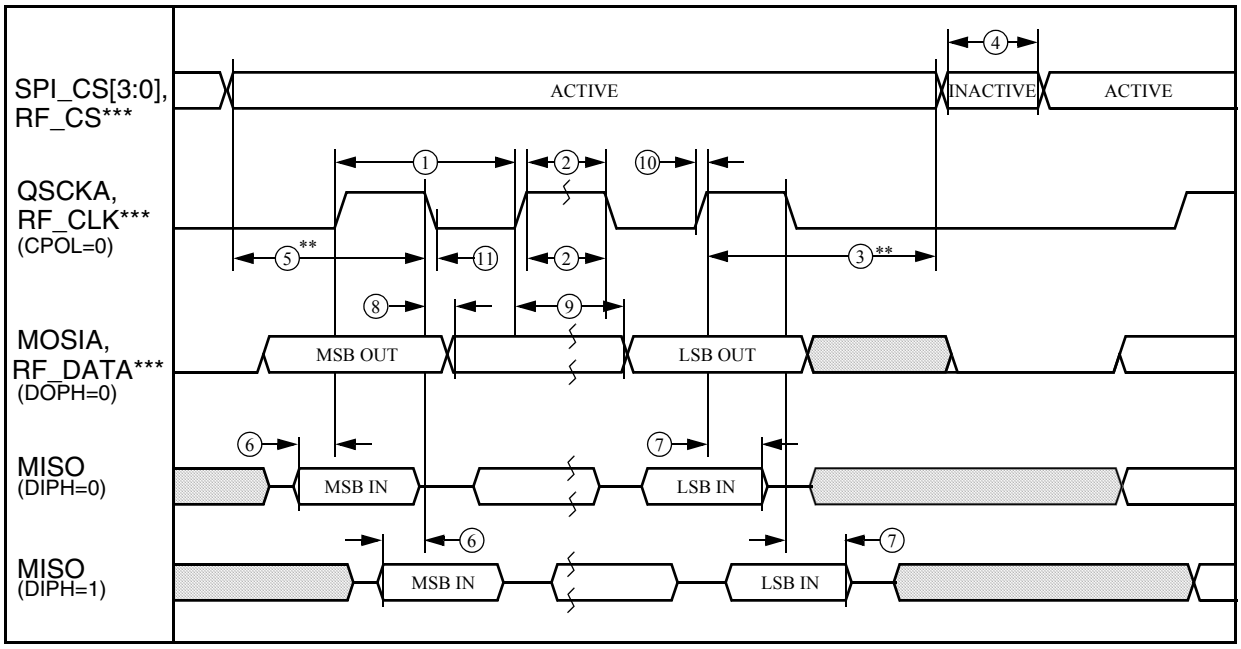


Figure 75-9. MQSPI Timing Specifications (CPOL=0, DOPH=0)

** Chip Select lead/lag time is defined in relation to the edge of QSCK where the slave device is expected to latch data on MOSI.

***During Fast digital DC Adapt period, the DCADAPT module takes over RF_CLK, RF_DATA, and RF_CS. See Chapter 28, "Digital Fast DC Adapt (DCADAPT)."

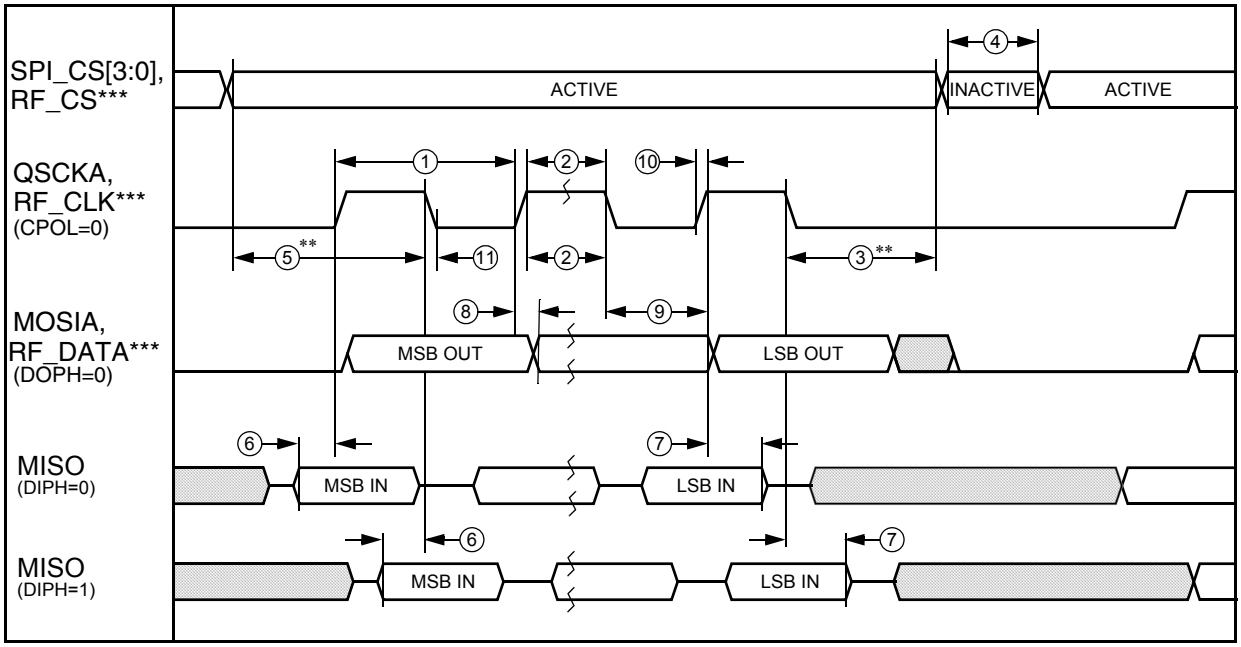


Figure 75-10. MQSPI Timing Specifications (CPOL=0, DOPH=1)

* Chip Select lead/lag time is defined in relation to the edge of QSCK where the slave device is expected to latch data on MOSI.

***During Fast digital DC Adapt period, the DCADAPT module takes over RF_CLK, RF_DATA, and RF_CS. See Chapter 28, "Digital Fast DC Adapt (DCADAPT)."

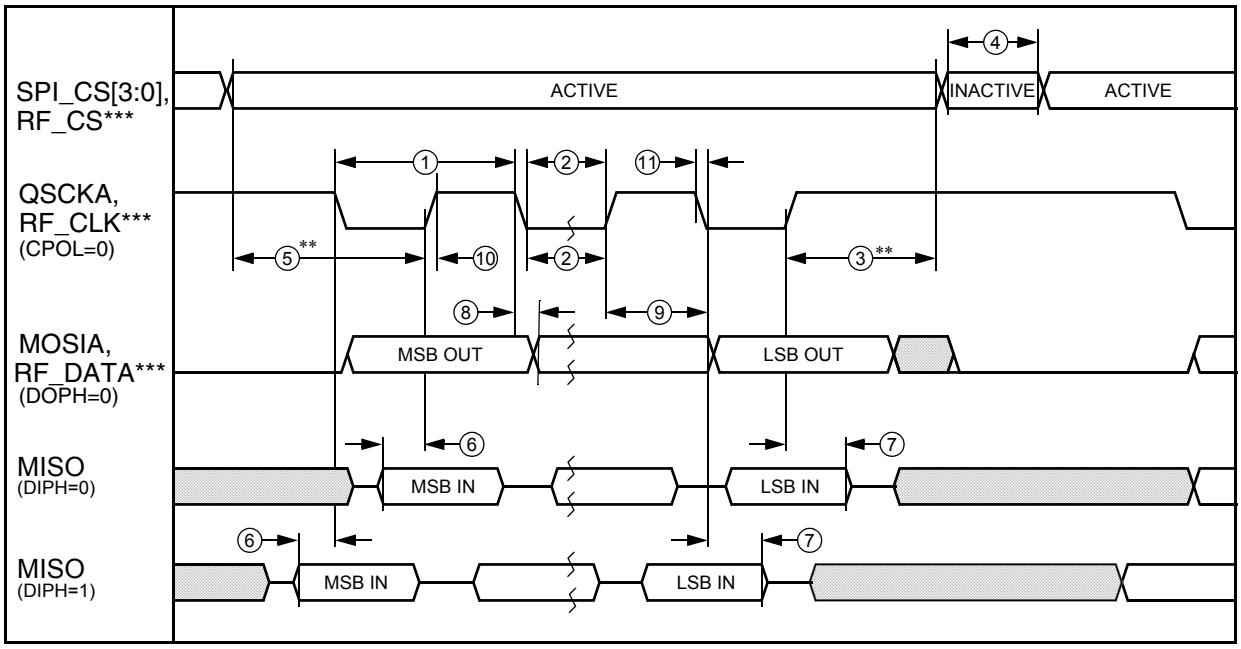


Figure 75-11. *MQSPI Timing Specifications (CPOL=1, DOPH=0)

** Chip Select lead/lag time is defined in relation to the edge of QSCK where the slave device is expected to latch data on MOSI.
***During Fast digital DC Adapt period, the DCADAPT module takes over RF_CLK, RF_DATA, and RF_CS. See Chapter 28, "Digital Fast DC Adapt (DCADAPT)."

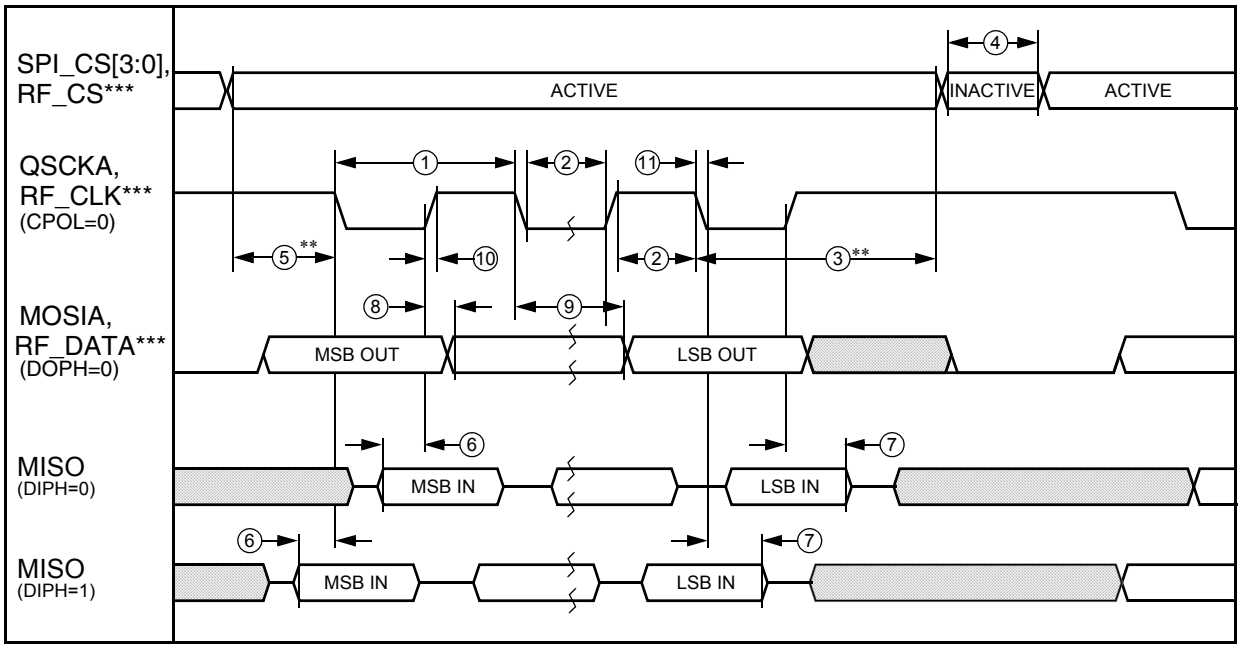


Figure 75-12. MQSPI Timing Specifications (CPOL=1, DOPH=1)

** Chip Select lead/lag time is defined in relation to the edge of QSCK where the slave device is expected to latch data on MOSI.
***During Fast digital DC Adapt period, the DCADAPT module takes over RF_CLK, RF_DATA, and RF_CS. See Chapter 28, "Digital Fast DC Adapt (DCADAPT)."

Table 75-17. MQSPI Timing

Num	Characteristics	Symbol	Expression	Min	Max	Unit
1	Cycle time *	T_{QCYC}		1	504	T_{REF}
2	Clock (SCK) High or Low Time*	T_{SW}		0.5	252	T_{REF}
3	Chip Select Lag Time*	T_{LAG}		1.5	-	T_{REF}
4	Inter-Message Transfer Delay**	T_{TD}		1	∞	T_{QCYC}
5	Chip Select Lead Time*	T_{LEAD}		1	128	T_{QCYC}
6	Data Setup Time (Inputs, See note 1)	T_{SU}		45	-	ns
7	Data Hold Time (Inputs, See note 1)	T_{HI}		6	-	ns
8	Data Valid (After SCK edge)	T_V		-	5	ns
9	Data Hold Time (Outputs)	T_{HO}		0	5	ns

T_{REF} is the period of the PAT_REF clock
Refer to EIM for rise and fall times.
* Timing dependant on Software programable delay.
** Inter Message Transfer Delay is dependant on operation.

75.3.9 USB Timing Specifications

Refer to Chapter 21, “Universal Serial Bus Interface (USB),” for further information.

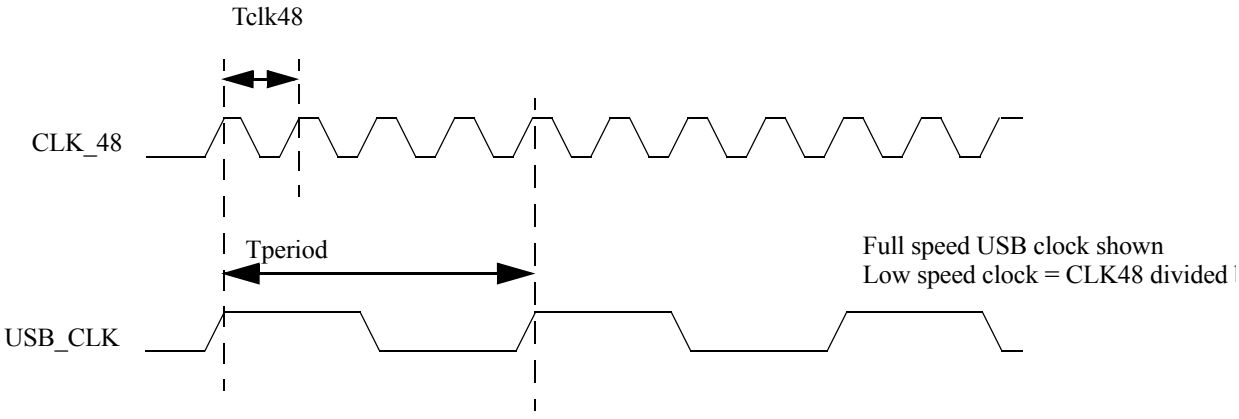


Figure 75-13. USB Clocking

Table 75-18. USB Clocking

Parameter	Symbol	Notes	Min	Typ	Max	Units
CLK48	Tclk48	Derived from USB PLL		48		Mhz
Full-speed Data Rate	Tfdrate	USB full speed data rate	11.97		12.03	Mb/s
Frame Data Rate	Tframe	Rate of interrupt for Start of Frame	0.9995		1.005	ms
Frame Tracking Range	Trange	SOF tracking range	-60		60	Tperiod

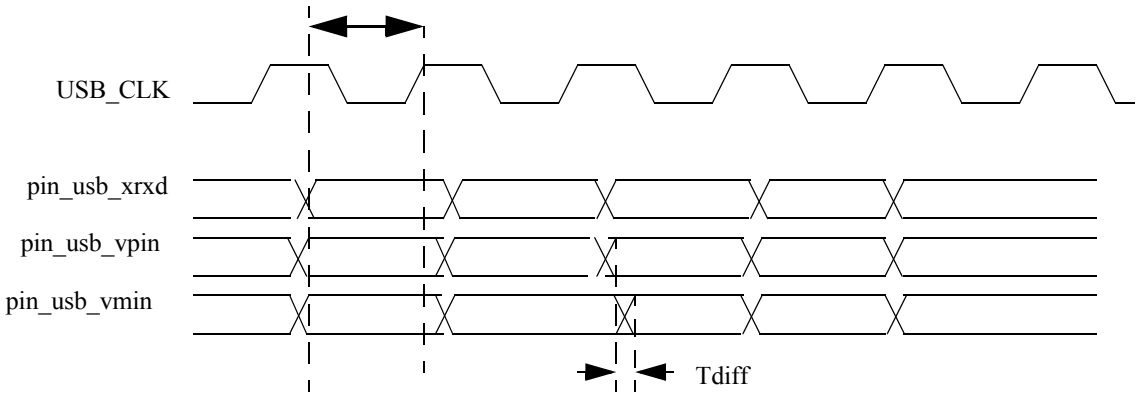


Figure 75-14. USB Input Timings

Table 75-19. USB Input Timings

Parameter	Symbol	Notes	Min	Type	Max	Units
Receive Setup	Trsu	Asynchronous Interface				ns
Full speed Differential	Tdiff	See section 7.1.4 of USB 1.1 Specification		0		ns

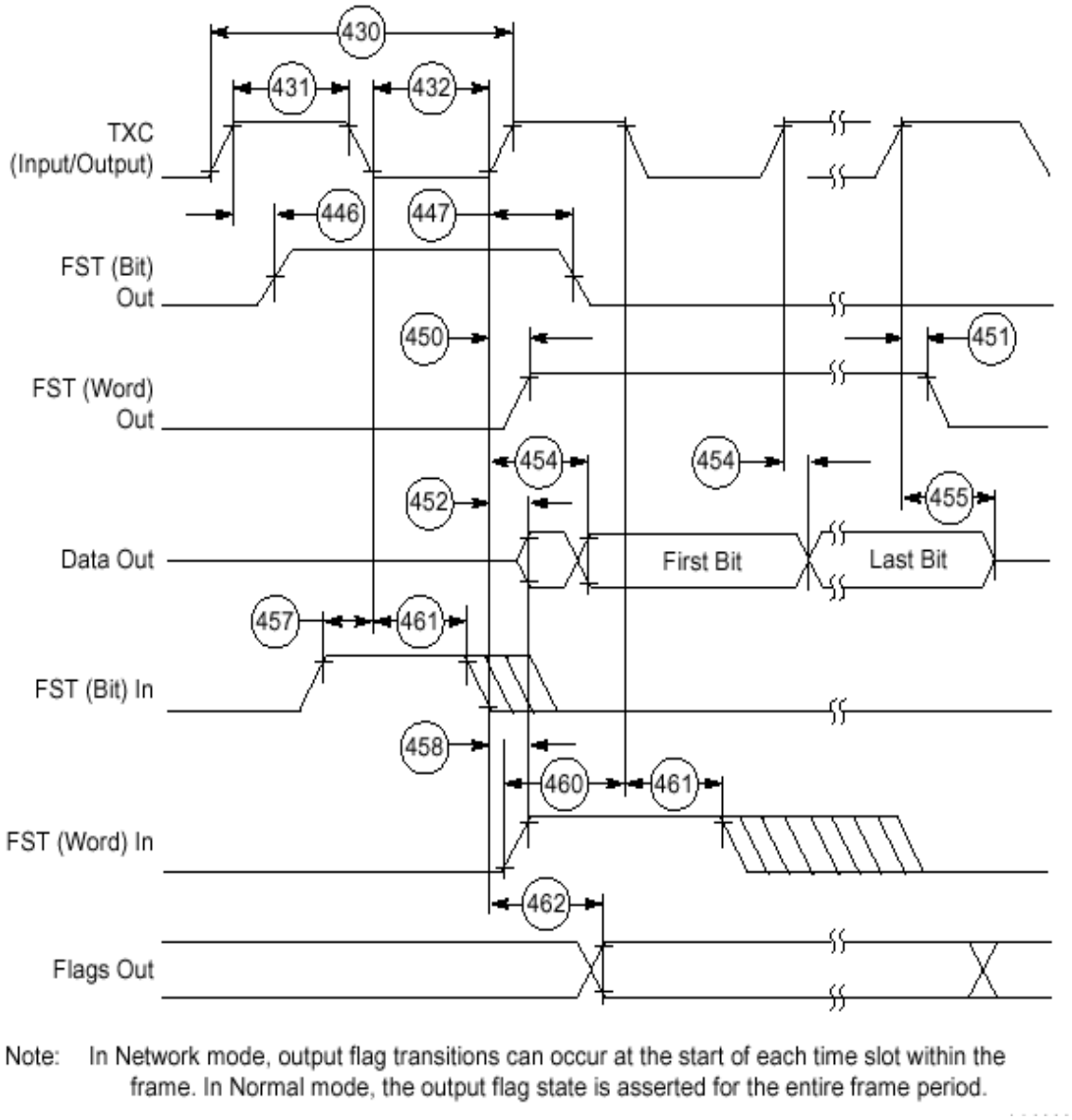
75.3.10 BBP and SAP Timing Specifications

Table 75-20. BBP Timing Specification

Case	Meaning
t_{SSICC}	SSI clock cycle time
TXC	Transmit Clock (on SCK pin)
RXC	Receive clock (on SC0 or SCK pin)
FST	Transmit frame sync (on SC2 pin)
FSR	Receive frame sync (SC1 or SC2 pin)
i ck	Internal Clock
x ck	External clock
i ck	Internal clock, Asynchronous mode (Asynchronous implies that TXC and RXC are two different clocks)
i ck	Internal clock, Synchronous mode (Synchronous implies that TXC and RXC are the same clock)
bl	Bit length
wl	Word length
wr	Word length relative

The following are the assumptions made during the calculations:

1. All clocks are ideal.
2. The delays of pad are calculated using the assumption of 25pf load, 1.5ns input transition using the .lib models.
3. GPIO delays are calculated using prelayout netlist and ideal clocks.
4. The starting reference for all calculations are boundary clock signals.
5. All time are calculate assuming that the posedge of clock will asserted at boundry pin of chip at time t=0 ns.
6. Negative setup time mean clock path delay is more than data path delay (i.e.data can be launch t time unit later, after the assertion of posedges of clock at SCKB pin).
7. Negative hold time mean data path delay is more than clock path delay (i.e.data should not change till t time unit before the clock edge assertion as SCKB).
8. The BBP supports bit clock = 4*tdc but max bit clock frequency is 22.8 Mhz .



Note: In Network mode, output flag transitions can occur at the start of each time slot within the frame. In Normal mode, the output flag state is asserted for the entire frame period.

Figure 75-15. BBP/SAP Transmitter Timing

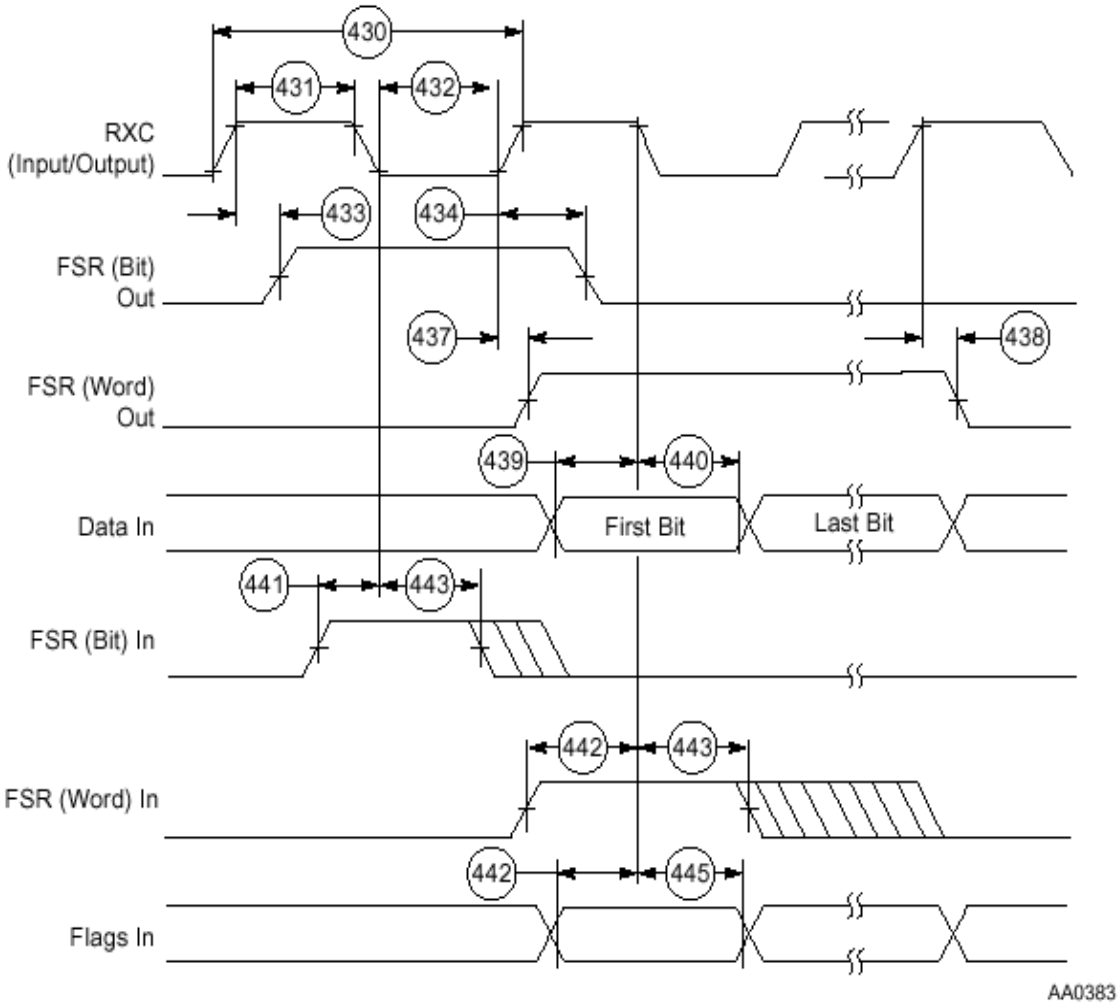


Figure 75-16. BBP/SAP Receiver Timing

Table 75-21. BBP Timing

Num	Characteristics	Symbol	Expression	130 MHz DSP clock		Case	Unit
				Min	Max		
430	Clock Cycle	t_{SSICC}	4x T_{DC} 4 x T_{DC}	43.8 43.8		x ck i ck	ns ns
431	Clock high period		2 x T_{DC} 2 x $T_{DC - 0.88}$	21.9 21.02		x ck i ck	ns ns
432	Clock low period		2 x T_{DC} 2 x $T_{DC - 0.88}$	21.9 21.02		x ck i ck	ns ns
433	RXC rising edge to FSR Out (bl) high				21.17 1.68	x ck i ck	ns ns

Table 75-21. BBP Timing (Continued)

Num	Characteristics	Symbol	Expression	130 MHz DSP clock		Case	Unit
				Min	Max		
434	RXC rising edge to FSR out (bl) low				20.76 1.27	x ck i ck	ns ns
437	RXC rising edge to FSR out (wl) high				21.16 1.52	x ck i ck	ns ns
438	RXC rising edge to FSR out (wl) low				20.75 1.10	x ck i ck	ns ns
439	Data in setup time before RXC (SCK in Synchronous mode) falling edge				-3.9 15.89	x ck i ck	ns ns
440	Data in hold time after RXC falling edge				4.37 -15.42	x ck i ck	ns ns
441	FSR input (bl, wr) high before RXC falling edge				-5.20 15.38	x ck i ck	ns ns
442	FSR input (wl) high before RXC falling edge				-5.2 15.38	x ck i ck	ns ns
443	FSR input hold time after RXC falling edge				5.42 -13.92	x ck i ck	ns ns
444	Flags input setup before RXC falling edge				-4.76 12.77	x ck i ck	ns ns
445	Flags input hold time after RXC falling edge				4.83 -12.92	x ck i ck	ns ns
446	TXC rising edge to FST out (bl) high				21.17 1.27	x ck i ck	ns ns
447	TXC rising edge to FST out (bl) low				20.76 1.27	x ck i ck	ns ns
450	TXC rising edge to FST out (wl) high				21.16 1.52	x ck i ck	ns ns
451	TXC rising edge to FST out (wl) low				20.75 1.10	x ck i ck	ns ns
452	TXC rising edge to data out enable from high impedance				18.46 -0.20	x ck i ck	ns ns
454	TXC rising edge to data out valid				22.33 2.63	x ck i ck	ns ns
455	TXC rising edge to data out high impedance				18.46 -0.2	x ck i ck	ns ns

Table 75-21. BBP Timing (Continued)

Num	Characteristics	Symbol	Expression	130 MHz DSP clock		Case	Unit
				Min	Max		
457	FST input (bl, wr) setup time before TXC falling edge				-3.9 15.89	x ck i ck	ns ns
458	FST input (wl) to data out enable from high impedance				13.96		ns ns
460	FST input (wl) setup time before TXC falling edge				-5.20 15.38	x ck i ck	ns ns
461	FST input hold time after TXC falling edge				5.42 -13.92	x ck i ck	ns ns
462	Flag output valid after TXC rising edge				not aplicable as pins are input only		

Table 75-22. Key To Table 2 SAP Timing

Case	Meaning
t_{SSICC}	SSI clock cycle time
TXC	Transmit Clock (on SCK pin)
RXC	Receive clock (on SC0 or SCK pin)
FST	Transmit frame sync (on SC2 pin)
FSR	Receive frame sync (SC1 or SC2 pin)
i ck	Internal Clock
x ck	External clock
i ck	Internal clock, Asynchronous mode (Asynchronous implies that TXC and RXC are two different clocks)
i ck	Internal clock, Synchronous mode (Synchronous implies that TXC and RXC are the same clock)
bl	Bit length
wl	Word length
wr	Word length relative

The following are the assumptions made during the calculations:

1. All clocks are ideal.
2. The delays of pad are calculated using the assumption of 25pf load, 1.5ns input transition using the .lib models.
3. GPIO delays are calculated using prelayout netlist and idela clcoks.
4. The starting reference for all calculations are boundary clock signals.
5. Negative hold times for some configurations is based on calculation of hold time requirement of data reaching flops inside the block with respect to clock. Since the clock travels faster than data, the hold time at the boundary is negative for some configurations.

Table 75-23. SAP Timing

Num	Characteristics	Symbol	Expression	130 MHz DSP clock		Case	Unit
				Min	Max		
430	Clock Cycle	t_{SSICC}	$6 \times T_{DC}$[1] $6 \times T_{DC}$	43.8 43.8		x ck i ck	ns ns
431	Clock high period		$3 \times T_{DC}$ $3 \times T_{DC\ -0.88}$	21.9 21.02		x ck i ck	ns ns
432	Clock low period		$3 \times T_{DC}$ $3 \times T_{DC-\ 0.88}$	21.9 21.02		x ck i ck	ns ns
433	RXC rising edge to FSR Out (bl) high				18.56 4.21	x ck i ck	ns ns
434	RXC rising edge to FSR out (bl) low				18.55 4.21	x ck i ck	ns ns
437	RXC rising edge to FSR out (wl) high				18.37 4.02	x ck i ck	ns ns
438	RXC rising edge to FSR out (wl) low				18.53 4.18	x ck i ck	ns ns
439	Data in setup time before RXC (SCK in Synchronous mode) falling edge			1.03 14.15		x ck i ck	ns ns
440	Data in hold time after RXC falling edge			1.51 -14.51		x ck i ck	ns ns
441	FSR input (bl, wr) high before RXC falling edge			0.81 14.53		x ck i ck	ns ns
442	FSR input (wl) high before RXC falling edge			0.81 14.53		x ck i ck	ns ns
443	FSR input hold time after RXC falling edge			1.33 -14.14		x ck i ck	ns ns
444	Flags input setup before RXC falling edge			-1.18 13.89	--- ---	x ck i ck	ns ns
445	Flags input hold time after RXC falling edge			1.87 -13.24	--- ---	x ck i ck	ns ns
446	TXC rising edge to FST out (bl) high				20.17 4.12	x ck i ck	ns ns
447	TXC rising edge to FST out (bl) low				20.16 4.12	x ck i ck	ns ns
450	TXC rising edge to FST out (wl) high				19.98 4.02	x ck i ck	ns ns

Table 75-23. SAP Timing (Continued)

Num	Characteristics	Symbol	Expression	130 MHz DSP clock		Case	Unit
				Min	Max		
451	TXC rising edge to FST out (wl) low				20.14 4.18	x ck i ck	ns ns
452	TXC rising edge to data out enable from high impedance				19.48 3.27	x ck i ck	ns ns
454	TXC rising edge to data out valid				18.14 2.18	x ck i ck	ns ns
455	TXC rising edge to data out high impedance				19.36 2.62	x ck i ck	ns ns
457	FST input (bl, wr) setup time before TXC falling edge			14.92 0.05		x ck i ck	ns ns
458	FST input (wl) to data out enable from high impedance				15.97	x ck i ck	ns ns
460	FST input (wl) setup time before TXC falling edge				14.92 0.05	x ck i ck	ns ns
461	FST input hold time after TXC falling edge			0.43 -15.45		x ck i ck	ns ns
462	Flag output valid after TXC rising edge			--- ---	sc0a_o and sc1a_o (flags) are not connected to any output in neptune lts.	x ck i ck	ns ns

1. For DSP clock less that 93 MHz, t_{SSICC} expression may be set to 4 x T_{DC}.

75.3.11 DMAC Timing Specifications

For DMAC electrical timings please refer to Section 59.6.1, “Serial Interface,” on page 59-33 and Section 59.6.2, “Parallel Interface,” on page 59-35.

75.3.12 JTAG Port

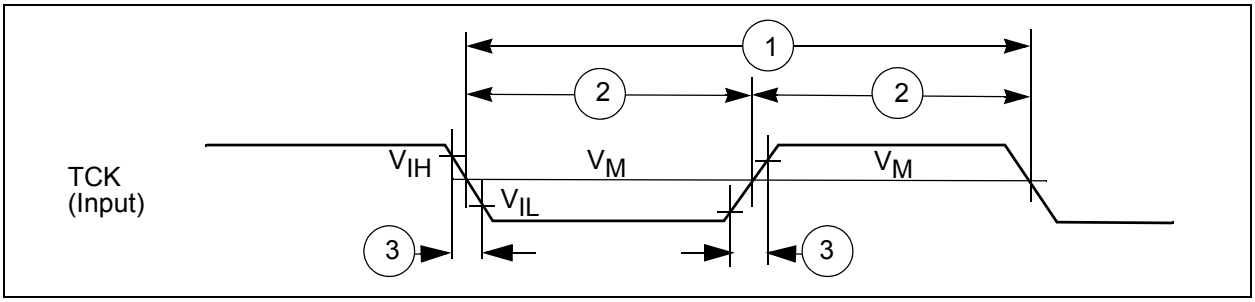


Figure 75-17. Test Clock Input Timing Diagram

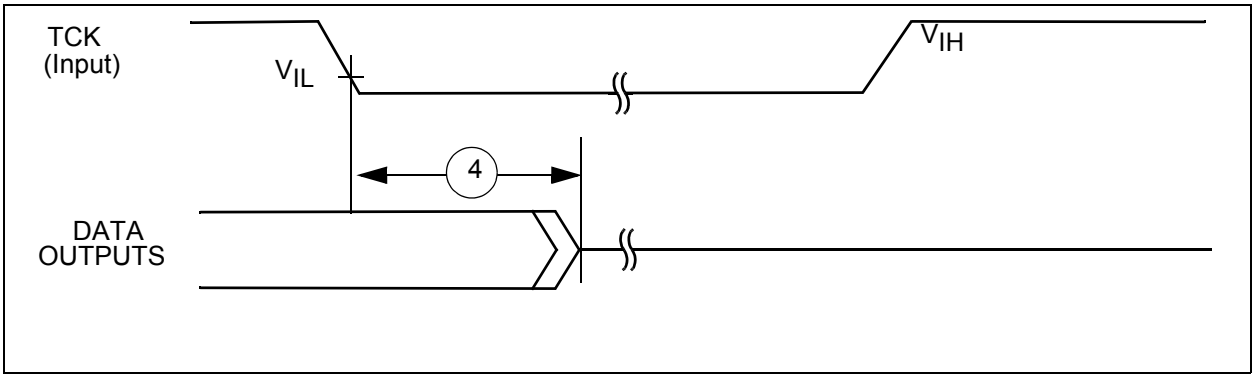


Figure 75-18. HIGHZ (JTAG) Timing Diagram

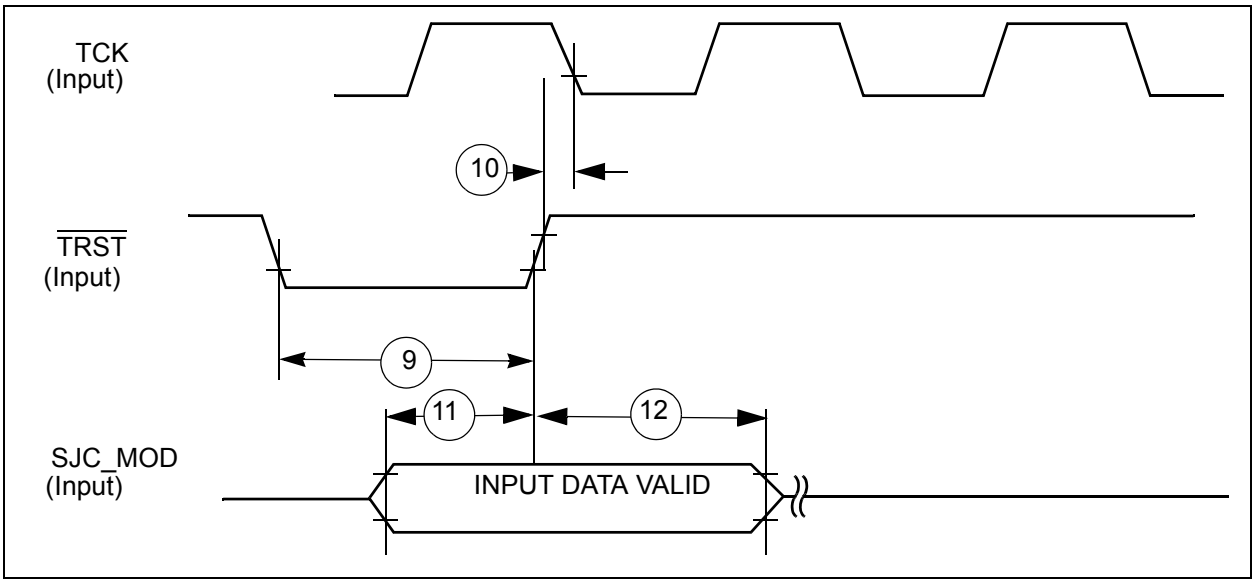


Figure 75-19. $\overline{\text{TRST}}$ Timing Diagram

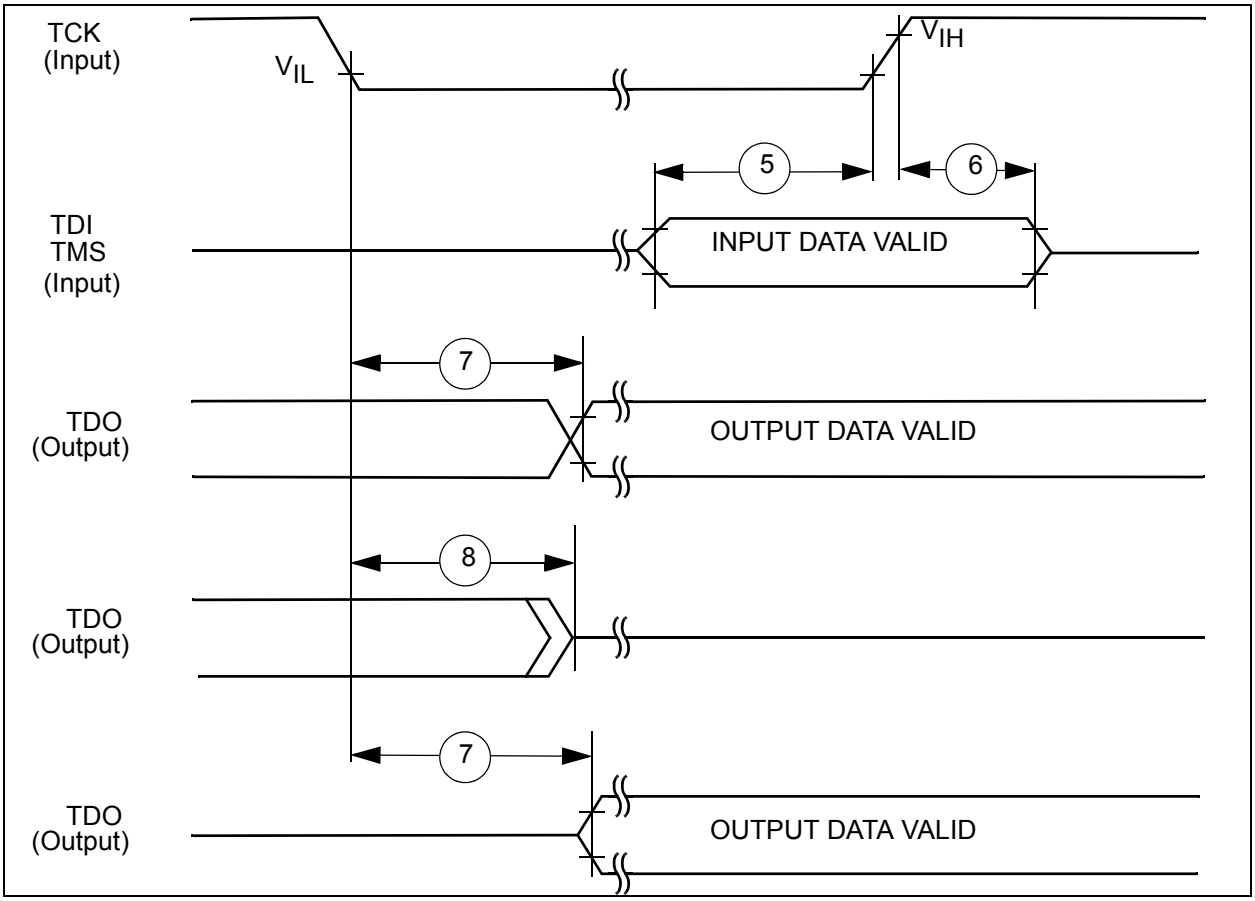


Figure 75-20. Test Access Port Timing Diagram

Table 75-24. JTAG Timing

Num	Characteristics	Symbol	Expression	@ 52 MHz		Unit
				Min	Max	
1	TCK Frequency of Operation	F_{tck}	$F_{mcu}/8$	0	6.5	MHz
2	TCK Clock Pulse Width Measured at 1.5V			77.0	—	ns
3	TCK Rise and Fall Times			2.5	8.0	ns
4	TCK Low to Output High Z			0	40.0	ns
5	TMS, TDI Data Setup Time			4.0	—	ns
6	TMS, TDI Data Hold Time			62.0	—	ns
7	TCK Low to TDO Data Valid			0	68.0	ns
8	TCK Low to TDO High Z			0	68.0	ns
9	$\overline{\text{TRST}}$ Assert Time			100.0	—	ns
10	$\overline{\text{TRST}}$ Setup Time to TCK Low			40.0	—	ns
11	SJC_MOD Setup Time to TRST High			40.0	—	ns

Table 75-24. JTAG Timing (Continued)

Num	Characteristics	Symbol	Expression	@ 52 MHz		Unit
				Min	Max	
12	SJC_MOD Hold Time to TRST High			40.0	—	ns

75.4 Analog Module Electrical Specifications

75.4.1 Receiver Analog Front-End Electrical Specifications

Please see Section 26.5, “RxAFE VAG Buffer,” on page 26-6 for RxAFE VAG Buffer Specifications.

Please see Section 26.6, “RxAFE Reference Voltage Generator,” on page 26-7 for RxAFE Reference Voltage Generator Specifications.

Please see Section 26.7, “Baseband Receive I/Q A/Ds,” on page 26-8 for RxAFE Baseband Receive I/Q A/D Specifications.

75.4.2 Transmitter & Receiver Synthesizer Electrical Specifications

Please see Section 30.6.2, “Power Supply Specification,” on page 30-24, Section 30.6.3, “Main Divider Specification,” on page 30-25, and Section 30.6.6, “Phase Detector and Charge Pump Specification,” on page 30-29.

75.4.3 Transmitter Dual Port D/A Electrical Specifications

Please see Section 30.5.3, “Dual-Port Specifications,” on page 30-19.

75.4.4 Power Amplifier Control Electrical Specifications

Please see Section 31.5, “PAC Analog Blocks,” on page 31-20. Specifications for the analog sub-blocks are available.

75.4.5 Voice Coder Electrical Specifications

Please see Section 25.5, “Voice Codec Electrical Specifications,” on page 25-47.

75.4.6 General Purpose A/D Converter Electrical Specifications

Please see Section 35.3, “General Purpose ADC Electrical Specification,” on page 35-7.

Chapter 76
Mechanical Specifications (PKG)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	06/17/02	Scott King	Updated information on 225 package. Added placeholders for other packages
0.2	07/18/02	Mark Babcock	Add package outline and pin assignments for 280-pin Production and Development footprints. Updated per DDTS# DSPhl15006.
0.3	08/20/02	Shannon Osgood	Updated per DDTS# DSPhl15061. Replaced 225 drawing.
0.4	7 Nov 2002	Mark Babcock	Fix typo errors, adjust format. Add note for Neptune LTE and Neptune ULS.
0.5	05/07/03		Updated for LTE specification release.

76.1 MAPBGA Packages

The Neptune LTE device will be available in the following packages:

- 280 MAPBGA 13x13mm 0.65 mm pitch package for production.
- 280 MAPBGA 13x13mm 0.65 mm pitch package for development.

76.2 280 MAPBGA 13X13mm 0.65mm pitch for Production.

Reference Figure 76-1 package drawings and dimensions for the 280 MBGA Production package.
Reference Table 76-1 for ball pad to signal name.

76.2.1 280 MAPBGA Package Diagrams (13 x 13 mm)

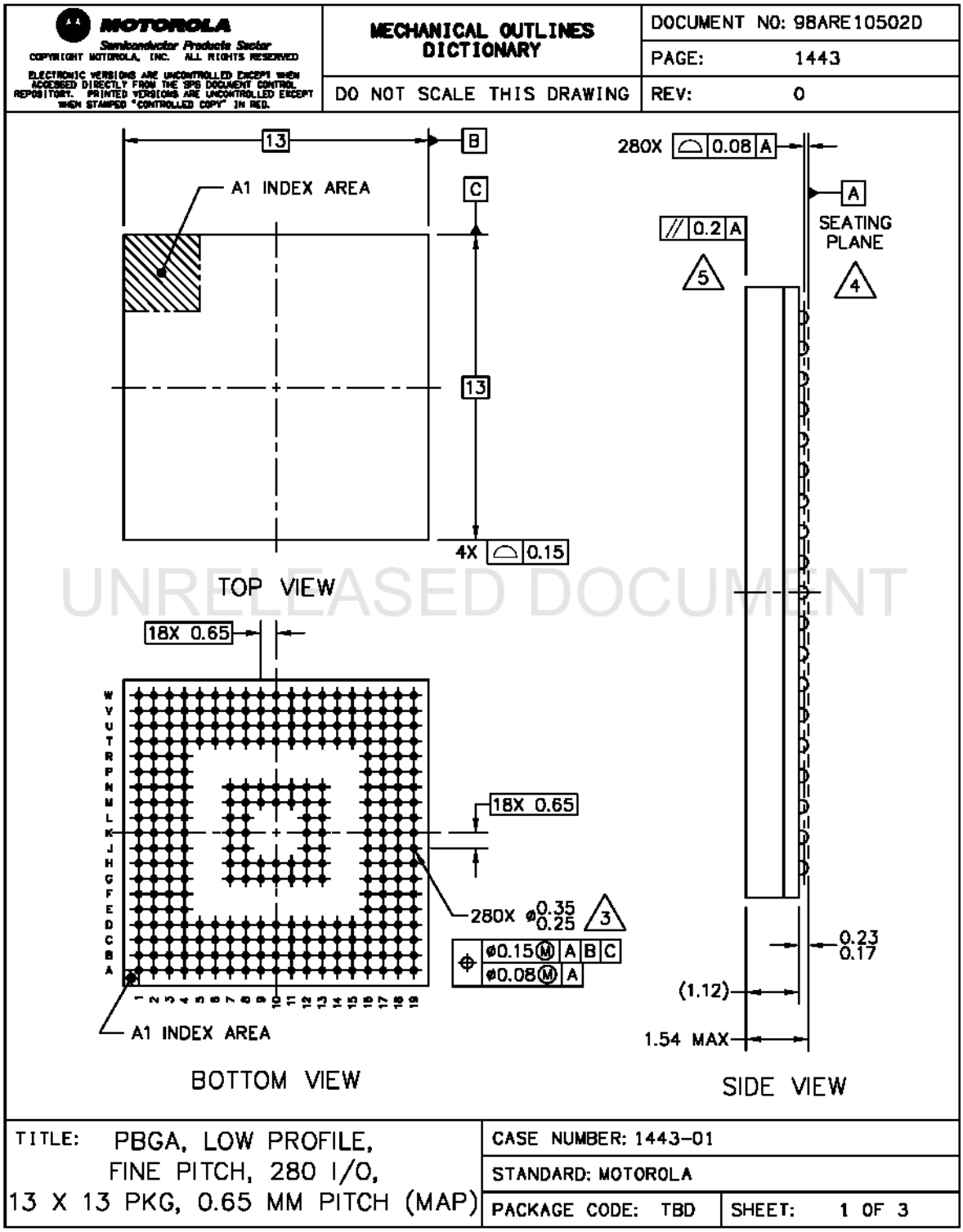


Figure 76-1. Mechanical Drawing MAPBGA 280

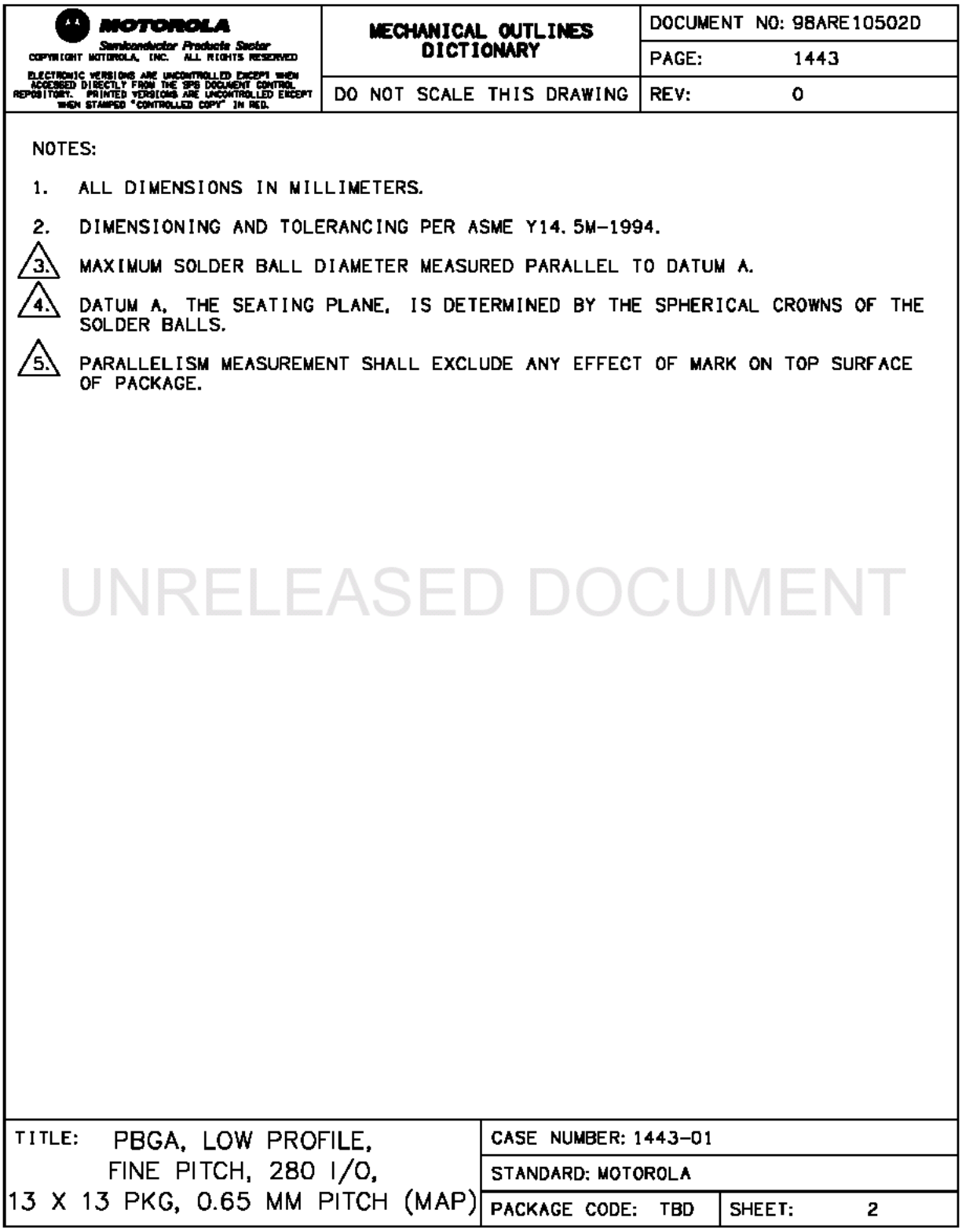


Figure 76-2. Mechanical Drawing MAPBGA 280, Notes

76.2.2 280 MAPBGA Production Package Pinout

Table 76-1. Neptune LTE 280 Production MBGA Ball Pad to Signal Name Net List

Pin	Pin Name	Pin	Pin Name	Pin	Pin Name
A1	VSS_CP	**B5**	TRKG_OSC_OUT	**C9**	VAG_RX
A2	VSS_CP	**B6**	VCO1_P	**C10**	VSS_TXRX
A3	VSSA	**B7**	REG_BYP_TXRX	**C11**	VDD(3)
A4	XTAL	**B8**	RX_IX	**C12**	ATST_P
A5	REG_BYP_PRE	**B9**	RX_QX	**C13**	INT4
A6	VCO1_N	**B10**	PA_DET	**C14**	INT0
A7	REF_HI	**B11**	VSS(3)	**C15**	USB_VPOUT
A8	RX_I	**B12**	SRDA	**C16**	USB_VMIN
A9	RX_Q	**B13**	STDA	**C17**	CVDD(2)
A10	PA_OUT	**B14**	INT1	**C18**	URTS1
A11	BVDD	**B15**	PC14	**C19**	URXD1
A12	SC2A	**B16**	USB_XRXD	**D1**	VAG_CODEC
A13	SC1A	**B17**	USB_SUSPEND	**D2**	LVSS
A14	PE12	**B18**	CVSS(2)	**D3**	ADC_P
A15	SJC_MOD	**B19**	CVSS(2)	**D4**	TX_CP
A16	USB_VMOUT	**C1**	VAGOUT_CODEC	**D5**	RX_CP
A17	USB_VPIN	**C2**	VSS_CODEC	**D6**	REF_CODEC_N
A18	CVSS(2)	**C3**	DAC_N	**D7**	VDD_CODEC
A19	CVSS(2)	**C4**	VDD_CP	**D8**	TX_MOD
B1	VSS_CP	**C5**	VDDA	**D9**	VDD_TXRX
B2	VSS_CP	**C6**	REF_CODEC_P	**D10**	REF_LO
B3	CP_BIAS	**C7**	VSS_PRE	**D11**	ATST_N
B4	EXTAL	**C8**	VSS_TXRX(3)	**D12**	PA_REF

Table 76-1. Neptune LTE 280 Production MBGA Ball Pad to Signal Name Net List (Continued)

Pin	Pin Name	Pin	Pin Name	Pin	Pin Name
D13	SCKA	G1	ROW5	H11	NC_H9
D14	PC12	G2	ROW7	H12	NC_H9
D15	USB_TXENB	G3	COLUMN0	H13	REG_BYP_CODEC
D16	CTS2	G4	ROW2	H16	AVSS(6)
D17	UCTS1	G7	JVSS	H17	AVDD(5)
D18	SPI_CS6	G8	STANDBY	H18	NC_H18
D19	INT5	G9	ADC_N	H19	A9
E1	ADC_DATA	G10	PE11	J1	FVSS
E2	LVDD	G11	PE10	J2	VDD(4)
E3	CKIL	G12	SC0A	J3	TEST0
E4	DAC_P	G13	NC_H9	J4	SIM_RST
E16	BVSS	G16	NC_J12	J7	ROW4
E17	UTXD1	G17	EB1*	J8	NC_H8
E18	AVDD(6)	G18	CS4*	J12	NC_J12
E19	A15	G19	A10	J13	NC_J12
F1	ROW1	H1	JVDD	J16	A8
F2	COLUMN1	H2	ROW6	J17	A19
F3	ROW0	H3	TEST1	J18	A16
F4	LOW_BATT*	H4	VSS(4)	J19	R/W*
F16	A14	H7	ROW3	K1	TEST2
F17	A13	H8	NC_H8	K2	FVDD
F18	A12	H9	NC_H9	K3	SIM_DIO
F19	A11	H10	NC_H9	K4	CVSS(3)

Table 76-1. Neptune LTE 280 Production MBGA Ball Pad to Signal Name Net List (Continued)

Pin	Pin Name	Pin	Pin Name	Pin	Pin Name
K7	MOD	**M4**	LCD_CLK	**N16**	AVSS(3)
K8	NC_H8	**M7**	D0	**N17**	RXD2
K12	NC_J12	**M8**	NC_H8	**N18**	ECB*
K13	NC_J12	**M9**	NC_M9	**N19**	QVSS
K16	EB0*	**M10**	NC_M9	**P1**	SDATA
K17	CS3*	**M11**	NC_M9	**P2**	LCD_RS
K18	A18	**M12**	NC_M9	**P3**	CVDD(3)
K19	AVSS(5)	**M13**	CS5	**P4**	D11
L1	SIM_CLK	**M16**	A5	**P16**	CKO
L2	LCD_DATA[1]	**M17**	QVDD	**P17**	NC_P17
L3	LCD_DATA[0]	**M18**	REG_BYP_CORE	**P18**	AVDD(3)
L4	LCD_DATA[4]	**M19**	AVSS(4)	**P19**	A7
L7	A20	**N1**	LCD_DATA[3]	**R1**	SIM_PD
L8	NC_H8	**N2**	LCD_DATA[5]	**R2**	AVSS(7)
L12	NC_J12	**N3**	LCD_CS	**R3**	A22
L13	NC_J12	**N4**	A21	**R4**	AVSS(1)
L16	LBA*	**N7**	D12	**R16**	AVDD(2)
L17	VDD(2)	**N8**	D15	**R17**	A6
L18	VSS(2)	**N9**	TOUT10	**R18**	A17
L19	AVDD(4)	**N10**	SPI_CS8	**R19**	A4
M1	SIM_VCCEN	**N11**	DVDD	**T1**	D3
M2	LCD_DATA[2]	**N12**	TRST	**T2**	D10
M3	A23	**N13**	TXD2	**T3**	D9

Table 76-1. Neptune LTE 280 Production MBGA Ball Pad to Signal Name Net List (Continued)

Pin	Pin Name	Pin	Pin Name	Pin	Pin Name
T4	D13	U8	RF_DATA	V12	MOSIA
T5	A24	U9	STDB	V13	RESET_IN*
T6	GSM*_DCS	U10	TOUT12	V14	EBW*
T7	TOUT9	U11	CTS1	V15	TMS
T8	MISOB	U12	SPI_CS1	V16	RTS2
T9	SCKB	U13	WDOG*	V17	CS1*
T10	INT6	U14	TDO	V18	CS0*
T11	QSCKA	U15	MCU_DE*	V19	CS0*
T12	SPI_CS0	U16	CVDD(1)	W1	AVDD(7)
T13	SPI_CS2	U17	A0	W2	AVDD(7)
T14	CVSS(1)	U18	A1	W3	D5
T15	RTCK	U19	A2	W4	D7
T16	OE*	V1	AVDD(7)	W5	RESET_OUT*
T17	A3	V2	AVDD(7)	W6	VDD(1)
T18	CS2*	V3	D4	W7	TOUT8
T19	BURSTCLK	V4	D14	W8	INT3
U1	D2	V5	AVDD(1)	W9	RF_CS
U2	D1	V6	RX_EN_OUT	W10	SC2B
U3	D8	V7	RF_CLK	W11	TOUT5
U4	D6	V8	TOUT11	W12	SPI_CS3
U5	VSS(1)	V9	EVDD	W13	ADC_SYNC
U6	TOUT7	V10	DVSS	W14	TDI
U7	EVSS	V11	MISOA	W15	DSP_DE*

Table 76-1. Neptune LTE 280 Production MBGA Ball Pad to Signal Name Net List (Continued)

Pin	Pin Name	Pin	Pin Name	Pin	Pin Name
W16	TCK	**W18**	CS0*	**W19**	CS0*
W17	AVSS(2)				

76.3 280 MAPBGA 13X13mm 0.65mm pitch for Development.

Reference Figure 76-1 package drawings and dimensions for the 280 MBGA Development package.
Reference Table 76-2 for ball pad to signal name.

76.3.1 280 MAPBGA Development Package Pinout

Table 76-2. Neptune LTE 280 Development MBGA Ball Pad to Signal Name Net List

Pin	Pin Name	Pin	Pin Name	Pin	Pin Name
A1	VSS_CP	**B5**	TRKG_OSC_OUT	**C9**	VAG_RX
A2	VSS_CP	**B6**	VCO1_P	**C10**	VSS_TXRX
A3	VSSA	**B7**	REG_BYP_TXRX	**C11**	VDD(3)
A4	XTAL	**B8**	RX_IX	**C12**	ATST_P
A5	REG_BYP_PRE	**B9**	RX_QX	**C13**	INT4
A6	VCO1_N	**B10**	PA_DET	**C14**	INT0
A7	REF_HI	**B11**	VSS(3)	**C15**	USB_VPOUT
A8	RX_I	**B12**	SRDA	**C16**	USB_VMIN
A9	RX_Q	**B13**	STDA	**C17**	CVDD(2)
A10	PA_OUT	**B14**	INT1	**C18**	URTS1
A11	BVDD	**B15**	PC14	**C19**	URXD1
A12	SC2A	**B16**	USB_XRXD	**D1**	VAG_CODEC
A13	SC1A	**B17**	USB_SUSPEND	**D2**	LVSS
A14	PE12	**B18**	D16	**D3**	ADC_P
A15	SJC_MOD	**B19**	D19	**D4**	TX_CP
A16	USB_VMOUT	**C1**	VAGOUT_CODEC	**D5**	RX_CP
A17	USB_VPIN	**C2**	VSS_CODEC	**D6**	REF_CODEC_N
A18	CVSS(2)	**C3**	DAC_N	**D7**	VDD_CODEC
A19	CVSS(2)	**C4**	VDD_CP	**D8**	TX_MOD
B1	VSS_CP	**C5**	VDDA	**D9**	VDD_TXRX
B2	VSS_CP	**C6**	REF_CODEC_P	**D10**	REF_LO
B3	CP_BIAS	**C7**	VSS_PRE	**D11**	ATST_N
B4	EXTAL	**C8**	VSS_TXRX(3)	**D12**	PA_REF

Table 76-2. Neptune LTE 280 Development MBGA Ball Pad to Signal Name Net List (Continued)

Pin	Pin Name	Pin	Pin Name	Pin	Pin Name
D13	SCKA	G1	ROW5	H11	D20
D14	PC12	G2	ROW7	H12	D22
D15	USB_TXENB	G3	COLUMN0	H13	REG_BYP_CODEC
D16	CTS2	G4	ROW2	H16	AVSS(6)
D17	UCTS1	G7	JVSS	H17	AVDD(5)
D18	SPI_CS6	G8	STANDBY	H18	DB_SZ0
D19	INT5	G9	ADC_N	H19	A9
E1	ADC_DATA	G10	PE11	J1	FVSS
E2	LVDD	G11	PE10	J2	VDD(4)
E3	CKIL	G12	SC0A	J3	TEST0
E4	DAC_P	G13	D21	J4	SIM_RST
E16	BVSS	G16	D23	J7	ROW4
E17	UTXD1	G17	EB1*	J8	D29
E18	AVDD(6)	G18	CS4*	J12	EB3*
E19	A15	G19	A10	J13	DB_SHOW_RW
F1	ROW1	H1	JVDD	J16	A8
F2	COLUMN1	H2	ROW6	J17	A19
F3	ROW0	H3	TEST1	J18	A16
F4	LOW_BATT*	H4	VSS(4)	J19	R/W*
F16	A14	H7	ROW3	K1	TEST2
F17	A13	H8	D31	K2	FVDD
F18	A12	H9	D17	K3	SIM_DIO
F19	A11	H10	D18	K4	CVSS(3)

Table 76-2. Neptune LTE 280 Development MBGA Ball Pad to Signal Name Net List (Continued)

Pin	Pin Name	Pin	Pin Name	Pin	Pin Name
K7	MOD	M4	LCD_CLK	N16	AVSS(3)
K8	D28	M7	D0	N17	RXD2
K12	DB_STRB	M8	D26	N18	ECB*
K13	QSTAT4	M9	D24	N19	QVSS
K16	EB0*	M10	QSTAT0	P1	SDATA
K17	CS3*	M11	QSTAT1	P2	LCD_RS
K18	A18	M12	QSTAT2	P3	CVDD(3)
K19	AVSS(5)	M13	CS5	P4	D11
L1	SIM_CLK	M16	A5	P16	CKO
L2	LCD_DATA[1]	M17	QVDD	P17	D25
L3	LCD_DATA[0]	M18	REG_BYP_CORE	P18	AVDD(3)
L4	LCD_DATA[4]	M19	AVSS(4)	P19	A7
L7	A20	N1	LCD_DATA[3]	R1	SIM_PD
L8	D27	N2	LCD_DATA[5]	R2	AVSS(7)
L12	QSTAT3	N3	LCD_CS	R3	A22
L13	DB_SZ1	N4	A21	R4	AVSS(1)
L16	LBA*	N7	D12	R16	AVDD(2)
L17	VDD(2)	N8	D15	R17	A6
L18	VSS(2)	N9	TOUT10	R18	A17
L19	AVDD(4)	N10	SPI_CS8	R19	A4
M1	SIM_VCCEN	N11	DVDD	T1	D3
M2	LCD_DATA[2]	N12	TRST	T2	D10
M3	A23	N13	TXD2	T3	D9

Table 76-2. Neptune LTE 280 Development MBGA Ball Pad to Signal Name Net List (Continued)

Pin	Pin Name	Pin	Pin Name	Pin	Pin Name
T4	D13	**U8**	RF_DATA	**V12**	MOSIA
T5	A24	**U9**	STDB	**V13**	RESET_IN*
T6	GSM*_DCS	**U10**	TOUT12	**V14**	EBW*
T7	TOUT9	**U11**	CTS1	**V15**	TMS
T8	MISOB	**U12**	SPI_CS1	**V16**	RTS2
T9	SCKB	**U13**	WDOG*	**V17**	CS1*
T10	INT6	**U14**	TDO	**V18**	D30
T11	QSCKA	**U15**	MCU_DE*	**V19**	EB2*
T12	SPI_CS0	**U16**	CVDD(1)	**W1**	AVDD(7)
T13	SPI_CS2	**U17**	A0	**W2**	AVDD(7)
T14	CVSS(1)	**U18**	A1	**W3**	D5
T15	RTCK	**U19**	A2	**W4**	D7
T16	OE*	**V1**	AVDD(7)	**W5**	RESET_OUT*
T17	A3	**V2**	AVDD(7)	**W6**	VDD(1)
T18	CS2*	**V3**	D4	**W7**	TOUT8
T19	BURSTCLK	**V4**	D14	**W8**	INT3
U1	D2	**V5**	AVDD(1)	**W9**	RF_CS
U2	D1	**V6**	RX_EN_OUT	**W10**	SC2B
U3	D8	**V7**	RF_CLK	**W11**	TOUT5
U4	D6	**V8**	TOUT11	**W12**	SPI_CS3
U5	VSS(1)	**V9**	EVDD	**W13**	ADC_SYNC
U6	TOUT7	**V10**	DVSS	**W14**	TDI
U7	EVSS	**V11**	MISOA	**W15**	DSP_DE*

Table 76-2. Neptune LTE 280 Development MBGA Ball Pad to Signal Name Net List (Continued)

Pin	Pin Name	Pin	Pin Name	Pin	Pin Name
W16	TCK	**W18**	CS0*	**W19**	CS0*
W17	AVSS(2)				

Appendix A
Neptune Bootloader (ABOOT)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	18 June 2002	Scott King	Updated dsp boot section
0.2	19 August 2002	Shannon Osgood	Updated per DDTS# DSPhl15158.
0.3	05/07/03		Updated for LTE specification release.

A.1 Introduction

The Neptune Bootloader consists of two small programs residing in the DSP and MCU program ROM which are executed when the DSP and MCU exit the reset state. The MCU and DSP core boot sequences are controlled by external pins at reset.

The main purpose of the MCU bootloader is to download a program via the UART peripheral to the internal MCU RAM.

The main purpose of the DSP bootloader is to provide for MCU-DSP communication so the MCU can download a DSP program through the MCU-DSP Interface (MDI) to the DSP program RAM.

The MCU and DSP bootloaders also contain code for SPS test modes including burnin.

The Neptune bootloader is based on the Patriot bootloader.

A.2 MCU Boot Sequence

A.2.1 Pin Configuration at Reset

The MCU boot sequence is selected by the MOD, PC12, PB1 (SCKA) and PB4 (SRDA) pins at reset as described in Table A-1, and Table A-2 below.

Table A-1. Boot Source Configuration

MOD	Boot Source
0	External FLASH (development package only)
1	Internal Boot ROM

Table A-2. Boot Mode Configuration - Internal Boot ROM

PC12	PB1 (SCKA)	PB4 (SRDA)	Boot Mode
1	X	X	PCS Boot Mode
0	0	X	SPS Burnin Test (Infinite 13N Test loop)
0	1	1	SPS Normal Boot Mode - 13N Power On Memory Test followed by UART Boot Sequence
0	1	0	SPS Rx/Tx external control mode

In addition, the PA13 (INT4) pin is used to configure the clocks for the SPS boot modes as shown in the table below:

Table A-3. SPS MCU Boot Modes: Clock Configuration

PA13 (INT4)	REFPLL	PAT_REF	MCU Clock	DSP Clock
0	not enabled	uncorrected 13MHz	uncorrected 13MHz	uncorrected 13MHz
1	208MHz	corrected 13MHz	52MHz	104MHz

A.2.2 MCU Boot Mode Descriptions

If the MOD[1] pin is driven low before the Neptune is reset, then the MCU will use the first word from the external FLASH attached to EIM chip select 0 ($\overline{CS0}$) as its reset vector. If this pin is driven high before reset, then the internal MCU boot ROM will be used instead. Refer to Chapter 75, “Neptune LTE Electrical Requirements (ELEC),” for exact timing requirements for this hardware configuration pin.

When booting from the internal ROM, the boot mode is further defined according to PC12, PB1 (SCKA), and PB4 (SRDA) pins. Based on the state of these pins as shown in Table A-2, the internal boot ROM will either execute the PCS ROM code; execute the SPS Normal boot mode; execute the SPS Burnin test, or start the SPS Rx/Tx external control mode.

For the SPS Normal Boot and Burnin Test modes, the bootloader will use PA7 (TOUT10) as the external indication of memory test pass or failure. PA7 (TOUT10) will be set high if the single 13N test on the internal MCU RAM passes when configured in SPS Normal Boot mode, and PA7 (TOUT10) will toggle if the mcu and dsp mcu boot code tests pass when configured in SPS Burnin mode.

1. The MOD pin has an internal pull-up resistor, so if it is not pulled low externally, the MCU will boot from internal ROM.

A.2.3 PCS MCU Boot Mode

If the SPS Boot mode is not selected, the PCS ROM code will not branch to the SPS boot code.

A.2.4 SPS MCU Boot Modes

Common to all SPS boot modes is the selection of the clock configuration. In the Patriot bootloader, the DSP clock was configured by the DSP bootloader, but since this function is handled by the MCU in Neptune, the MCU bootloader must perform this task. The clocks are chosen according to the configuration of the PA13 (INT4) pin as shown above in Table A-3. Note that the case where PA13 (INT4) is high is the reset state of the pin (no external configuration is required). The case where PA13 (INT4) is high is intended to be used for burnin, but can be used for other boot modes as well.

A.2.4.1 SPS MCU Burnin Test Mode

The 13N memory test is performed indefinitely in the SPS Burnin Test mode.

The 13N memory test will be performed over the entire MCU RAM, including the multiplexed RAM. The DSP and MCU memory's "selftime write" will be disabled for the burnin memory tests. The burnin code will also read the MDI general purpose flag 0, which is an indication of the DSP burnin code's pass/fail status (see Section A.3.2.1, "Burnin Mode.")

PA7 (TOUT10) will toggle as long as the MCU and DSP tests both pass.

Note that MCU and DSP burnin modes are both selected by the PB1 (SCKA) pin. This is different than the Patriot bootloader wherein the MCU and DSP burnin modes were selected by independent pins.

A.2.4.2 SPS MCU Normal Boot Mode

The SPS MCU Normal Boot mode allows code to be downloaded from a host to the Neptune MCU RAM, and also to external ram in CS1.

If this mode is selected, the first event which will happen is that the bootloader will establish communications with an external host computer assuming 13MHz PAT_REF and 19200 baud rate.

To be compatible with the Patriot bootloader and the Patriot Test Executive (PTE) host program, the bootloader will expect to receive a pilot or homing pattern that the host will generate: "aA" (0x61, 0x41). Once the bootloader detects this pattern, it will respond with an acknowledgement or service request of the host: 0x5A. The host will then proceed with the boot loading procedure defined below.

A.2.4.2.1 Preamble Search

In normal boot mode, once the 13N memory test is complete and the application has established communications on UART, the first set of four bytes transmitted from the host to the MCU is a preamble that will ensure that the host and the bootloader are in sync. On power up, the bootloader will wait for this preamble sequence and will not accept any further information until the verification of the preamble. The preamble sequence is as follows: 0x55 0x55 0x55 0xAA

Once the preamble is received, the bootloader will proceed. At the end of the download sequence, a checksum is performed on the received data (refer to Section A.2.4.2.5). If the checksum fails, the bootloader will return to the preamble search and attempt to reestablish communications with the host.

A.2.4.2.2 Destination Address

The second set of four bytes loaded from the host contain the destination address where the downloaded program that follows will be placed. For example, if the destination address is the beginning of the internal RAM, the following four bytes should be transmitted by the host: 0x01 0xf9 0x30 0x00. The valid range for internal RAM is defined in Chapter 6, “Chip Configuration and Memory Maps.” It is the responsibility of the external driver to avoid overrun of the allotted space.

A.2.4.2.3 Download Size

The next set of two bytes loaded from the host contain the total number of data bytes that will be transferred from the host computer to the internal/external RAM. This number is represented as a hexidecimal number with count starting with 1. For example, if 16 bytes of data are to be downloaded, the following two bytes should be transmitted by the host: 0x00 0x10. Once the total number of bytes specified in the download size have been loaded, and the checksum test passes, the bootloader will execute a jump and begin executing the downloaded code at the destination address specified in Section A.2.4.2.2.

A.2.4.2.4 Baud Rate Update

The UART transaction defaults to a baud rate of 19200, but changes to the rate may be embedded in the last two bytes of the header data loaded from the host to the MCU. These two bytes are loaded into the UART BRM MOD Register (UBMR), which determines the baud rate according to the following formula:

$$\text{UBMR value} = ((\text{CKIH}) / (16 \times \text{Baud Rate})) - 1$$

or

$$\text{Baud Rate} = (\text{CKIH}) / (16 \times (\text{UBMR value} + 1))$$

The default UBMR value on power up will be set to 41 (0x0029) for a 13MHz PAT_REF which represents a baud rate of 19200. The external driver must always send these two bytes even if the baud rate does not need to be changed. If the bootloader reads these bytes to be both zeros then there will be no change in the baud rate and the default value would apply. When changing the baud rate from the default, the host PC must provide sufficient time for the MCU to switch the baud rate.

A.2.4.2.5 Checksum

In order to provide some level of security and integrity of the data transferred, the total number of bytes transferred in the destination address, download size, baud rate update, and downloaded data would be one byte checksummed as follows:

total = sum of all bytes transferred except the preamble and the checksum byte.

checksum = 0xFF - (total MOD 256)

If the checksum fails, the bootloader will repeat the download procedure, beginning with the preamble search.

If the checksum passes, the bootloader will exit by branching to the first address of the downloaded program (the destination address).

NOTE:

The first instruction in all downloaded code must be in ARM mode (32bit opcodes). To save ROM space, the bootloader itself may be written in THUMB mode (16bit opcodes). If this is the case, the bootloader will switch to ARM mode before branching to the downloaded code.

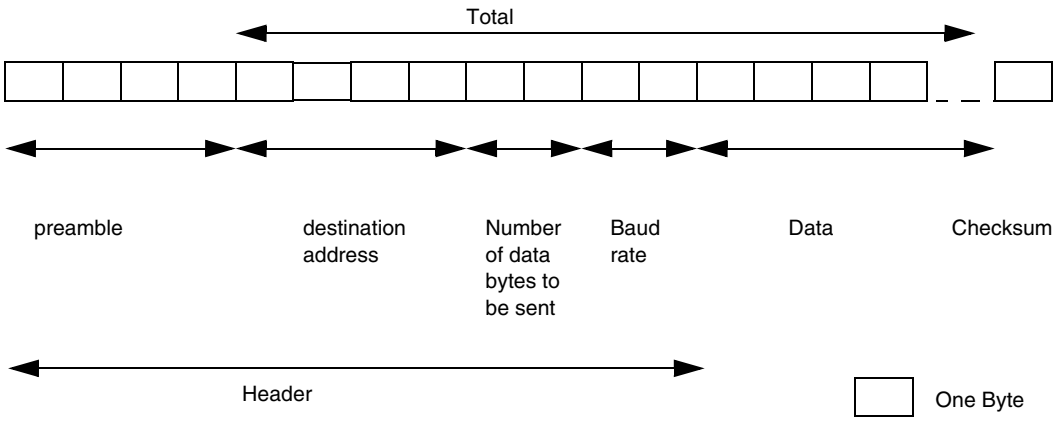


Figure A-1. Diagram of the UART Transmission

A.2.4.3 SPS MCU Rx/Tx External Control Boot Mode

This boot mode is proved for development and test only. It allows external control and data access to the receive and transmit modules on Neptune, defined as follows:

- A2DIGL SPI inputs come from external pins instead of the MQSPI
- TOUT signals (RX_ACQ, DMCS) are driven by external pins instead of the L1T
- RxCPROC and TX SSI signals are routed to external pins instead of the BBP
- ANATEST is configured to accept inputs from A2DIGL
- Signals from the mixed-signal modules are configured through the GPIO

This allows mixed-signal test equipment to directly interface with the mixed-signal modules without having to program the MCU and DSP cores.

If selected, the Rx/Tx external control boot mode will perform the following configurations (see text and figures describing the “control” configuration in Section 6.4, “Mixed-Signal Modules Interconnections,” on page 6-64):

- program the GPIO module to route the correct pins to the A2DIGL secondary SPI inputs
- program the GPIO module to route the correct pins to the A2DIGL secondary TOUT inputs
- program the GPIO module to route the RxCPROC and TX SSI signals to external pins
- program the A2DIGL to use it’s secondary SPI and TOUT inputs.

Following this, the code will execute a branch-to-self instruction forever.

A.3 DSP Boot Sequence

The DSP56600 boots directly from address P:$8000. The DSP bootloader includes one “main” bootloader protocol called Protocol A, and boot-modes for Motorola test purposes.

A.3.1 Pin Configuration at Reset

The DSP bootloader configures the SAP pins as general purpose input pins, and reads STDA to determine the boot mode, as described in Table A-4. The Protocol A boot mode define a messaging protocol that the MCU and DSP will use to communicate via the MDI.

If the SPS Test Mode is chosen as the boot mode configuration, the bootloader samples the SAP port pins SCKA, SC2A, SC1A, and SC0A to determine which one of the SPS test modes to enter, as described in Table A-5.

Note that for all these modes, the DSP clock will be configured by the MCU SPS boot code (see Section A.2.1, “Pin Configuration at Reset.”)

Table A-4. DSP Boot Mode Pin Configuration

STDA	Boot Mode
0	SPS Test Modes
1	Protocol A -- Normal Mode, Shared memory AND Messaging UNIT

Table A-5. SPS Test Mode Pin Configuration

SCKA	SC2A	SC1A	SC0A	SPS Mode
0	X	X	X	Burnin
1	0	0	0	Ext_word
1	0	0	1	Protocol B boot
1	0	1	0	Protocol C boot
1	0	1	1	Wait
1	1	0	0	Serial
1	1	0	1	Typ2
1	1	1	0	Stop
1	1	1	1	Ext_byte

A.3.2 SPS Test Mode Descriptions

Protocol A is described in Section A.3.3. The SPS test modes listed in Table A-5 are described below.

A.3.2.1 Burnin Mode

The burnin test mode executes an infinite loop of a 13N test sequence followed by a typ2 test sequence. After each iteration, a successful test will clear the MDI general purpose flag 0. If the test fails, MDI general purpose flag 0 will be set. The 13N test sequence tests all DSP RAM in the X, Y, and P spaces by writing known data patterns to the memory, reading back what was written, and verifying that the data is the same. The typ2 test sequence is used to exercise the DSP core by executing a series of multiply and accumulate instructions, and testing the output result to make sure it is correct.

Entry to the Burnin mode is also available via“Protocol A” boot mode. This entry method allows the dsp clock to be configured prior to burnin mode. See Section A.3.3.2.10, “burnin.request,” for information on entry to burnin mode through Protocol A.

A.3.2.2 Ext_word Mode

The ext_word test mode allows the DSP to read 24 bit data from the dsp master mode external program bus and write the data to a location in PRAM. The ext_word test mode should only be used if the device is in DSP master mode. Since the device may not be configured for DSP master mode immediately upon exiting reset, the boot mode will look for a sync pattern first.

The external data should be arranged in the following sequence:

1. Preamble of desired number of $5a bytes followed by one non-$5a byte
2. BCR value (16bit)
3. Load address (16-bit)
4. Number of words to load (16-bit)
5. Start address (16-bit)
6. Sequence of data to store in program RAM starting at the load address (24-bit each)
7. Repeat previous step until all data has been loaded.
8. Jump to start address

A.3.2.3 Typ2 Mode

The typ2 test mode executes an infinite loop of typ2 test sequences. The typ2 test sequence is used to measure typical dynamic power dissipation by executing a series of multiply and accumulate instructions. After each iteration of the instructions, the data in accumulator B is moved into x:$ff.

A.3.2.4 Ext_byte Mode

The ext_byte test mode allows the DSP to read 24 bit data from the dsp master mode external program bus and write the data to a location in PRAM. The ext_byte test mode is identical to ext_word, except that the DSP reads the data 8 bits at a time. The ext_byte test mode should only be used if the device is in dsp master mode. Since the device may not be configured for DSP master mode immediately upon exiting reset, the boot mode will look for a sync pattern first.

The data should be arranged in the following sequence in little-endian convention:

1. Preamble of desired number of $5a bytes followed by one non-$5a byte
2. BCR value (16bit)
3. Load address (2 bytes)
4. Number of words to load (2 bytes)
5. Start address (16-bit)
6. Sequence of data to store in program RAM starting at the load address (3 bytes each)
7. Repeat previous step until all data has been loaded.
8. Jump to Start address

A.3.2.5 Wait Mode

The wait test mode executes a WAIT instruction.

A.3.2.6 Serial Mode

The serial test mode will input a program sequence through the SAP serial port. The SAP port is configured for an external clock and frame sync, 8-bit words, word-length frame-sync, positive polarity on frame-sync, data latched in on falling edge of clock, and the MSB is transmitted first. The serial test data sequence is identical to ext_byte, except that the sequence begins with a preamble of as many $5a bytes as desired, followed by a non-$5a byte. Data should be arranged in the following sequence in little-endian convention:

1. Preamble of desired number of $5a bytes followed by one non-$5a byte
2. Load address(2 bytes)
3. Number of words to load (2 bytes)
4. Start address (2 bytes)
5. Sequence of data to store in program RAM starting at the load address (3 bytes each)
6. Repeat previous step until all data has been loaded.
7. Jump to Start address

A.3.2.7 Stop Mode

The stop test mode executes a STOP instruction.

A.3.2.8 DSP Bootloader Protocol B and C Specifications

Protocols B and C are backup protocols built into the bootloader to handle the scenarios where either the shared memory of the MDI or the message unit of the MDI are not functional. If both units are functional then the bootloader should use the normal bootloader Protocol A.

The shared memory (Protocol B) and messaging unit (Protocol C) boot modes are more primitive than the normal boot mode. The shared memory boot uses only the shared memory of the MDI, and the messaging unit uses only the messaging unit registers of the MDI. These protocols are described below.

It should be noted that if the messaging unit is not operational, then it may not be possible for the MCU to hold the DSP in reset.

A.3.2.8.1 Protocol B: Shared Memory Boot

Since the MDI memory values are undefined at reset, this boot mode requires a bit of MCU-DSP synchronization prior downloading the code. The first two 16bit words in the shared MDI memory space are reserved for synchronization messages. The following outlines the necessary steps the MCU needs to take to download DSP code in the boot mode:

- Download up to 3.5k-1 DSP program words to the MDI memory starting at the fifth MDI memory location. Note that the most signification portion is stored first (refer to Section A.3.3.3.1, “Accessing 24 bit Program Memory.”)
- Write the number of words downloaded to the 3rd MDI memory location
- Write synchronization word 1 (0x1234) to MDI shared memory location 0
- Wait for the DSP to acknowledge this by writing confirmation word 1 (0xabcd) to MDI shared memory location 1
- Write synchronization word 2 (0x5678) to MDI shared memory location 0

- Wait for the DSP to acknowledge this by writing confirmation word 2 (0xcdef) to MDI shared memory location 1
- The DSP should now be reading the program from the MDI memory locations and will jump to P:$0000 after the last word has been read.

A.3.2.8.2 Protocol C: Messaging Unit Boot

The messaging unit memory boot mode can be used if all that is required is to fill the lower 0.5k DSP program RAM and begin execution at DSP program address P:$0000.

Since this mode uses the MDI messaging unit registers which, there is no need for additional synchronization logic. In this mode, the MCU should first write the number of words which will be downloaded to MDI_R0. Following this, the MCU should write 511 DSP program words one at a time to the two messaging unit register (MDI_R0 and MDI_R1 with the most significant portion in MDI_R0). Since the DSP will read MDI_R0 first, the MCU should also write MDI_R0 first. Also, the MCU should make sure the DSP has read the register before writing the next value by polling the transmit empty bits in the MDI status register.

A.3.3 DSP Bootloader Protocol A Specification

The normal MDI boot mode (Protocol A) uses MDI communication between the DSP and MCU to implement the following functions:

- Download to the DSP program, X, or Y RAM
- Upload from the DSP program, X, or Y memories (RAM or ROM)
- Run diagnostic tests on the DSP RAM
- Start the DSP at a given program address (jump to a given address)

Upon entering the normal boot mode, the DSP will wait until a message has arrived from the MCU. Upon reception of the message, the DSP will perform the necessary actions and in most cases return an acknowledgment message to the MCU. The DSP will remain in the normal boot mode, waiting for MCU messages and executing them, until the MCU requests the DSP to exit the boot mode and start the user's application.

A.3.3.1 Message Formats

The normal boot mode uses both the MDI messaging unit registers and the XRAM shared memory for message transfers. In the cases where the message is fairly short, one or both messaging unit registers are used. If the message is longer (such as downloading a program to the DSP), MDI_R0 is used to point to the rest of the message in the XRAM shared memory.

NOTE:

Herein, the messaging unit registers (MTR0, MTR1, MRR0, and MRR1 for the MCU transmit and receive registers, respectively; DTR0, DTR1, DRR0, and DRR1 for the DSP transmit and receive registers, respectively) will be referred to as MDI_R0 and MDI_R1. That is, MDI_R0 may represent MTR0, MRR0, DTR0, or DRR0; and MDI_R1 may represent MTR1, MRR1, DTR1, or DRR1.

Table A-6 summarizes the messages which the bootloader supports. Initially, the bootloader is in an idle loop awaiting a message from the MCU. Upon reception of the message, the DSP will process and execute the command, and then send an acknowledgment message back to the MCU. The start_application.request message has no acknowledgment message. Also, if the DSP receives a message it does not recognize, it will return a special invalid opcode response.

Table A-6. Message Summary

Message from MCU to DSP	Message Opcode Number	Long or Short	Acknowledgment Message from DSP to MCU	Message Opcode Number	Long or Short
memory_write.request	1	long	memory_write.response	1	short
memory_read.request	2	long	memory_read.response	2	long
memory_check.request	3	long	memory_check.response	3	long
start_application.request	4	long	(none)	NA	NA
use_pll.request	5	short	use_pll.response	5	short
burnin.request	6	short	(none)	NA	short
(invalid message)	other	either	invalid_opcode.response	42	short

A.3.3.1.1 Short Message Format

The format of short messages is shown below in Figure A-2, and described in Table A-7. The most significant bit of MDI_R0 is used to indicate whether the message is a short message (S=1) or a long message (S=0). The P field is currently not used by the bootloader, but is provided for future implementations. The least significant bits of MDI_R0 hold the message opcode, which were shown in Table A-6. The middle bits (8-12) can contain message information if needed. If the short message uses the MDI_R1 register as well, the DW bit (bit 14) in MDI_R0 should be set to 1.

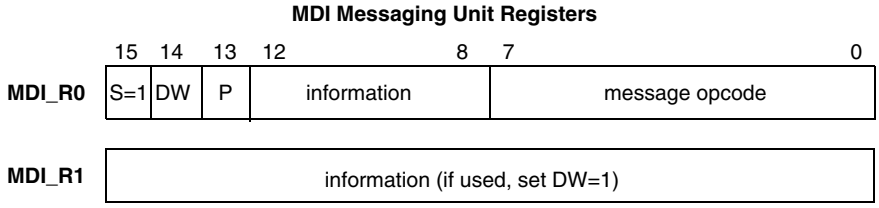


Figure A-2. Format of Short Messages

Table A-7. MDI_R0 Description (Short Messages)

Name	Description	Settings
S Bit 15	**Message Size --** Indicates whether the message is a short message or a long message.	1 = Short Message
DW Bit 14	**Double Word Message —** Indicates whether the short message uses only MDI_R0 or both MDI_R0 and MDI_R1.	0 = Single word short message. Uses MDI_R0 only. 1 = Double word short message. Uses MDI_R0 and MDI_R1.

Table A-7. MDI_R0 Description (Short Messages) (Continued)

Name	Description	Settings
P Bit 13	**Message Priority —** Indicates the priority of the message.	0 = Normal Priority Message 1 = High Priority Message
Optional Info Bit 12-8	**Optional Information —** These bits are available to be programmed by the user as desired.	N/A
Opcode Bits 7-0	**Message Opcode —** Indicates the function the bootloader should perform. Refer to Table A-6.	N/A

A.3.3.1.2 Long Message Format

The format of long messages is shown below in Figure A-3, and described in Table A-8. The long message is indicated by the S bit in MDI_R0 being set to 0. The UA and P fields are currently not used by the bootloader, but are provided for future implementations. The least significant bits of MDI_R0 indicate an offset address into the XRAM shared memory. The first entry in the XRAM shared memory at the indicated offset location is the message opcode. This is followed by as many information words as are necessary. Note that the offset is 13 bits, so it can point to an offset anywhere in the 8kx16 Dual Port XRAM + MDI shared memory space.

In the Figures blow, DSP_BASE refers to the beginning of the combined Dual Port XRAM and MDI shared memory at X:$0000. MCU_BASE refers to the beginning of the Dual Port XRAM at $2380_0000 when the offset is in the Dual Port XRAM region (offset = 0 to 3FF). When the offset is in the MDI shared memory register (offset = 400 to 1FFF), MCU_BASE is at address $2485_C000.

NOTE:

In the MCU Memory Map, the Dual Port XRAM (MDPI XRAM) is defined at addresses $2380_0000 - $2380_07FF, and the MDI shared memory is defined at addresses $2485_C800 - $2485_FFFF. When writing long message information to the Dual Port XRAM, the offset must be in the range 0 to 3FF. When writing long message information to the MDI shared memory, the offset must be in the range 400 to 1FFF.

If using the Dual Port XRAM to transfer data as part of a MCU to DSP long message, the MCU application must ensure that one additional write to the Dual Port XRAM occurs prior to sending the long message opcode in order to clear the late write buffer. Table A-6 specifies which MCU request opcodes are long messages. If the MCU fails to make this additional access to the Dual Port XRAM, then the data in the last word of the buffer stored in the Dual Port XRAM will be invalid. The first address in the Dual Port XRAM ($2380_0000/$0000) has been reserved as the late write buffer flush address in the bootloader code.

WARNING:

The additional write required to clear the Dual Port XRAM late write buffer is a new requirement for Neptune LTS/LTE/ULS. Legacy MCU code may not work properly if the long message buffer address is within the Dual Port XRAM area.

If the MCU specifies that the DSP should use the Dual Port XRAM to transfer data as part of a DSP to MCU long message, the bootloader code must ensure that one additional write to the Dual Port XRAM occurs prior to sending the long message opcode in order to clear the late write buffer. Table A-6 specifies

which DSP response opcodes are long messages. The DSP is required to make this additional access to the Dual Port XRAM so that the last word in the long message buffer will be valid when the MCU reads the data.

WARNING:

The bootloader code will use the first address in the Dual Port XRAM ($2380_0000/$0000) to flush the late write buffer. Therefore, this address should not be used in a buffer by any other opcode.

Refer to Chapter 8, “DSP Memory (DSPMEM),” for more information on the Dual Port XRAM late write buffer.

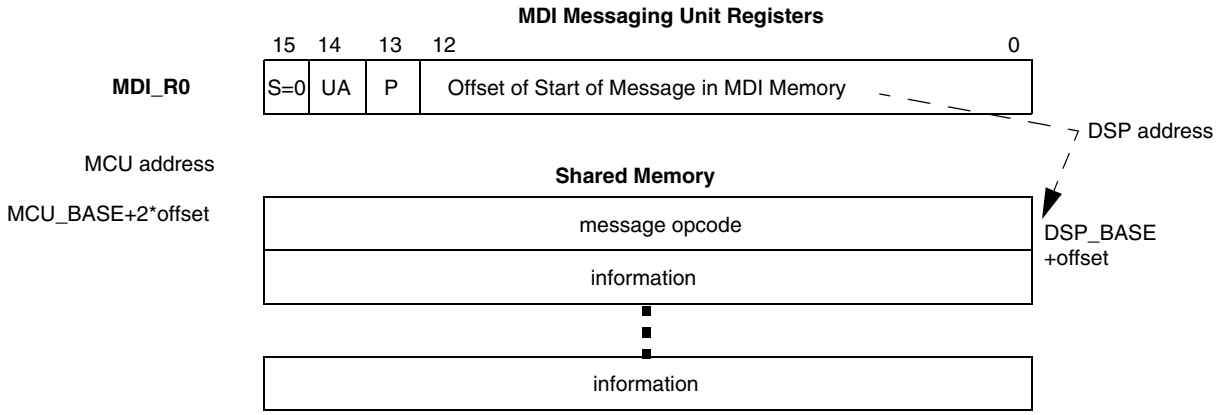


Figure A-3. Format of Long Messages

Table A-8. MDI_R0 Description (Long Messages)

Name	Description	Settings
S Bit 15	**Message Size --** Indicates whether the message is a short message or a long message.	0 = Long Message
UA Bit 14	**Queue Pointer —** Indicates whether to use the already established Rx queue pointer to retrieve this message, or to initialize the RX queue pointer using the address specified here.	0 = Use the already established Rx queue pointer to retrieve this message 1 = Initialize the RX queue pointer using the address specified here
P Bit 13	**Message Priority —** Indicates the priority of the message.	0 = Normal Priority Message 1 = High Priority Message
Offset Bits 12-0	**Offset of Start of Message in MDI Memory —** Indicates an offset address into the XRAM shared memory. The first entry in the XRAM shared memory at the indicated offset location is the message opcode as described in Table A-6. This is followed by as many information words as are necessary. Note that the offset is 13 bits long, so it can point to an offset anywhere in the 8kword XRAM shared memory space.	N/A

A.3.3.1.3 XYP Fields

The following fields is used in several of the messages to request a specific memory space.

Table A-9. XYP Fields

XYP (bits 15-13)	Memory Space
000	Program, Page 0
001	Program, Page 1
010	Program, Page 2
011	Program, Page 3
100	Program, Page 4
101	Reserved
110	X
111	Y

A.3.3.2 Message Structure

The following sections detail the structure of each of the messages.

A.3.3.2.1 memory_write.request

This is a long message from the MCU to the DSP for the purposes of writing to the DSP program or data RAM. The detailed structure of this long message is shown in Figure A-4. Following the memory_write.request opcode in MDI memory, the next entry is the number of DSP words to write. The next entry contains two fields: the XYP field in bits 15-13 determines which memory space will be accessed according to Table A-9 below; and bits 12-0 indicate the location in the MDI memory space of the data to be written to the DSP. The last entry (destination address) in the shared memory indicates the DSP address to which the data will be written.

In most cases, it makes sense for the source address offset to point to the word following the destination address. (In this case, the offset in MDI_R0 and the source_address_offset are related by source_address_offset = offset + 4.) However, the protocol allows for the data to be located anywhere in the XRAM shared memory space.

NOTE:

When writing to DSP Program memory, each 24 bit program word requires two 16 bit shared memory words to represent it. The number of DSP words to write should contain the number of 24 bit words, and the number of 16 bit words read from the source address by the bootloader will be two times the number of 24 bit words. Refer to Section A.3.3.3.1, “Accessing 24 bit Program Memory,” for more information.

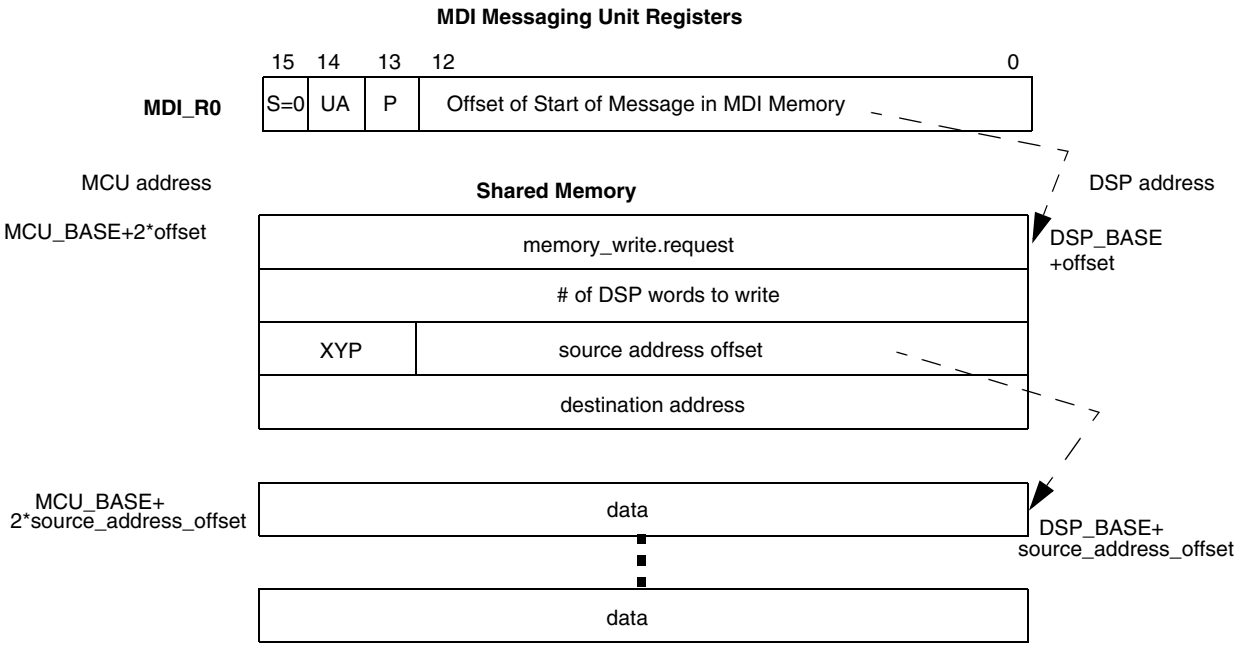


Figure A-4. Format of memory_write.request Message

NOTE:

It is the user's responsibility to ensure that the data space and address specified in the memory_write.request message are in the RAM space.

A.3.3.2.2 memory_write.response

This is a short message from the DSP to the MCU in response to a memory_write.request message. The format of this message is shown in Figure A-5. Note that the MDI_R1 register is not used. If the RET field is 0, the memory_write.request was a success; if the RET field is 1, the memory_write.request was a failure. Thus, since the memory_write.response opcode is "1", the MCU should expect the DSP to return MDI_R0=0x8001 if the memory write was successful.

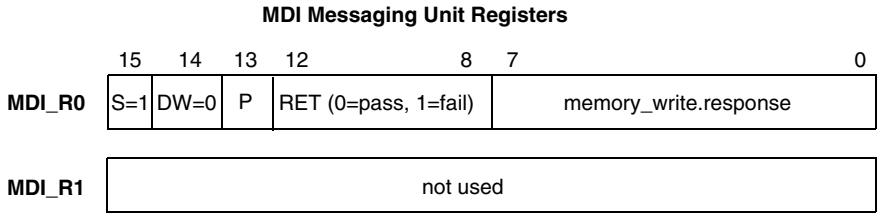


Figure A-5. Format of memory_write.response Message

A.3.3.2.3 memory_read.request

This is a long message from the MCU to the DSP requesting an upload of data from the program, X, or Y data spaces. The detailed structure of this long message is shown in Figure A-6. Following the memory_read.request opcode in MDI memory, the next entry is the number of DSP words to be read. The next entry contains two fields: the XYP field in bits 15-13 determine which memory space will be accessed according to Table A-9; and bits 12-0 indicate the location in the MDI memory space which the DSP will use in its long reply message. The last entry (source address) indicates the DSP program, X, or Y space address where the data will be read from. Any valid RAM or ROM address may be specified.

The choice of destination address offset is arbitrary, but care should be taken so that the DSP does not write over the original message which the MCU has written in the XRAM shared memory.

NOTE:

When reading from DSP Program memory, each 24 bit program word requires two 16 bit shared memory words to represent it. The number of DSP words to read should contain the number of 24 bit words, and the number of 16 bit words written to the destination address by the bootloader will be two times the number of 24 bit words. Refer to Section A.3.3.3.1, "Accessing 24 bit Program Memory," for more information.

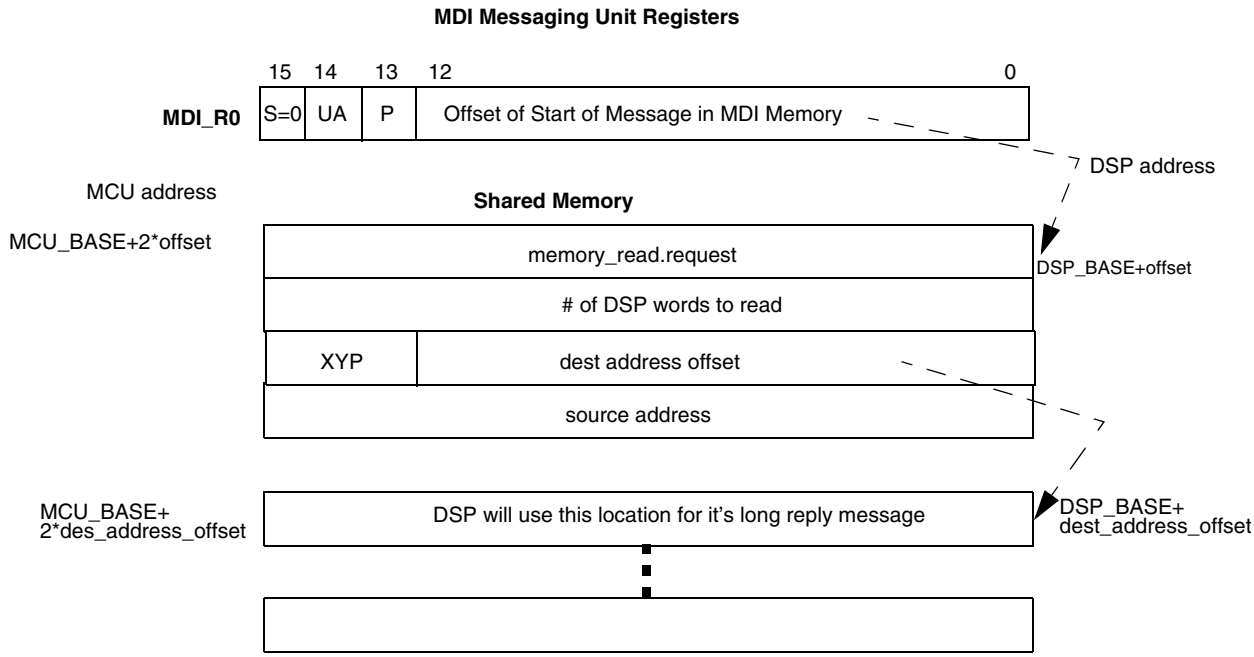


Figure A-6. Format of message_read.request Message

A.3.3.2.4 memory_read.response

This is a long message from the DSP to the MCU in response to a memory_read.request message. The detailed structure of this long message is shown in Figure A-7. Note that this long message will be located in the XRAM shared memory at the location defined by the memory_read.request message (destination address field).

Following the memory_read.response opcode in MDI memory, the next entry is the return code: a 0x0000 will indicate success, and a 0x0001 will indicate failure. The only reason for a failure would be if an invalid memory space was specified (XYP bits in the memory_read.request message equals a reserved value). If the return code indicates a failure, the remaining entries will not be written by the DSP.

Following the return code, the next number is the number of DSP words read. The next entry contains two fields: the XYP field in bits 15-13 indicate which memory space was accessed according toTable A-9; and bits 12-0 indicate the location in the MDI memory space of the data read from the DSP. The last entry (source address) indicates the DSP program, X, or Y space address where the data was read from.

In all cases, the bootloader will define the destination address offset to point to the word following the source address. Therefore, the destination address offset is related to the offset by dest_address_offset = offset + 5.

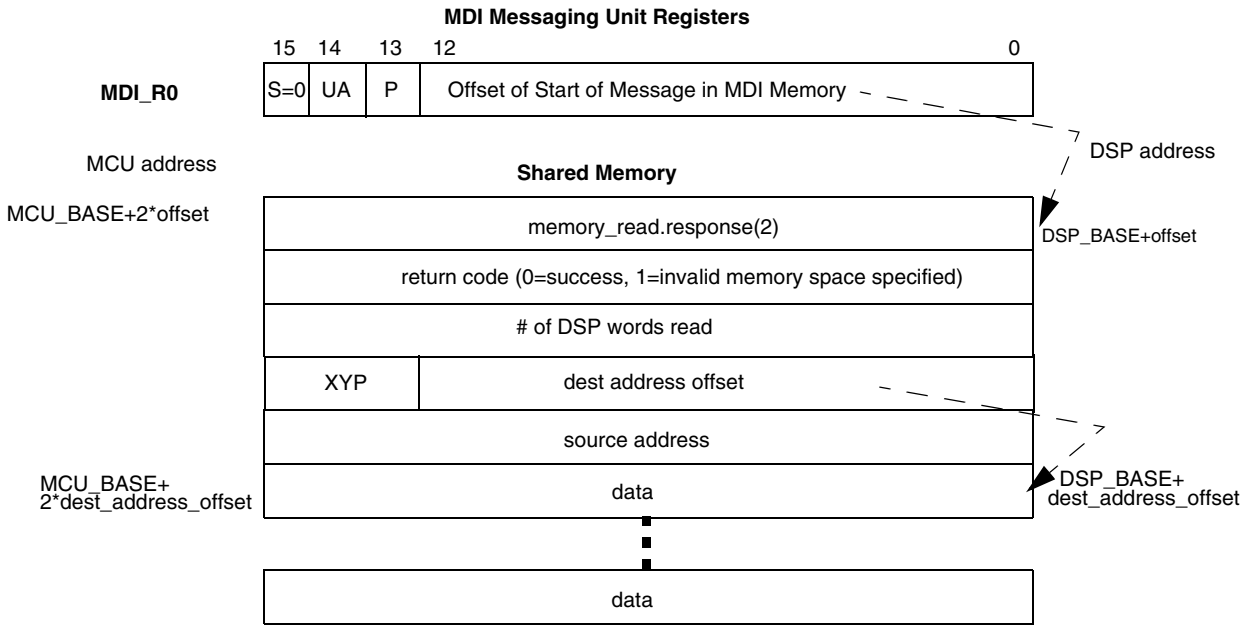


Figure A-7. Format of message_read.response Message

A.3.3.2.5 memory_check.request

This is a long message from the MCU to the DSP requesting a test of the DSP RAM memory. A memory_check.request to a memory space that only has ROM will result in an error reply.

The format of this long message is shown below in Figure A-8. Following the memory_check.request opcode entry in the shared memory, the next entry serves two purposes: the XYP field in bits 15-13 specify which memory space will be tested (see Table A-9); and bits 12- 0 specify the MDI address the DSP should use for its long reply message.

Normally, the return address offset will point to the next word (in this case return_address_offset = offset + 2). However, the protocol allows for the long reply message to be located anywhere in MDI memory.

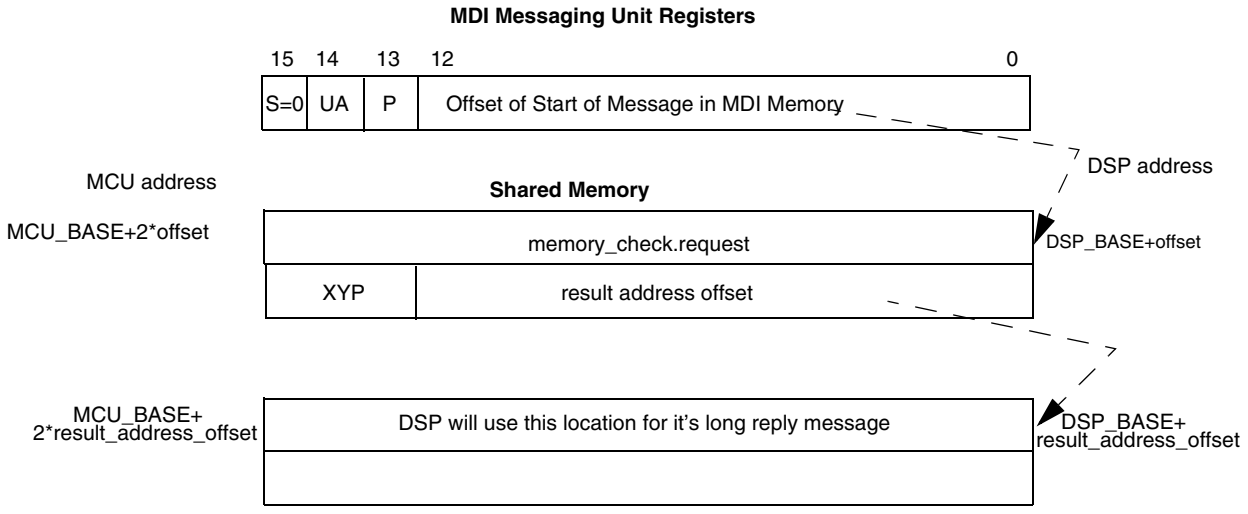


Figure A-8. Format of memory_check.request Message

A.3.3.2.6 memory_check.response

This is a long message from the DSP to the MCU in response to a memory_check.request message. The detailed structure of this long message is shown in Figure A-9. Note that this long message will be in the XRAM shared memory at the space defined by the memory_check.request message (result address offset field).

Following the memory_check.response opcode in MDI memory, the next entry is the return code; a 0 will indicate success, a 1 will indicate a memory failure, and a 2 will indicate that an invalid memory region was specified in the MEM field.

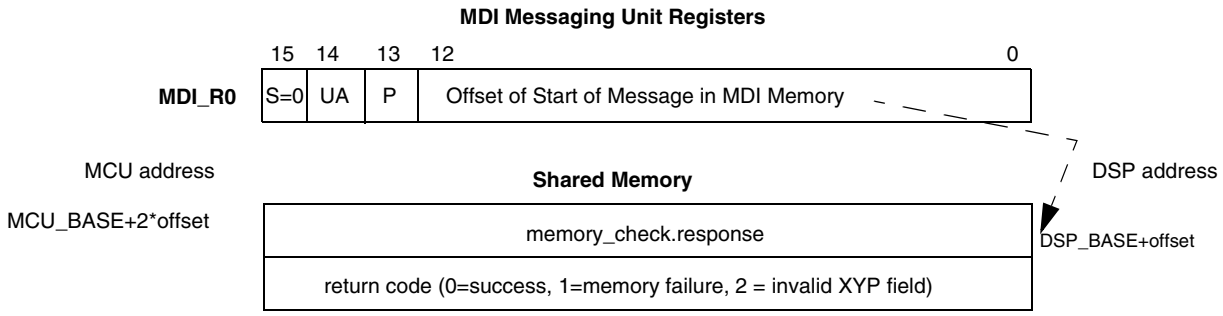


Figure A-9. Format of memory_check.response Message

A.3.3.2.7 start_application.request

This is a long message from the MCU to the DSP requesting the DSP to leave the boot mode and execute the user program. The detailed structure of this long message is shown in Figure A-10. Following the start_application.request opcode in XRAM shared memory, the next entry is the starting address in the program memory.

When the DSP receives this message, it will configure the paging logic for Page A, jump to the specified program address location and begin executing code at that location. No response message will be generated from the DSP.

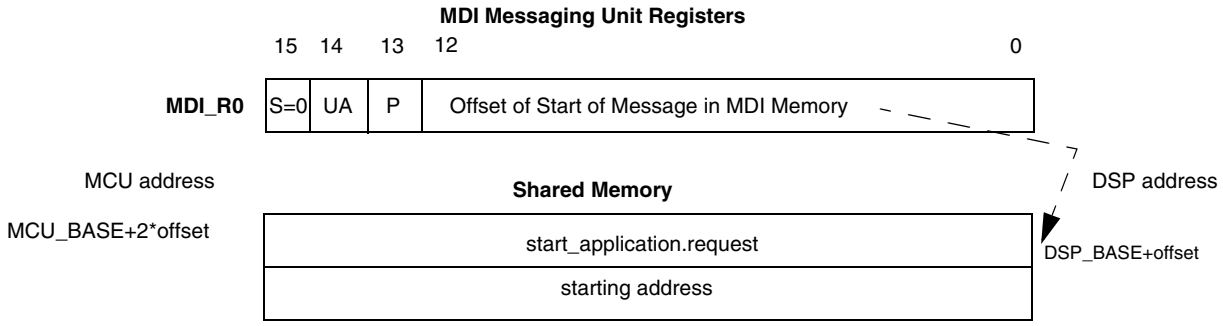


Figure A-10. Format of start_application.request message

A.3.3.2.8 use_pll.request

This is a short, double word message from the DSP to the MCU to request the DSP to switch it's clock from the pat_ref clock to the refpll clock. The divider factor bits to use are given in the 2nd word of the message.

When the DSP receives this message, it will configure the DCLKG to use the refpll output with the given divider bits, and reply with a use_pll.response message.

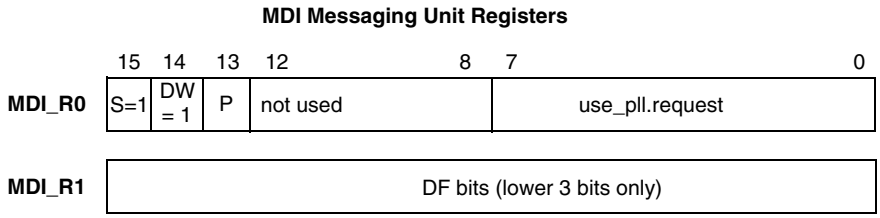


Figure A-11. Format of invalid_opcode.response message

A.3.3.2.9 use_pll.response

This is a short, double word message from the DSP to the MCU in response to a use_pll.request.

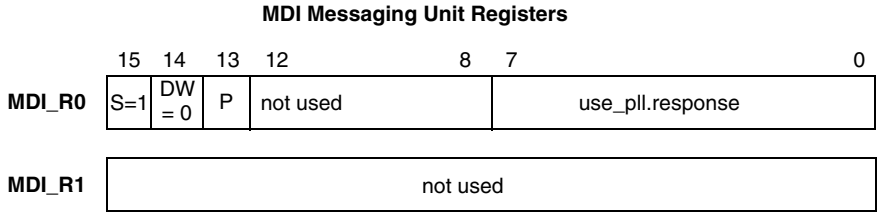


Figure A-12. Format of invalid_opcode.response message

A.3.3.2.10 burnin.request

This is a short, single word message from the DSP to the MCU to request the DSP to jump to the burnin test. No response message will be generated from the DSP. See Section A.3.2.1, “Burnin Mode,” for more information on the burnin mode.

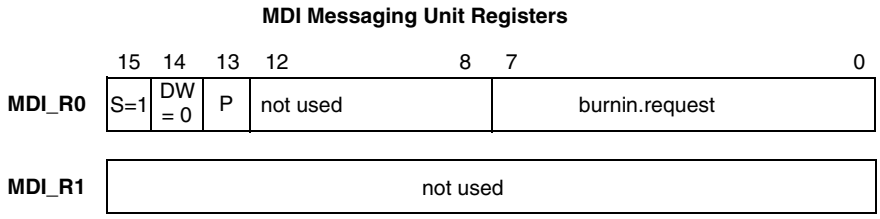


Figure A-13. Format of invalid_opcode.response message

A.3.3.2.11 invalid_opcode.response

This is a short, double word message from the DSP to the MCU in response to a unrecognized short or long message opcode. The format of this message is shown in Figure A-14. The RET field in MDI_R0 is used to indicate whether the DSP received a bad long or a bad short message. The offending opcode is returned in the MDI_R1 register.

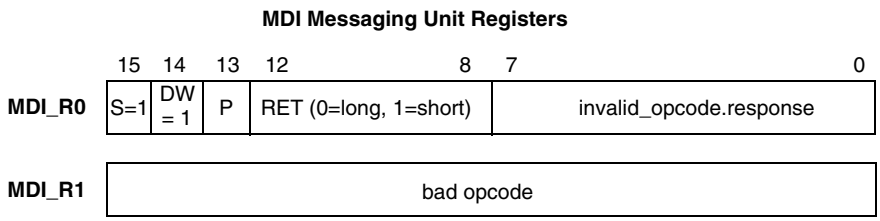


Figure A-14. Format of invalid_opcode.response message

A.3.3.3 Protocol A Usage Notes

There are certain procedures that should be followed when using the normal boot mode. These are described in the sections below.

A.3.3.3.1 Accessing 24 bit Program Memory

Downloads and uploads of DSP program memory require two words in the XRAM shared memory space for each program word, since DSP program words are 24 bits. For the memory_write.request message, the most significant portion (top 8 bits) should always be stored in the lower shared memory address, followed by the least significant (bottom 16 bits) in the higher shared memory address in order to be written correctly to the program memory by the bootloader. For the memory_read.request, the bootloader will store most significant portion in the lower shared memory address, followed by the least significant in the higher shared memory address. This is illustrated in Figure A-15.

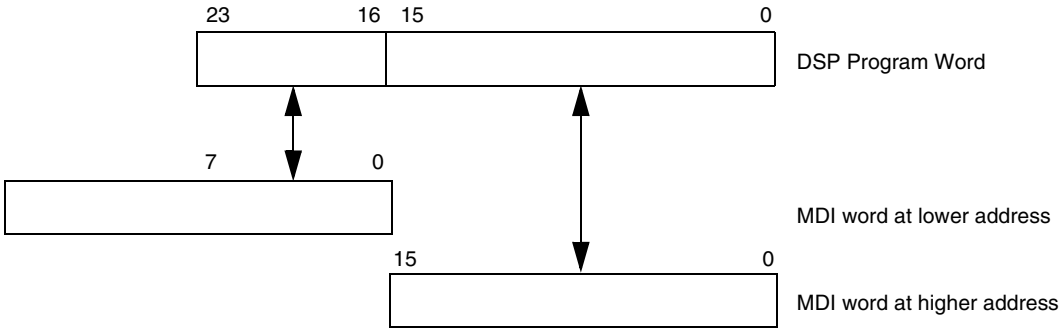


Figure A-15. Mapping of DSP Program Memory Words to MDI Message Words

A.3.3.3.2 Shared Memory Size

The size of the XRAM shared memory is 8k words. This fundamentally limits the upload and download size. Larger uploads and downloads will need to be split into several uploads and downloads. The DSP does not perform any error checking to make sure the addresses are within the XRAM memory space; this is left to the MCU program.

A.3.3.3.3 DSP Message Reception

The DSP bootloader program determines if there is a message by polling the MDI status register bit which indicates a new message has been received in the MDI_R0 register. Therefore, the MCU should write MDI_R0 as the last step to initiate the DSP reception of the message.

A.3.3.3.4 Long Reply Message Address

For long reply messages, where reply offset address is defined by the invoking long MCU message, care should be taken so that the DSP will not write over the original message which the MCU has written in the XRAM shared memory. The reason for this is that the DSP may need to access the original message while it is writing its response message.

A.4 Complete Pin Allocation Summary

Table A-10 lists the pins that are used to configure the Neptune bootloader at reset. Pull-up and/or pull-down resistors can be applied to these pins to cause the MCU and DSP to enter the desired mode.

Table A-10. Allocation of Neptune Pins to Various Boot Configurations

Pin	Purpose
MOD	ROM source for MCU boot
PC12	MCU PCS/SPS Boot Mode Select
PB1 (SCKA), PB4 (SRDA)	SPS MCU Mode Selects
PA13 (INT4)	SPS - MCU and DSP Clock configuration
PA7 (TOUT10)	SPS burnin/13N output signal
STDA	DSP Normal/SPS Boot Mode select
SCKA, SC2A, SC1A, SC0A	SPS DSP Mode Selects

Appendix B
Design for Testabilty Methodology (DFT)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	05/07/03		Updated for LTE specification release.

B.1 Introduction[1]

The purpose of this chapter is to describe the Design For Test (DFT) methodology and techniques used in the Neptune chip. A brief explanation of the specific design implementation is given in some areas to help clarify the DFT methodology. This chapter should be referenced by all Neptune derivatives. Each derivative should have a document describing any exception and DFT connections specific for that chip.

The main objectives of this chapter is to describe how to obtain observability, controllability, achieve a high fault grade coverage, and be Failure Analysis friendly in the Neptune and all derivatives in an automated repeatable manner. To accomplish these objectives, certain DFT techniques will be described to aid the designers in choosing the most efficient means of testing their circuit. The general approach of the test methodology will be outlined.

The general approach for the Neptune is to adopt a primarily *full-scan* design methodology for structural testing. If the test structures are architected correctly, and if designers adhere to certain design guidelines, then an *Automatic Test Pattern Generation* (ATPG) tool can be used to provide the manufacturing test vectors.

The goal of the chosen test strategy is to provide test vectors with at least >95% gate level single stuck-at -faults coverage. To accomplish this goal, a full-scan approach using an ATPG tool for all standard logic and a BIST (*Built In Self Test*) for memory structures will be suggested to provide manufacturing test vectors. This goal also includes providing these vectors in a timely manner with minimal manual intervention.

Neptune supports multiple types of test modes to produce a high quality test solution. The fault models used for scan based testing include stuck-at, transition, IDDQ, and path delay. A functional test suite must also be generated by design for characterization and failure analysis. These patterns may also be used to address faults which may not be adequately covered by structural tests.

1. This specification is a “living” document, which will be updated as necessary during Neptune evolution. It contains hidden text comments which should be turned off for any printing that is to be distributed. If this sentence and the previous one appear, then you have comments turned ON.

In addition to manufacturing test, other test environments were considered during the Test Methodology phase. Specific features were included in the test design to address problem test areas such as: board test, burn-in, wafer probe, failure analysis, and debug.

Each Designer is in charge of performing scan insertion and running the ATPG Design Rule Checker in his (her) own block. Any questions related to this topic can be directed to DFT team.

ATPG and Test pattern Gate Level verification and Test pattern Timing verification will be handled by the DFT team and supported by designers.

This preliminary version is subject to changes and additions at anytime. Any comments are welcome in order to help improve this chapter.

B.2 Chip Level Overview

Neptune is a Dual Core based chip dedicated for wireless communication. Neptune has a significant amount of memories, mixed-signal, and analog modules as well. Each of these areas will have a corresponding DFT methodology described in this document.

The two cores currently chosen are the S-Onyx as the DSP and ARM 7 as the MCU. Other cores which maybe substituted in the future will need to fit into this general DFT architecture. This chip configuration contains three kinds of peripherals, DSP peripherals only, MCU peripherals only and shared peripherals between DSP and MCU.

A common test architecture has been implemented for all the peripherals independently of their location. We necessarily won't have all the same peripherals for each Neptune derivative. So the test implementation must be a plug and play solution.

Neptune is leveraging "Patriot legacy" modules to maximize reuse and minimize risk and schedule impact. New modules and cores which are being generated use a newer methodology that affect the test and control of those sections during test modes. The new modules and cores will be more compliant to SRS standards. They will contain a different clocking methodology, possibly a different reset methodology, and a port naming convention. It is a requirement that no restrictions exist at the chip level in mixing old or new modules or cores in the same scan group. A scan group is one or more modules which are scan tested at the same time. It is a requirement that a common Test Control Module (TCM) be utilized for all designs both old and new.

More than one module and/or core maybe scan tested at the same time to improve test time. The maximum number of scan chains that can run in parallel is limited by the availability of pins for test and the power dissipation of the package.

An example Neptune configuration is shown in Figure B-1. All modules will be full-scan with the exception of the TCM, GPIO, and analog modules.

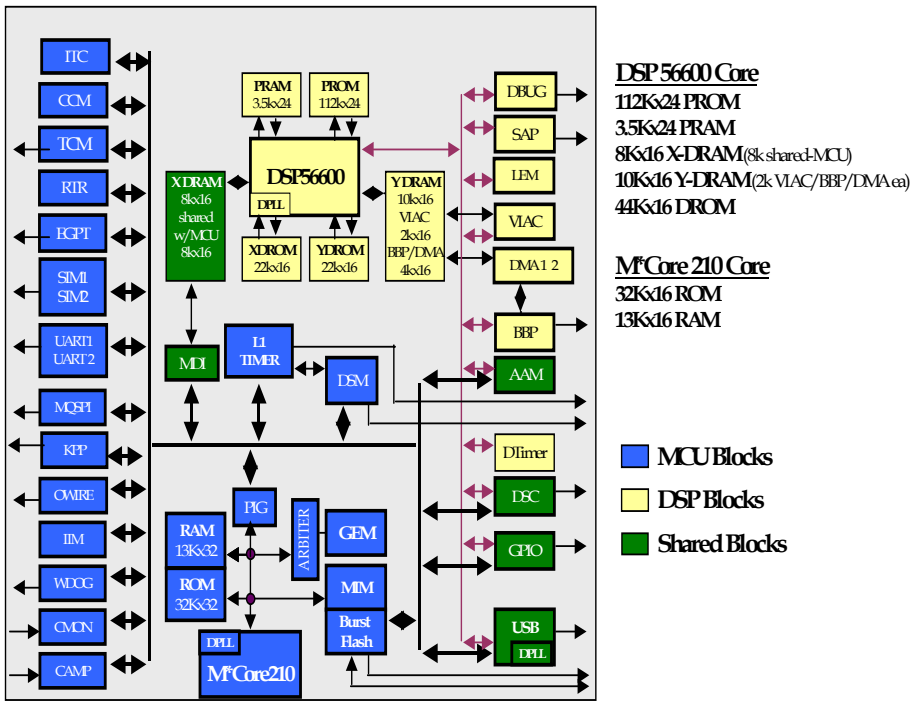


Figure B-1. Example Neptune Configuration

Entering scan mode operation is done by utilizing alternate bus master mode to configuring registers in the Test Control Module (TCM). The GPIO is also configured at this time to utilize pins which will be shared for scan test control, scan inputs and scan outputs. Refer to the section on “Entering Scan Mode” for more details.

B.2.1 Design Flow for Module Scan Insertion

Each module will be scan inserted as a stand-alone module. (This means that there is one or more scan chains per module depending upon the size of the module.) A standard scan insertion script is used to simplify the scan insertion process. This was done so that reuse, ease of integration, debug, and verification would be maximized. Each module is then verified in stand-alone mode to identify any problems which might effect test coverage. This flow applies for all modules. Cores will have a similar flow.

The module or core is then integrated into the chip level scan netlist. The modules or core is evaluated by the DFT team to insure no controlability or observability problem occurs. Automatic Test Pattern Generation (ATPG) takes place at the chip level. These patterns are re-simulated with Verilog to verify no issue or mismatch occurs.

The following diagram shows the major steps required by design to deliver a module to the DFT team. The Neptune scan insertion script checks many of the module design requirements described in the chapter. The following describes the major process steps.

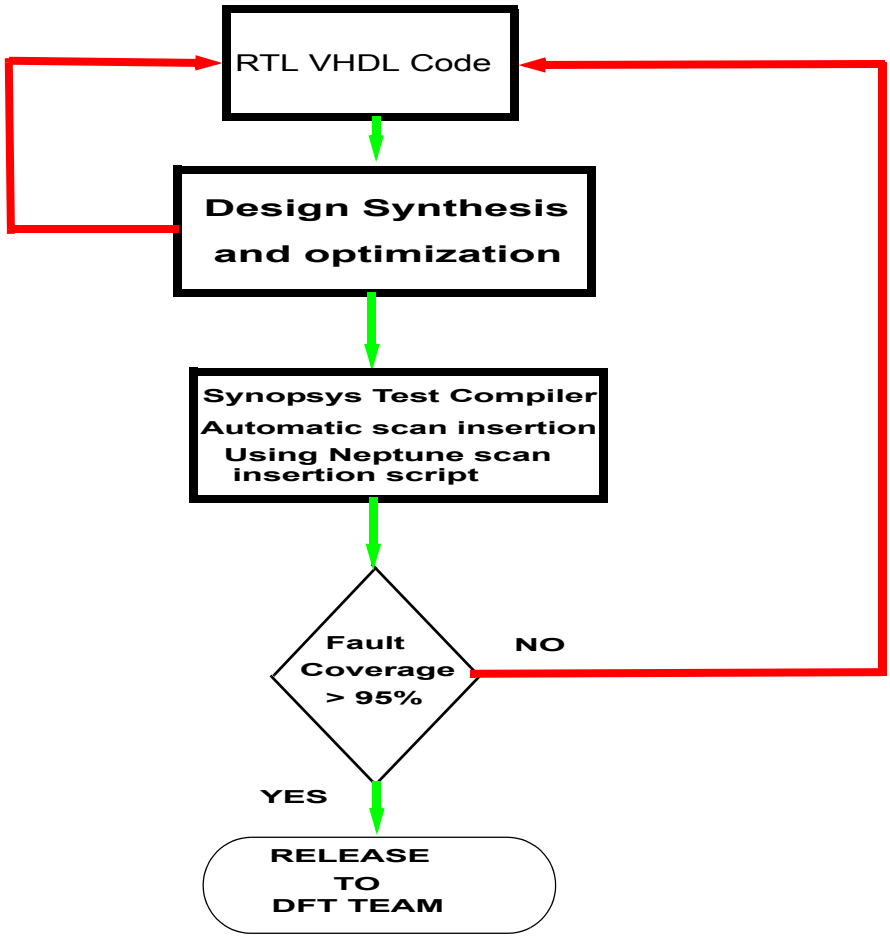


Figure B-2. Design Steps for Scan Insertion

It is the module designer's responsibility to successfully execute the scan insertion script. The DFT team can be used as a resource if questions or issues arise. The DFT team can give additional suggestions to design to improve test coverage where appropriate.

B.2.2 General ATPG Flow

The DFT team receives the Verilog netlist and the Synopsys DB file and verifies that the module meets all the DFT requirements. The synopsys DB file needs to have all of the contraints used in the synthesis process. The module is then integrated into a chip level module and ATPG is run to find any possible controlability or observability issues. The scan patterns are then run with the chip netlist using Verilog to identify any mismatches. This process is also repeated after the module has gone through Automatic Place & Route (ARP) with full SDF annotation.

As the design cycle continues, the module is included in a scan group and run with other modules in parallel. These vectors are also re-simulated in Verilog. The following are the major process steps in releasing patterns to test engineering.

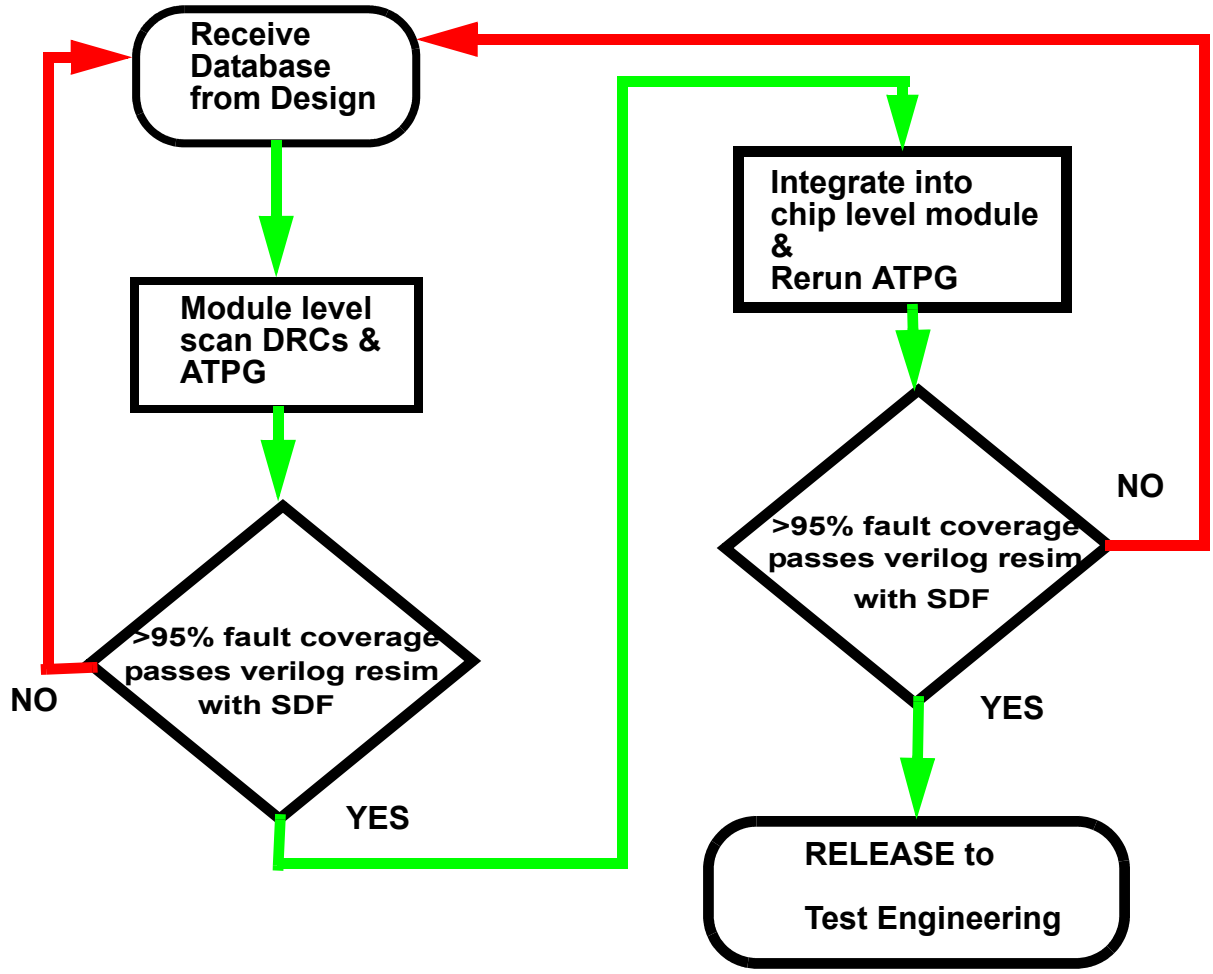


Figure B-3. Steps for DFT Team to Release to Test

B.2.3 Getting SDI and SDO Inputs In and Out of the Chip

Several modules are essential for the chip to enter into scan test mode. The Test Control Module (TCM), General Purpose I/O (GPIO), Clock Control Module (CCM), and WDOG must be operational to configure Neptune into scan mode.

After the chip has been successfully configured into scan test mode, the Test Control Module (TCM) drives two signals which are unique to each block (scan_mode & test_reset). A more detailed description of the logic connected to these signals can be found in the “Module Scan Requirements”. Since these signals are unique to each module and are programmable, isolation of a single module for debug and failure analysis can be achieved.

To reduce test time, modules can be run in parallel. The grouping of modules to be run together might be different for each Neptune derivative. Modules which functionally talk together might be put in the same scan group to improve test coverage when possible. The number of modules which can be grouped together is limited by the amount of pins available for sdis and sdos, the power dissipation of the package, and the tester memory depth.

The figure below depicts the common conductivity of getting sdi and sdo from the package pins to the module(s) under test. The GPIO is configured to simply connect the package pins to the scan inputs and scan outputs. Actual connections on peripherals sdos pass through the TCM to save routing as shown in Figure B-6. Other inputs, such as scan_en and test_clk, are connected the package pins in the same manner as sdi. Refer to the GPIO test mode configuration for more details.

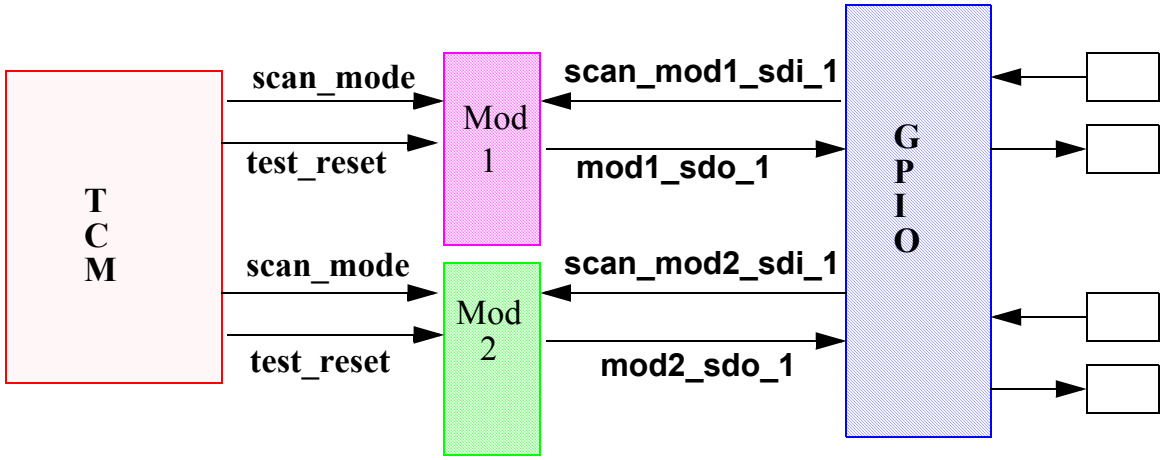


Figure B-4. General Configuration of sdi/sdo Connections

B.2.4 Legency Module Naming Convention for Test Signal Ports

Each module is required to have a standard scan interface. The ports which each scannable legacy module will have are described below. Refer to Figure B-5, “Module Scan Ports for Legacy Modules,” on page 7. Additional information of the use and connections of these signals can be found in “Module Scan Requirements”.

scan_mode: scan_mode signal coming from TCM. This signal is unique for each module. This signal is low in functional mode.

scan_en: scan enable signal coming from primary input of the chip. This signal is shared by the entire chip. This signal is low in functional mode.

test_reset: test reset signal coming from TCM. Only used in scan test mode and unique for each module. This signal is low in functional mode.

test_clk: scan test clock coming from TCM. Only used in scan test mode and is shared by the entire chip. This signal is not active in functional mode.

sdi_xx: for Scan Data Input. From primary inputs. xx depends on the number of scan chains. This naming convention is only being used for legacy modules. All new modules should use ***scan_<module_name> _sdi_xx.***

OR

scan_<module_name> _sdi_xx: Scan Data Input. xx depends on the number of scan chains. module_name should be the 3 or 4 letter acronym. This is the preferred naming convention for new modules.

<module_name> _sdo_xx: Scan Data Output. xx depends on the number of scan chains. module_name should be the 3 or 4 letter acronym. Additional information can be found in “Module Scan Requirements”.

scan_<module_name>_sdi_xx_bypass: If scan_mode is not selected, scan data from another scan chain in the same scan group can utilize the SDO port to the GPIO.

scan_<module_name>_div_in: scan data coming from another module. This is used to daisy chain all of the chains together into one large chain.

tcm_scan_divergence: scan divergence enable coming from TCM. Only used for failure analysis and debugging. This mode is described in detail in the module requirements section.

tcm_iddq: (**OPTIONAL**) test iddq enable coming from TCM. Only used to disable high current paths for IDDQ testing. Modules containing analog, tri-states, memories, pullups, pulldowns, pads, or other hi current paths use this signal to disable high current paths during iddq mode. This mode is described in more detail in the analog test section of this specification.

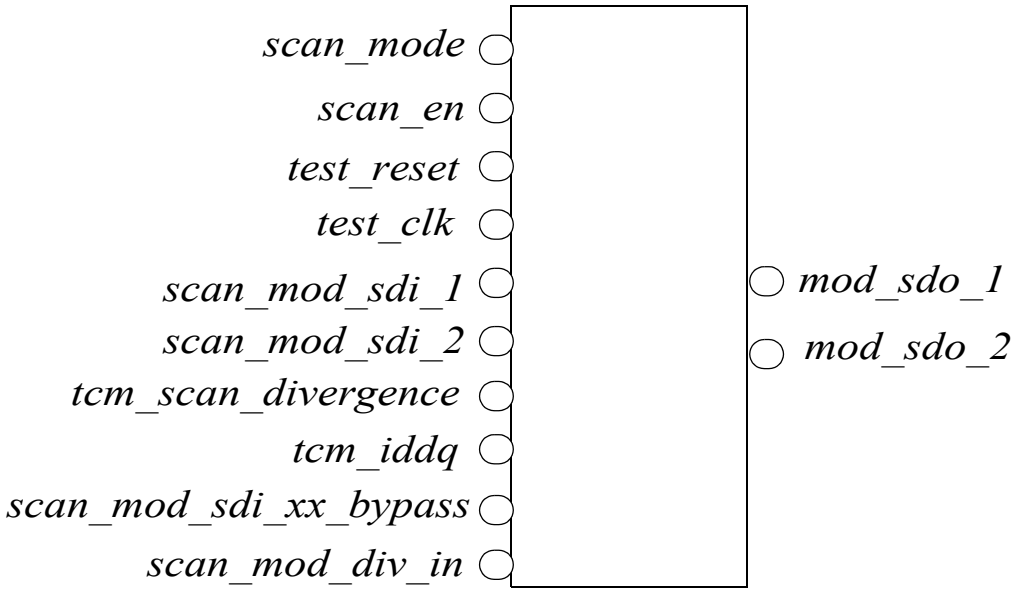


Figure B-5. Module Scan Ports for Legacy Modules

A detailed description of the logic connected to these signals can be found in the “Module Scan Requirements”.

B.2.5 Testing Cores in Neptune

At the Neptune chip level, the S-Onyxu and the ARM Platform must be handled carefully. Ideally these cores will be tested as any other module on the Neptune chip. These cores will support a mode to scan test the core, while not affecting signals to other modules (core test mode). They will also support a mode to test the ports of the core with other connected modules.

A detailed description of testing the S-ONYX and the ARM Platform is given in the core test section of this specification.

B.2.6 Bus Interface for Peripherals

Peripherals are not directly connected to the DSP bus or to the MCU bus. An interface is inserted between the bus and the peripheral itself. Additional information on these interfaces as it pertains to scan is available in the “Module Scan Requirements”.

DSP side and MCU side have their own kind of interface.

- DSP side:

  The DSP Bus Interface is called S-PMBIF (Synthesized Peripheral Module Bus Interface).

- MCU side:

  The MCU Bus Interface has two interfaces. MIG (Module Interface Gasket) and the JIG (Just another Interface Gasket).

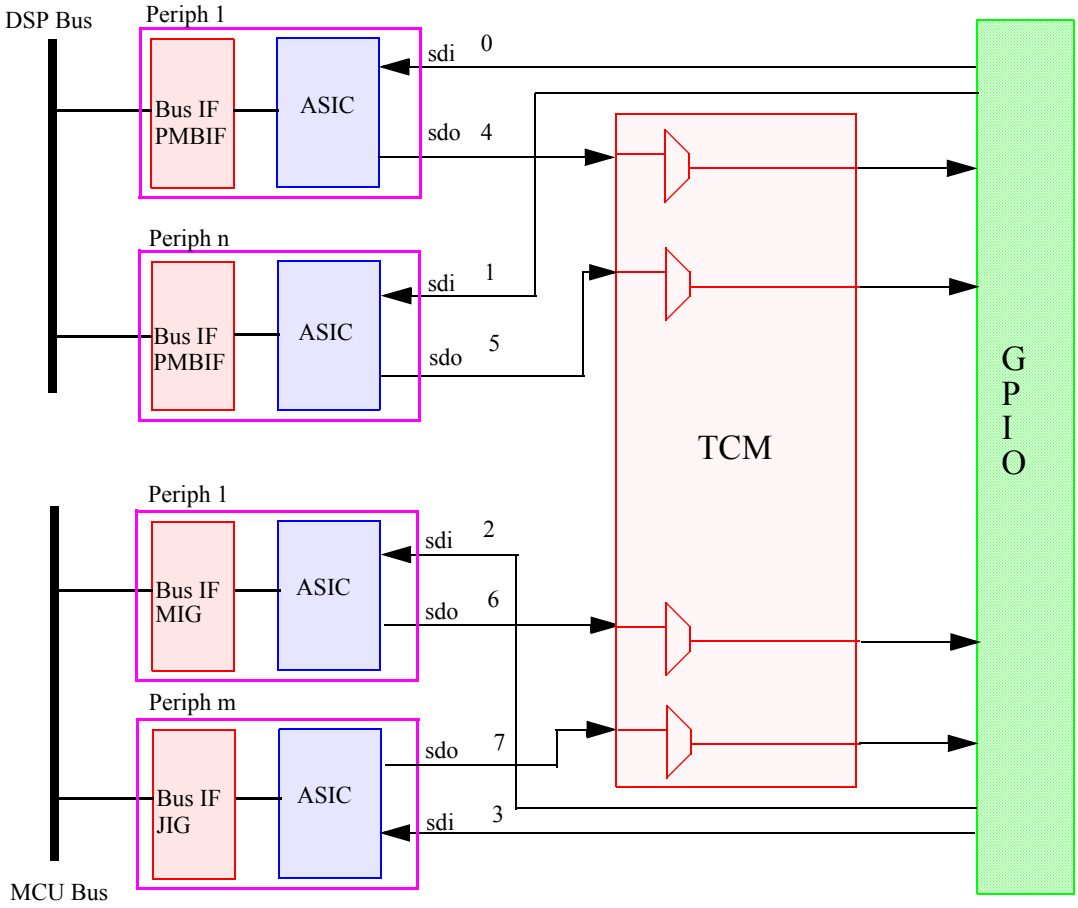


Figure B-6. Common Scan Test Bus for Peripherals

B.2.7 Maximum Scan Chain Length

Scan chain length directly impact the production test time. Since a design has a fixed number of scannable elements, two configurations are typically used: make few scan chains with a large amount of elements per chain or make many scan chains that are very short. The shorter the scan chain length, the more scan chains you have. Production test time is dictated by the length of the scan chains. The longer the scan length, the longer the test time.

Of course, the number of scan chains is limited by the number of scan_in and scan_out ports that are available. The production tester memory depth might also play a role in scan chain depth, but usually it is not the limiting factor.

From experience, the maximum length of a scan chain should be around **100 - 150 elements**. If at a later date this number needs to be increased because of a limitation in package pins or other design considerations, no module rework will be needed. Scan chains can simply be serially connected to make

longer chains at the chip level. This is done to support burn-in and scan divergence modes. All scan chains are connected together into one long scan chain for burn-in. This needs to be implemented for all future Neptune devices which require burn-in testing.

Scan chain should be similar length in a block or in the same group of blocks under test (if we apply parallel testing). This prevents wasteful tester memory usage. If a small and long scan chain are tested in parallel, tester memory will be used to fill in the missing cycles with dummy data to make the chains equal length.

B.2.8 Clock & Reset Control During Scan Testing

At the chip level, the clock and reset must be controllable via a primary package pin that the ATPG tool can control. In scan mode operation, pins will be used for test_clk and test_reset. These pins are mux'd using the same method as the sdi and sdo pins.

The addition of an ARM platform and new modules using an IPBUS makes the implementation of a common test_clk and test_rst more challenging. These modules will be IPBUS TAN-line compliant but will be driven by the TCM using the standard signals defined for legacy modules. The method to do this has not yet been fully defined.

The Clock Control Module (CCM), sections of the Test Control Module (TCM), the General Purpose IO (GPIO) and the analog sections will not use test_clk. Specific information for this exception can be found in the "Module Information" section.

B.2.9 Testability and Test Coverage

Multiple issues can dramatically affect a modules test coverage. The "DFT Guidelines" chapter are good guidelines that will help prevent most of these types of problems. The goal for Neptune is to have > 95% fault coverage and even higher test coverage.

Neptune has challenges with several issues such as bus contention, uncontrolled or unknown states, unoberservable signals, pin configuration, and testing across major busses. These topics will be discussed in this section.

B.2.9.1 Bus Contention

Having a stable bus interface stable during scan test is a must for Neptune. The Neptune test architecture depends upon the module following the proper usage of the mandatory module scan ports. The ATPG tool will not generate a test pattern with possible bus contention. Busses and global nets will be scan tested at the chip level. If bus contention is not prevented, significant test coverage loss will occur.

B.2.9.2 Unknown State Signals

Unknown states should not exist during scan test mode. All tri-state elements enables must be controlled as per the “Module Requirements” section of this specification.

If an unknown state signal is going directly to a scan element (SFF or SLatch), the scan_en input of this scan element should be connected to scan_mode (always at one during test mode) to avoid propagating or X state through the scan chain. All the scan elements connected this way in a block should be grouped together in the same scan_chain to improve ATPG effectiveness.

B.2.9.3 Unobservable Signals

To solve the problem of unobservable signals, we add some sequential logic (shadow logic) used for observability of these signals (this logic won’t be used in functional mode). A functional read/write simulation pattern to solve this problem will be sufficient but a fault simulation of this pattern by the design team is necessary to know coverage of the non observable faults by ATPG.

In addition, all the New Custom Module IO control ports (input/output enable, open drain enable, etc...) are not observable to external chip’s signals. In large New Custom Modules, with a lot of IO pins, additional General Purpose IO registers can solve this problem. A simple functional test will change the pin’s directions and drive data to the pins. This test will have to be provided by the designer.

Some DF techniques might be use for the second version of the Neptune chip.

B.2.9.4 Configuring the IO Pins During Test Mode

The input only and the output only pins are fully controllable and observable during Test Mode. However, bidirectional pins need to be configured to be inputs during Test Mode. This will ensure that the value, driven into the Customer Specific logic from the <pin>_i is always known and driven by the test pattern. This means that additional logic has to be added to the bidirectional control signals, coming from the Customer Specific logic to set the <pin>_ie high and the <pin>_oe low during Test Mode. An exception to this rule is a bidirectional pin that is used as a ‘scan_out’.

B.2.10 MCU PIG/MIG Interface

PIG (Peripheral Interface Gasket) and MIG (Module Interface Gasket) are blocks that interface the MCU peripherals and the MCU bus. PIG interfaces with the MCU and interfaces with a MIG in each peripheral.

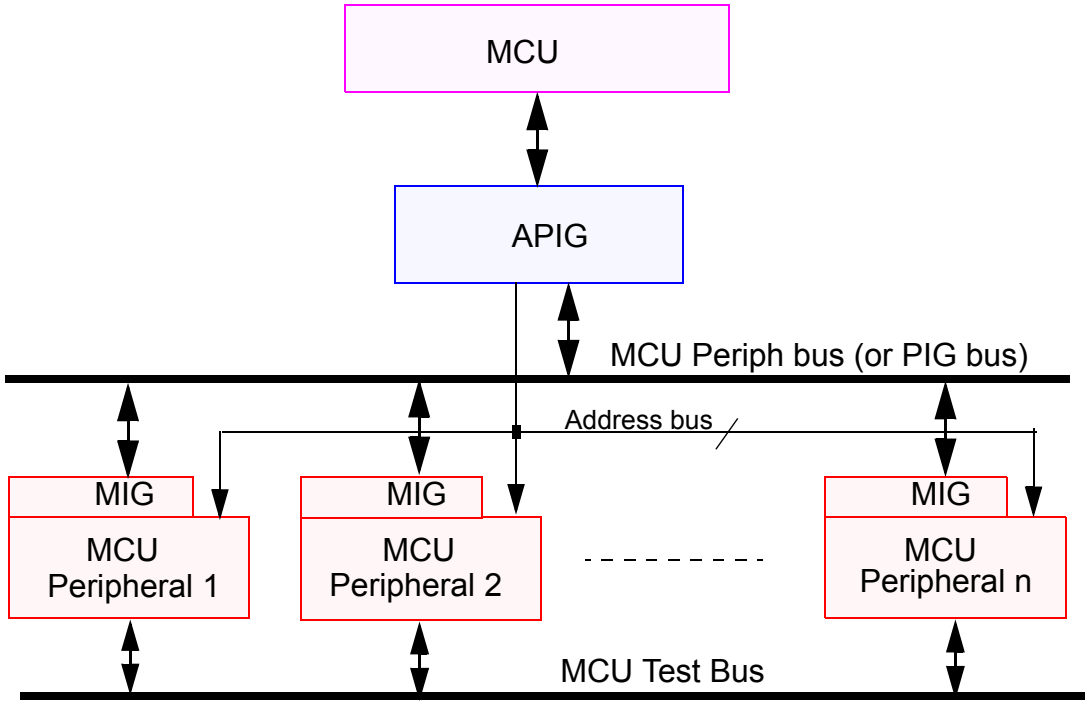


Figure B-7. MCU Peripheral Bus Interface

The PIG has been designed to be a synthesizable MuxDFF Full Scan module. The MIG control submodule has been changed to be compliant to the new PIG bus methodology. The boundary scan in the MIG is used to register data from the bus before entering combinational logic. This is done to help failure analysis determine if the problem is in the driving module or receiving module.

Each module on the PIG bus has a self contained scan chain. During the scan chain shifting, all signals on the PIG bus should be tri-stated. Only during the capture clock cycle will data be allowed to propagate from module to module. The PIG will drive several of these signals during the capture clock cycle. The ATPG process will only generate patterns which do not generate bus contention.

The PIG will be part of several scan groups because it functionally drives the PIG bus. In most tests, the PIG faults will be removed and the PIG will only be used to drive the bus. This will allow faults on the inputs of peripherals on the PIG bus be detected.

B.2.11 MCU TAN-line Interface to Test Control Module (TCM)

(PRELIMINARY)- This implementation does not correctly resolve the test_reset.

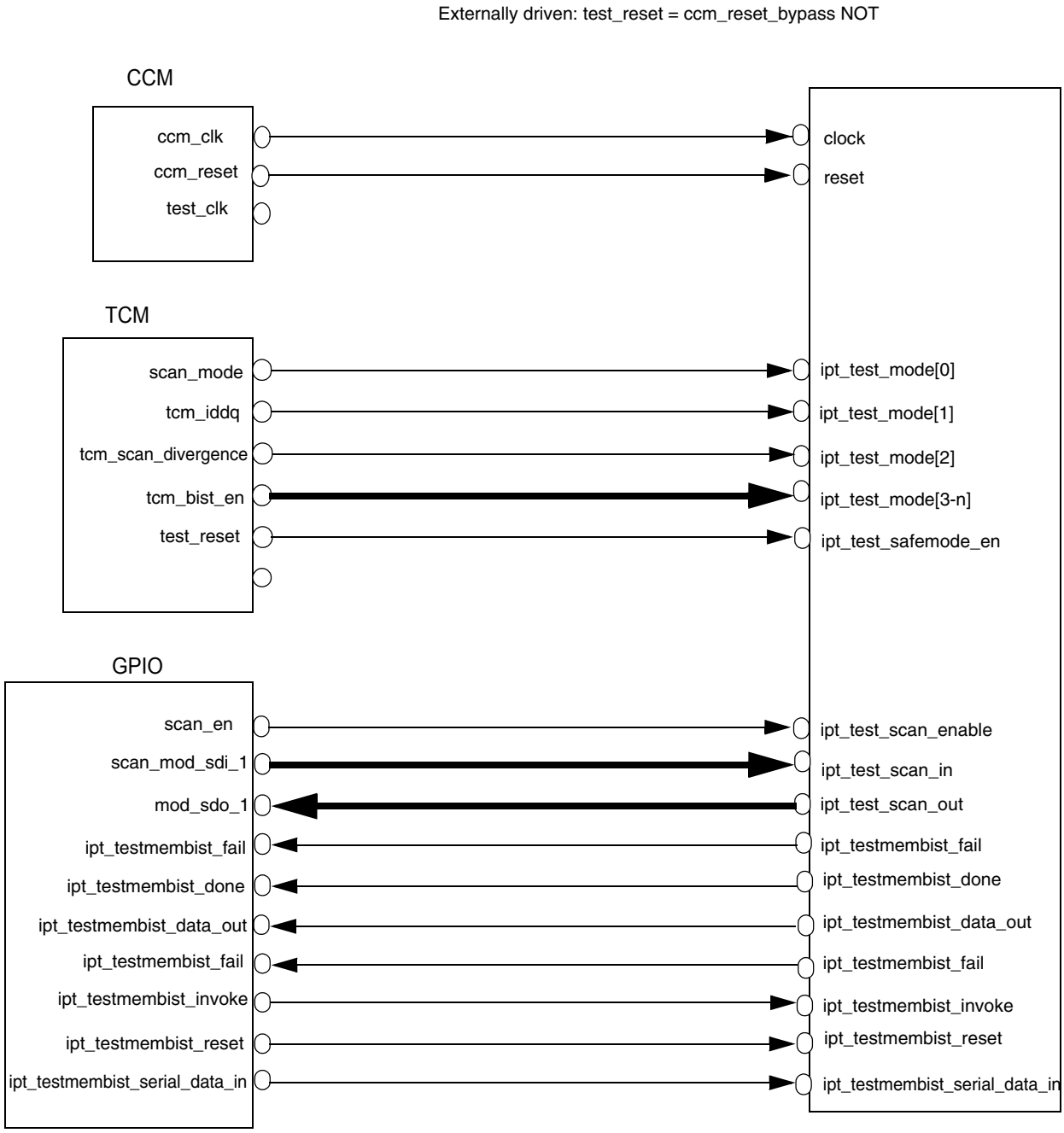


Figure B-8.

B.2.12 BIST/Memory Test

Memory BIST on Neptune is being redesigned by the ASP group. There are two BIST controllers, one for RAMs and one for ROM. For more information see the memory BIST section of this specification. It is the goal to have only one BIST for RAM and one for ROM. The exception to this might be if the BIST engine is enhanced to do redundancy correction for large arrays.

The MLB_TEST block located into the Test Control Module allows us to test internal and external MCU memories through the RAHB bus. More details is given in the TCM specifications.

Refer to the memory section of this specification for more details.

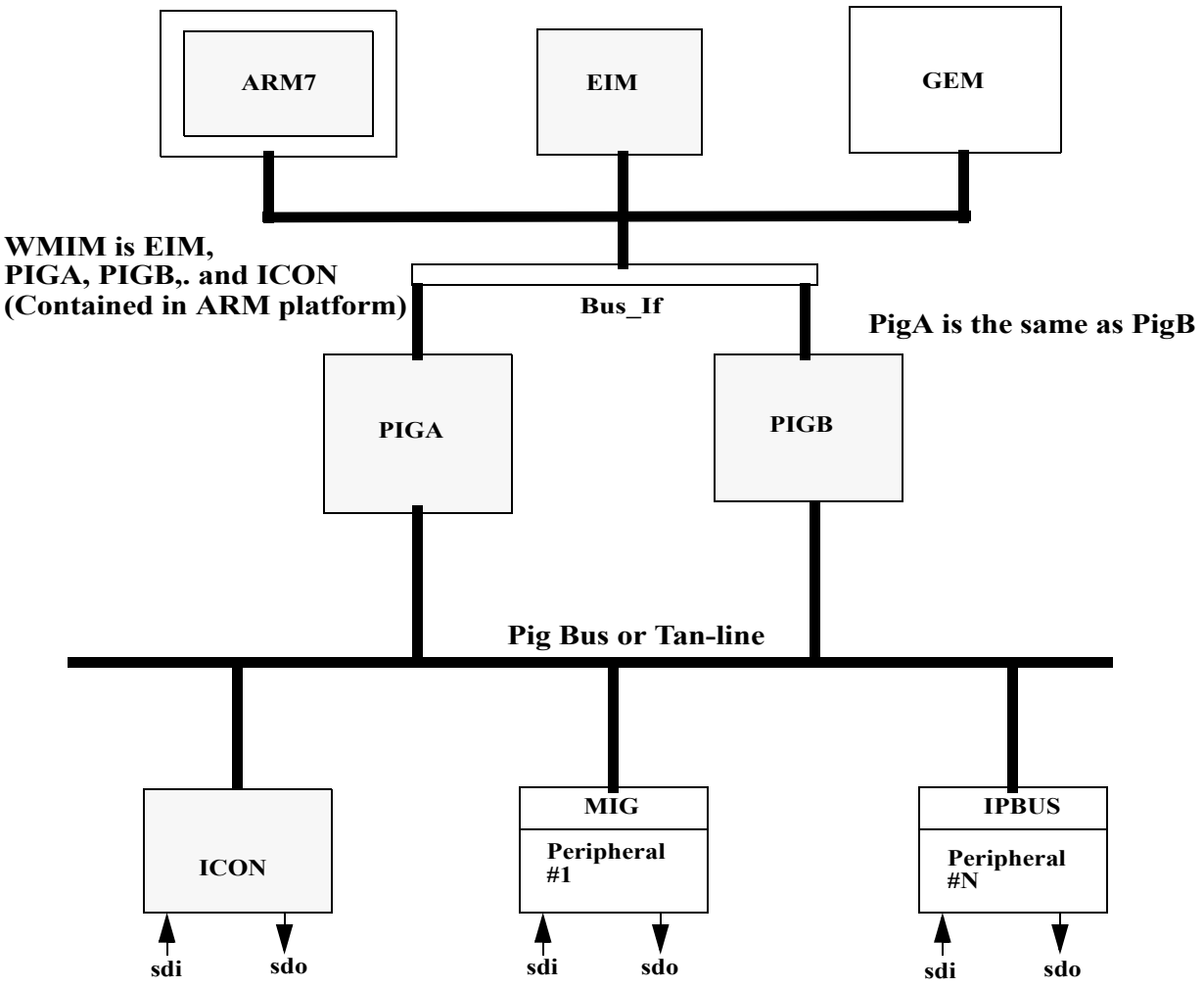
B.2.13 PIG Test

The PIG is a module designed in the same manner as other scan-based with MuxDFF full scan in the ARM Platform. It is tested with a scan pattern generated by the ATPG tool modules. To obtain the best possible test coverage of the PIG, care must be taken grouped with the PIG.

The scan methodology for the PIG bus has been re-architected from the Patriot HIP6W.During scan mode the tristate drivers can be efficiently tested during capture cycle. Previously, no module to module communication was allowed. All busses were unknown. Now a module can drive the bus during the capture clock cycle of scan testing.

The ability to drive and test the PIG bus will not significantly improve test coverage. Only minimal improvement to test coverage will be observed. However, test coverage is NOT the only metric to improve overall quality of the testing. The PIG bus wires take up noticeable area. The proper operation of this bus is critical to the overall operation of the Neptune. Testing this bus via scan will improve the failure analysis of the chip. Also, testing of the PIG bus allows the bus speed to be checked via scan using path and transition fault module. Critical timing paths from module to module can not be targeted with path delay testing. The last benefit to scan testing the bus will be to enable proper IDDQ scan based testing.

A few challenges exist with this methodology that must be overcome to insure proper functionality. A one clock cycle access from module to module across the PIG bus must be possible. If a functional path is a multi-cycle path, these paths will not be testable via scan based testing. Also, for verification of scan pattern using Verilog, full SDF annotation MUST BE provided for the chip level. In the past, this has not been available.



- Some peripheral scan groups will have the ARM platform wrapper chain

Figure B-9. Pig Bus Control During Scan Test

B.3 Chip Scan Control

Controlling the clocks during scan testing is crucial to obtain scan patterns that have a high fault coverage, are manufacturable, repeatable, and failure analysis friendly.

Neptune will have two basic clocking schemes during scan testing. The legacy modules will follow the Patriot clocking strategy. New modules will use a single clock strategy not utilizing the test_clk signal. Each approach has advantages and disadvantages which will be described in this section.

B.3.1 Clock Control for Scan Testing of Legacy Modules

The Neptune scan methodology is dependent on strict adherence to the Neptune clock methodology. While in scan test mode, all clocks are driven by the test_clock. The test_clock is being driven from a primary input pin during scan mode testing. **The test_clk pin becomes configured as a scan clock during fast scan initialization sequence. SCZA is the test clock.**

The scan insertion strategy being used assumes that only one clock domain exists (i.e., no special scan ordering is required for scan flops, ordering is necessary for scan latches.)

The basic clocking scheme consists of 2 clocking types: positive clock tree, and negative clock tree.

Figure B-10 depicts the Neptune positive clock tree methodology. After the mux and the treebuf, no other logic must exist on the clock net. Only cells inserted to balance the clock tree are allowed after the treebuf.

The disadvantage of this approach is that the clock gating logic is not being tested.

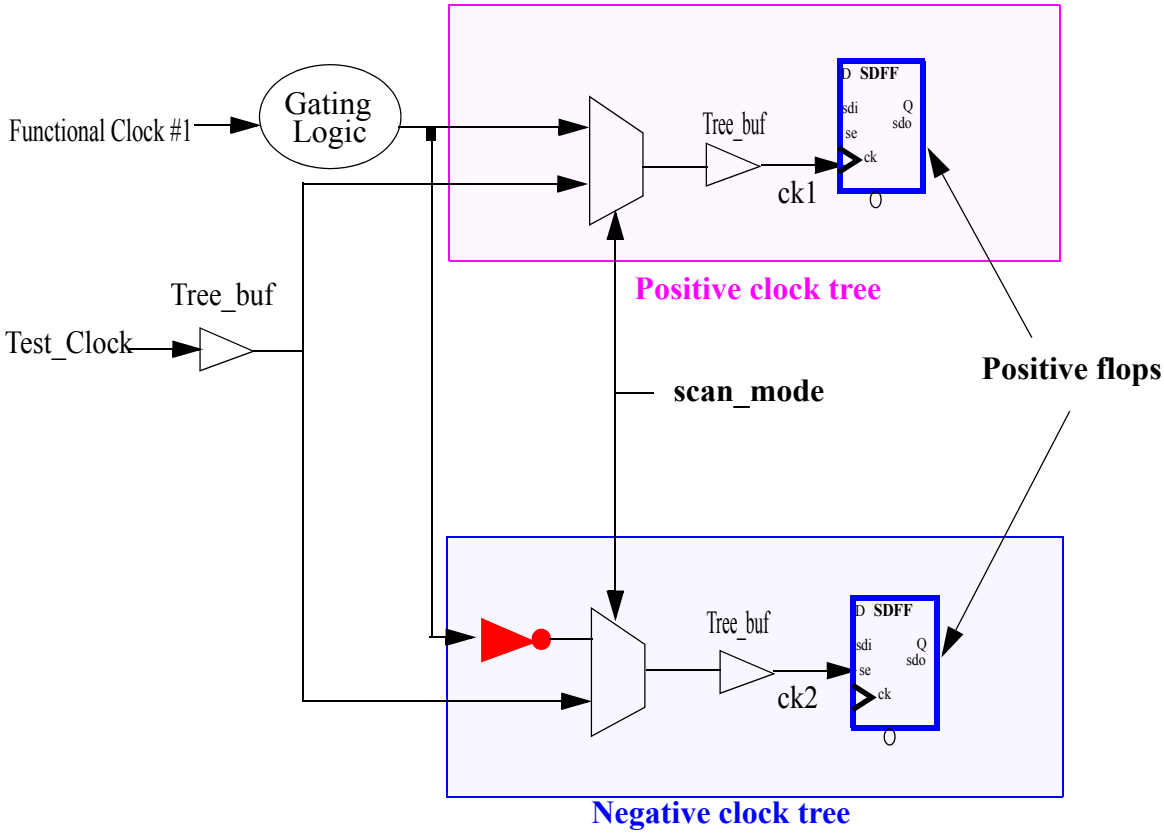


Figure B-10. Basic Clock Tree Methodology for Legacy Modules

If a negative edge triggered device is needed, a clock with an inverter prior to the mux must be used as shown above. All flops should positive edge triggered devices.

An exception to the above clocking rule can be found in modules that require a clock divider within the module. Figure B-11 depicts this type of clock divider network.

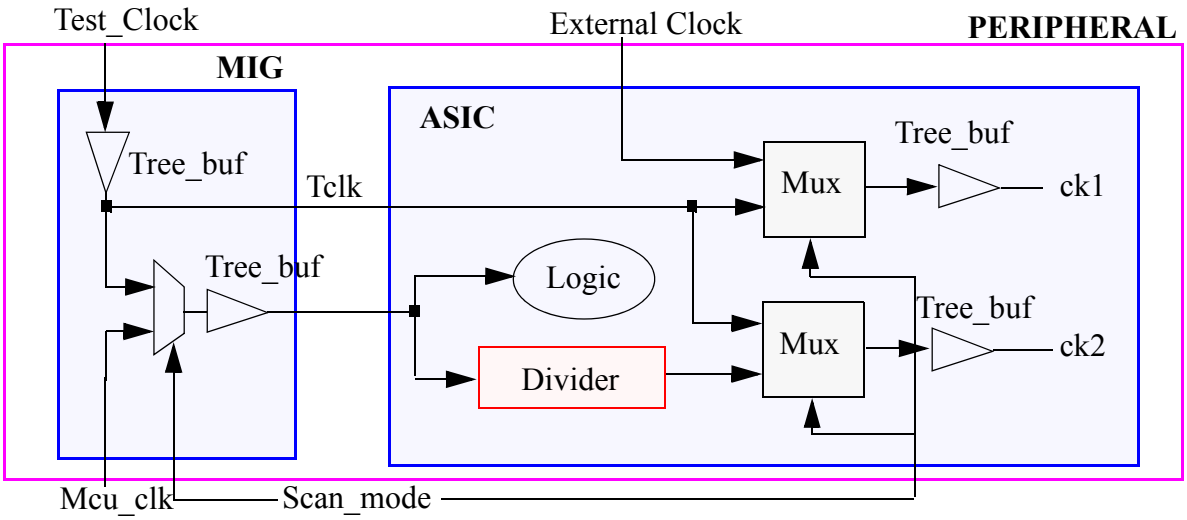


Figure B-11. Clock Tree with Clock Divider

In a few cases, registers might also be added to the module specifically for test reasons. The registers will not have a functional purpose.

The following figure depicts how to handle such special registers specifically for test purposes.

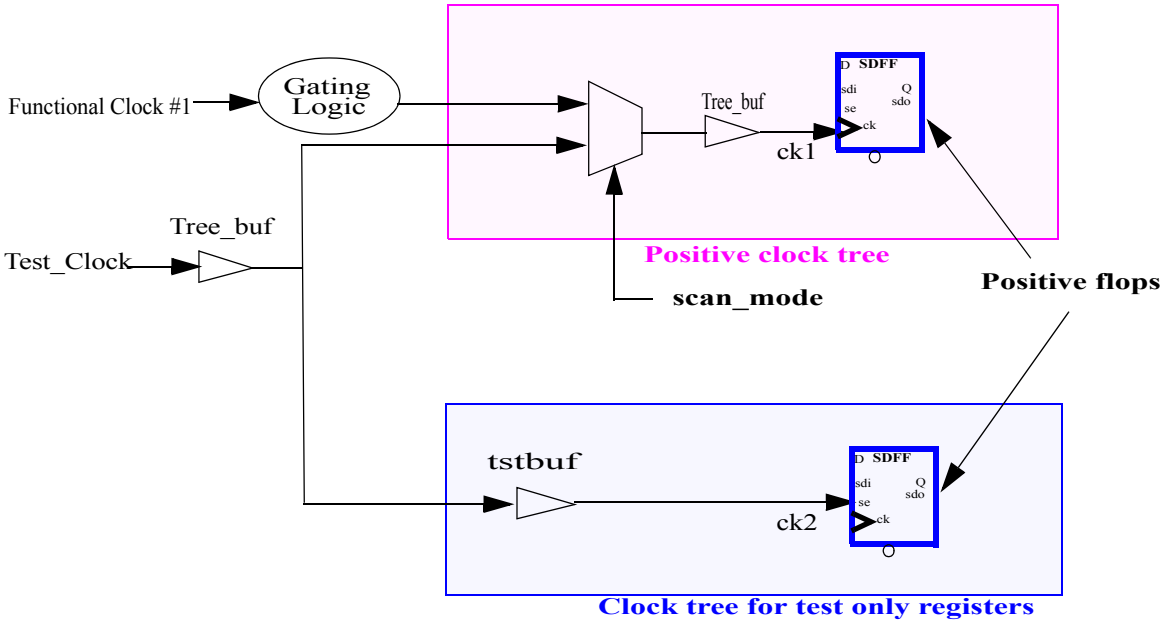


Figure B-12. Clock Tree for Test Only Registers

All of the clock methods described will have the test_clock active on all clock trees when scan_mode is enabled. This allows the chains to be generated independently of the actual physical clock domain.

Although one clock is fanning out to all the registers, it is doing so over different physical clock domains. This causes clock skew issues in the scan chain. This can cause significant problems with setup and hold time violations on the scan chain. The Neptune methodology addresses this problem by finding timing issues with re-simulation in Verilog using full back-annotated timing. These issues are then fixed by ECO or design change.

B.3.2 New Module Clocking Methodology

The ARM platform and new modules are implementing a clock gating strategy as shown in Figure B-13. The clock in MUST come from the ECB_N pin via the CCM module.A test case is being developed to compare the quality of the generated pattern sets to those generated for a non-gated circuit. More will be added in the next release of this specification. Results from another chip using this general methodology shows 6%-10% increase in pattern size due to this clock gate circuitry.

NOTE:

Scan_mode will replace scan-enable for version 2 of the Neptune chip.

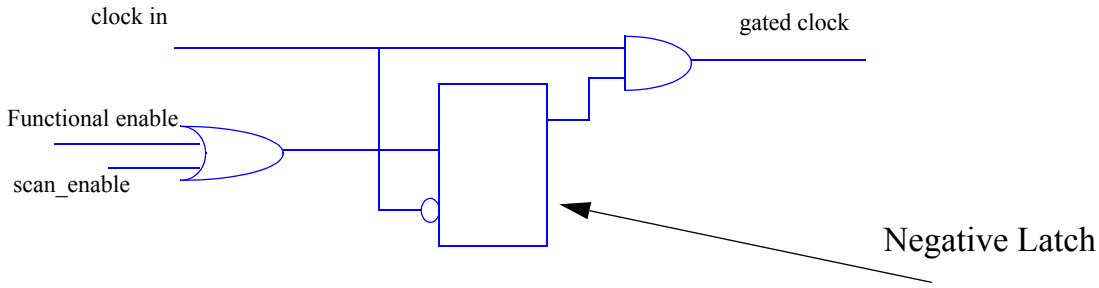


Figure B-13. Clock Gating Strategy for New Modules

This approach has the advantage of fully testing the clock gating logic via scan. However, if negative flops or latches are used within the block, the scan chain will need to be connected in a specific manner for the chain to shift correctly. Failure to correctly handle negative flops or latches within such modules will result in non-functioning scan.

Clock skew and insertion delay between the clock in at the module and the test_clk MUST BE MATCHED.

B.3.3 Test Register Block

The Test Register block contains 2 registers. One is the Testmode register and the second one is the Test Control register. Refer to Figure 3-5.

Testmode register generates scan_mode signals for each of the peripherals. Test Control register controls the test reset signal for each peripheral. The box Output Mux in Figure B-14 represents the series of Muxes **to configure the part to various scan modes.** Testmode register and Test Control register are described in the TEST_BLOCK specification. **Please refer to the TCM chapter form more information.**

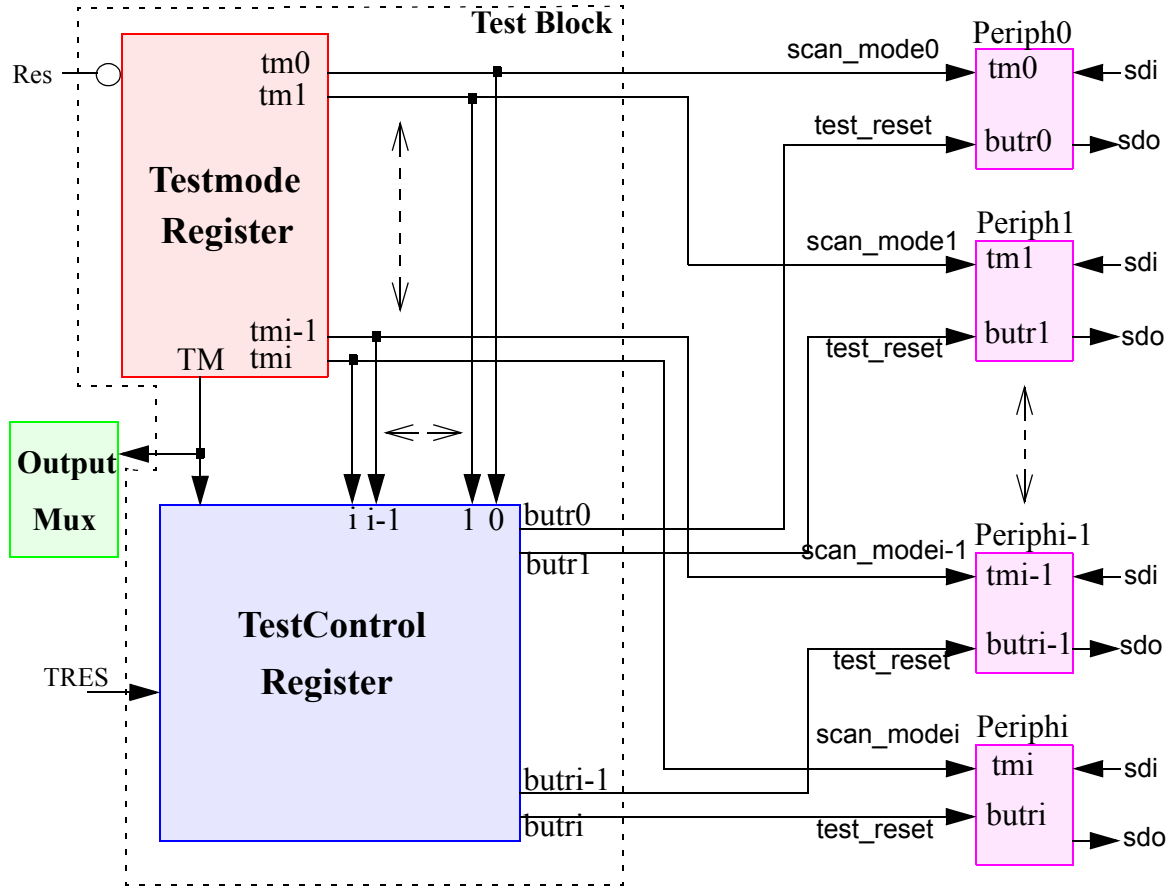


Figure B-14. Test Block/ Test Control Signals Distribution

B.3.4 External Clocks/Divided Clocks

There are 2 ways to route external or divided clocks in a DSP peripheral. The Designer can choose to send them to the PMBIF block so the buffering and the test clock methodology is taking care of, in that way PMBIF send back to the Asic the same buffered clock. Or, the designer can choose to manage his own clock inside the Asic is predominant, but in that way a mux with test_clock and controlled by scan_mode is mandatory. Nevertheless, the timing is going to be very critical, especially if different clocks are connected to a same scan chain, we take the risk to have clock skew problems.

Figure B-15 shows how to handle external/divided clocks routed to PMBIF. Figure B-16 represents clocks routed within the Asic without PMBIF interaction where a lot of care needs to be taken.

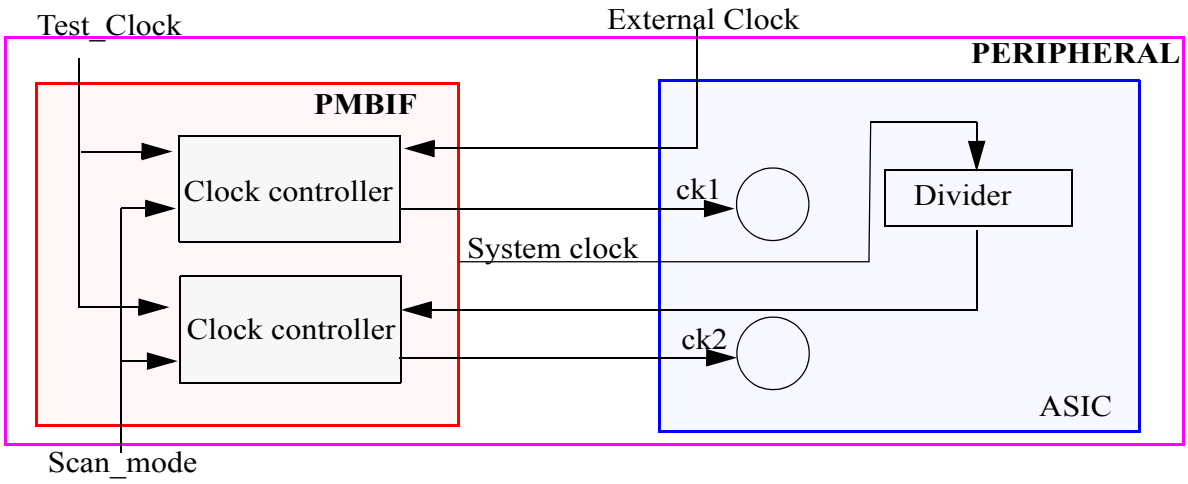


Figure B-15. External/Divided Clocks Routed to PMBIF

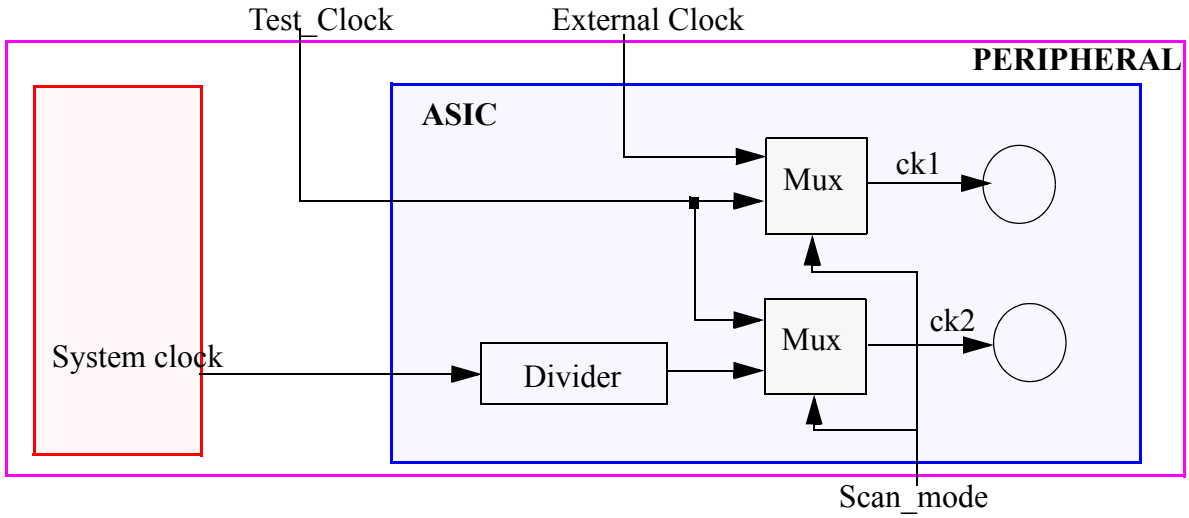


Figure B-16. External/Divided Clocks Routed Internally

B.4 Module Scan Requirements

The Test Control Module (TCM) controls the chip during test modes for all modules and cores. The legacy modules and new modules must be able to work within the same scan group. The new modules must achieve the same test functionality using the new design methodology for clocking, resets, and test control as legacy modules.

B.4.1 Module RESET Control for Legacy Modules

According to the Neptune Design Guidelines all registers can be reset. Functional reset and scan reset are different signals. Functional reset most often comes directly from the CCM reset (ccm_rst_b) or a derivative of it. For DSP peripherals using a PMBIF interface, PIRES is the functional reset signal. This is controlled during scan mode operation. Test reset (test_reset) is generated from the TCM using a primary input signal that is configured in the same manner as the sdi and sdo pins. The TCM can generate a unique test_reset signal for each module. If the module has been selected for scan testing (scan_mode=1), then

test_reset comes from a primary input, **STDA pin**. If the module has not been selected for scan testing (scan_mode=0), then test_reset is low. During chip reset, all test_reset and scan_mode signals are initialized to low. In test mode, **TRES_b** becomes a primary input to control reset signals of modules selected for scan testing.

Figure B-17 depicts what the reset logic within a module must look like. In this circuit, test_reset is active high while functional reset is active low. Also, test_reset takes priority over the functional reset signal.

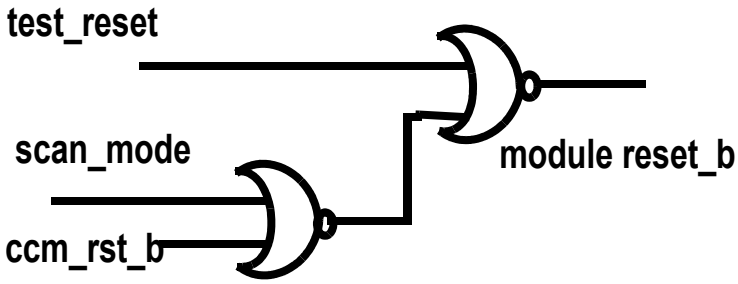


Figure B-17. Internal Module Reset Logic

B.4.2 Module RESET Control for New Modules

Reset during scan mode operation must be implemented with same functionality as the legacy modules. test_reset must take priority over functional reset in scan operation. If scan_mode = 0 and test_rest = 1, all tri-state devices must be disabled. Functional reset must become **inactive** if the module is in scan mode. It is under evaluation if this constraint can be removed.

It is the new module designers responsibility to have a that is functionally equivalent to that reset circuit as described in the legacy module section.

B.5 Removal of the Scan Data Output Tri-State Elements

The tri-state drivers traditionally on Patriot scan data outputs have also been removed for Neptune. This will make instantiating a specific library cell in RTL unnecessary. Thus allowing more portable and reusable RTL. Also, removal of tri-state elements simplifies the Static Timing Analysis (STA) effort.

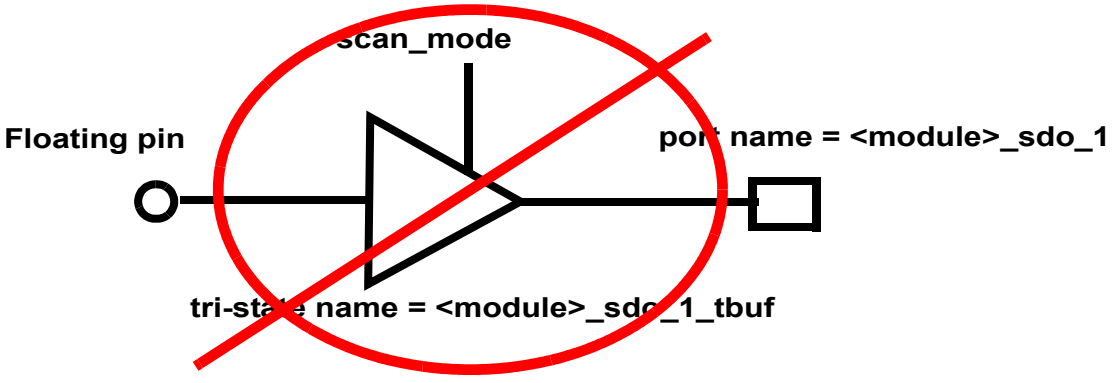


Figure B-18.

To accomplish removal of the SDO tri-state elements, a new port and mux has been added to the Neptune scannable modules (scan_mod_sdi_xx_bypass).

The new implementation using a standard mux is shown below. This change will remove instantiations of technology specific cells in the RTL code. This change will also simplify the static timing analysis stages of the design.

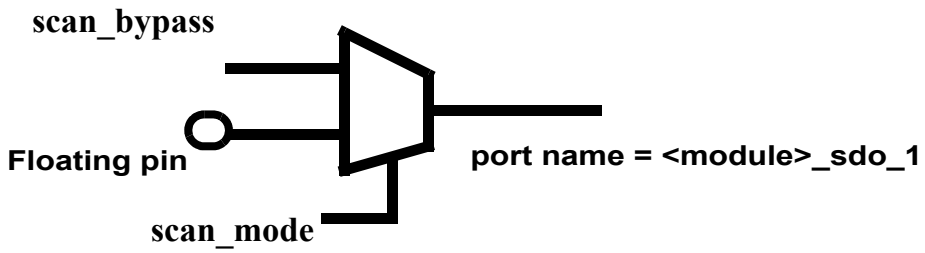


Figure B-19. NEW SDO Output Convention for Neptune

The diagram above shows that in RTL, the b input of the mux is left floating. The b input will be connected to the last flip flop in the scan chain. The a input to the mux is connected to another mux at the front of the scan chain to allow a bypass function.

B.5.1 Module SDO Output Signal(s)

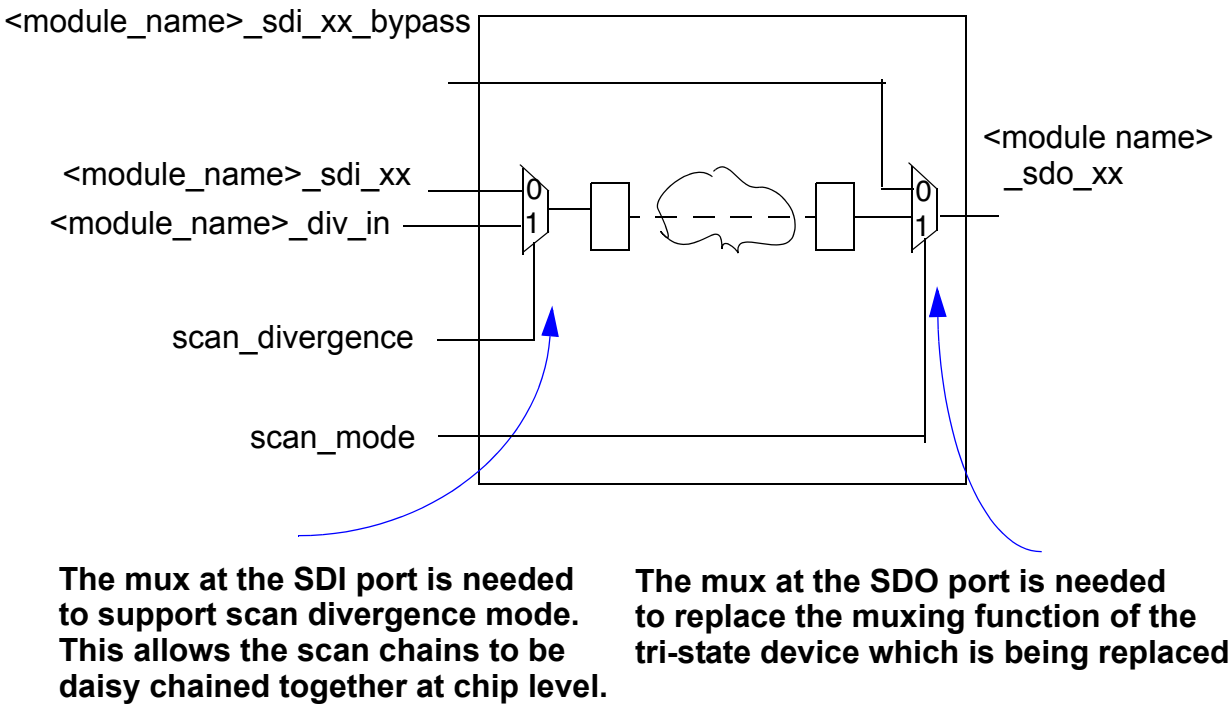


Figure B-20.

Additional details for scan connections and scan divergence section of this specification.

B.5.2 Module Scan Enable

The scan enable signal (scan_en) is a common input to all scannable modules. This signal is mux's in from a primary pin in the same manner as the sdi and sdo signals. Since this is a shared signal, all scannable modules scan_en will toggle when any scan test is active. This could be a problem if your module is not in the scan group selected. (i.e.,other modules are being scan tested but not your module)

Internal to each module, scan_en must be AND'd with scan_mode. Since this signal is specific to each module, this will prohibit the internal ports on all the registers from toggling.

Figure B-21 depicts the logic necessary to generate internal <module>_scan_en. This signal will be used by the scan insertion script to connect to all registers.

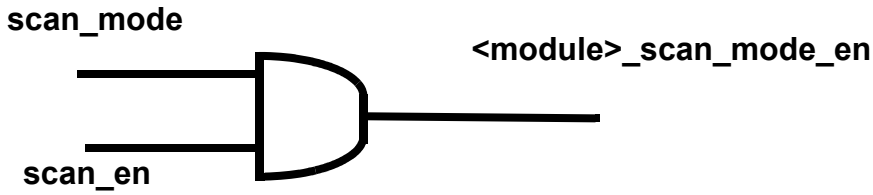


Figure B-21. Scan Enable Logic

B.5.3 Module Tri-State Enables for Functional Signals

Neptune has two main tri-state busses. There are also other tri-state signals within the Neptune. Since scan is now testing these busses, care must be taken to avoid contention during scan operation. A standard enable circuit has been developed that is scan friendly.

This circuit turns off the tri-state when in shift mode. While in functional or scan capture mode the flip flop determines if the enable id is active or not. The flip-flip must be scannable and have the reset operate as described above in the reset section. This simple circuit is shown in Figure B-22.

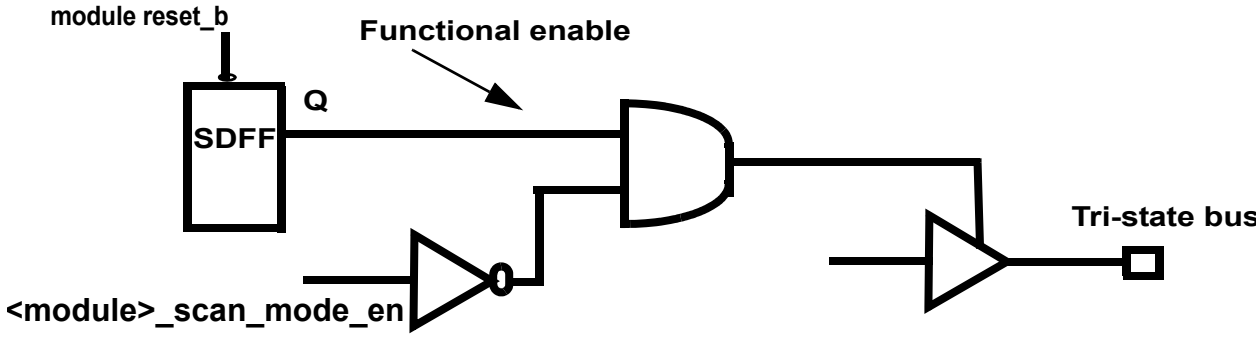


Figure B-22. Tri-State Logic

Table B-1 shows the relationship between the scan control signals, mode of operation, and the sdo tri-state modes. When it is not being scanned, it is in reset mode. By design, the modules will not drive the tri-state bus and no DFT feature is required.

Table B-1. Tri-State Enable Configuration

scan_mode	scan_en	ccm_rst_b	test_reset	tbuf enable	tbuf out	Comments
X	X	0	0	X	Z	Hard Reset
0	0	1	0	0	Z	Normal Mode
0	0	1	0	1	Data	Normal Mode
0	X	X	1	0	Z	Scan Other Mods
1	1	X	0	X	Z	Mod Shift
1	0	X	0	1	Data	Mod Capture

B.5.4 Summary of all the Scan Control in a DSP or MCU Peripheral

The test_reset net will be negated in functional mode. In functional mode, the bit that controls test_reset is cleared in the test_register block and scan_mode=0. During scan test mode, if the block is under test, scan_mode=1, so X=0 and test_reset signal can control all the internal reset pins to increase the test coverage. If the block is not under test but we are testing another peripheral, scan_mode=0 and test_reset=1, so the internal reset is enabled and keeps the block in a reset state.

It is a requirement that test_reset have priority over functional resets and is controllable via the TCM in test mode.

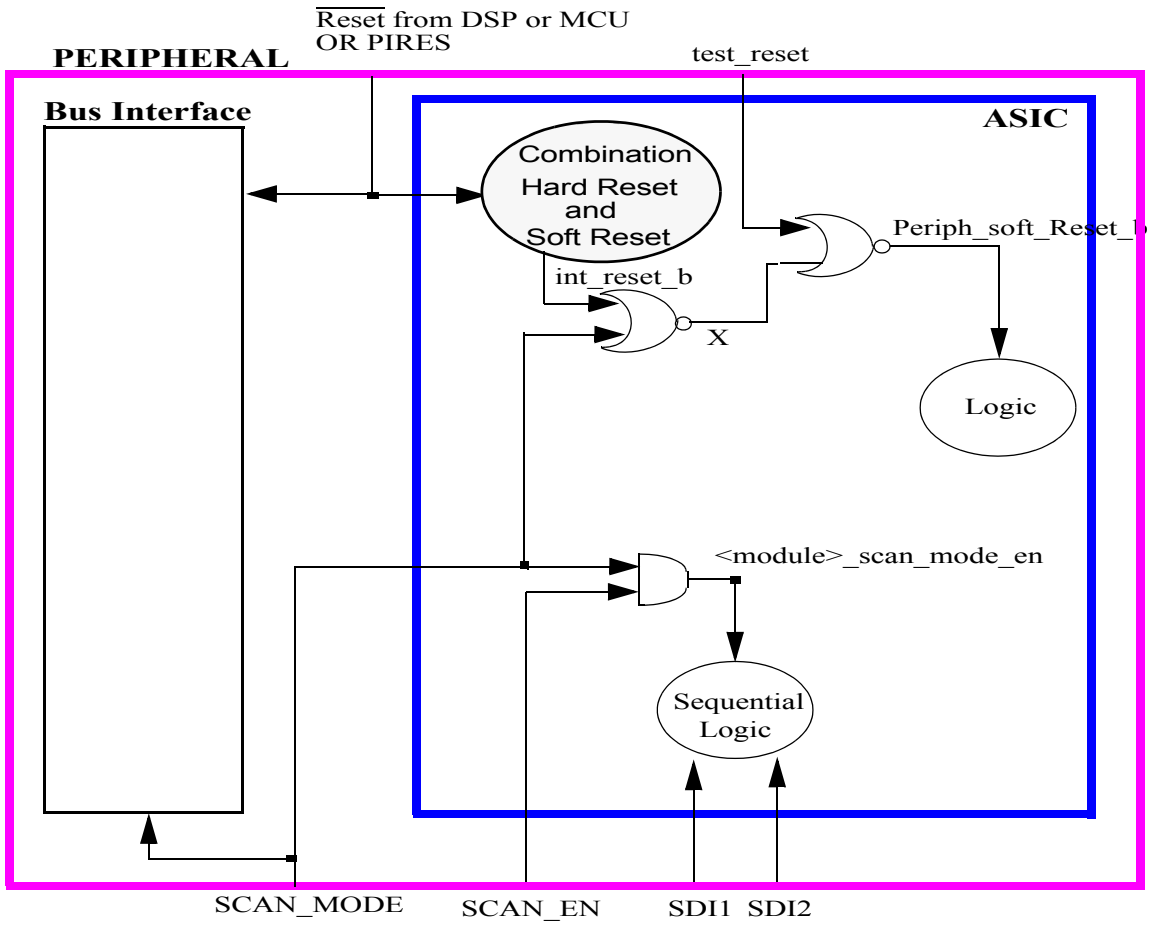


Figure B-23. Scan Control Summary

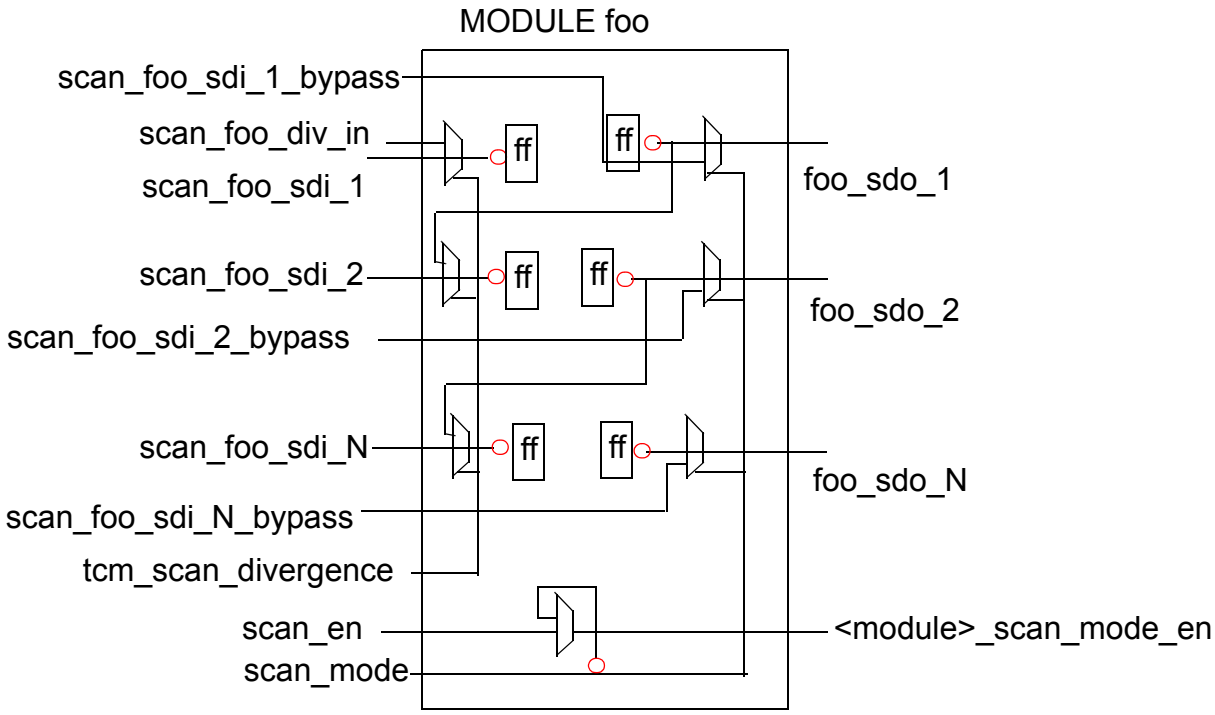


Figure B-24. Pre-Scan Insertion Example

In the example above, module foo has 3 scan chains. The connections, logic and ports required prior to scan insertion are illustrated above. Figure B-24 does not include clock and reset connections or associated logic. Please refer to the Neptune specification for clock and reset requirements for scan and test modes.

The RTL coding example on the next page implements the logic shown in Figure B-24.

NOTE:

The scan insertion script is being rewritten and will be upgraded for version 2 of the Neptune chip. The module foo will also be updated.

B.6 RTL Coding Examples

Code Example 0-1. RTL Code

```
// ---------------------------------------------------- //
// ---------------  RTL Coding Example ---------------- //
// ---------------------------------------------------- //
// The following RTL is an example to show the intent  //
// of the scan related signals and logic affecting RTL //
// code. All naming conventions with respect to the   //
// test related signals have been followed. This      //
// example should reflect the PRE-SCAN INSERTION       //
// diagram. Each designer is responsible for making   //
// sure all test related functions in the DFT          //
// methodology spec are followed.                      //
// ---------------------------------------------------- //
// Version   Date         Comments       //
//   A      Dec-04-00Initial creation.  -JJ         //
//   B           Dec-04-00Instantiate submodules for     //
//                     front_mux and back_mux. -JJ&ES //
// ---------------------------------------------------- //

module foo (scan_mode, scan_en, test_reset, test_clk,
            scan_foo_sdi_1, scan_foo_sdi_1_bypass,
            scan_foo_sdi_2, scan_foo_sdi_2_bypass,
            scan_foo_sdi_3, scan_foo_sdi_3_bypass,
            tcm_scan_divergence, scan_foo_div_in,
            foo_sdo_1, foo_sdo_2, foo_sdo_3);

input scan_mode;
input scan_en;
input test_reset;
input test_clk;
input scan_foo_sdi_1;
input scan_foo_sdi_2;
input scan_foo_sdi_3;
input scan_foo_sdi_1_bypass;
input scan_foo_sdi_2_bypass;
input scan_foo_sdi_3_bypass;
input tcm_scan_divergence;
input scan_foo_div_in;

output foo_sdo_1;
output foo_sdo_2;
output foo_sdo_3;

wire foo_sdi_1_in, foo_sdi_2_in, foo_sdi_3_in;
wire foo_sdo_1_out, foo_sdo_2_out, foo_sdo_3_out;

front_mux front_mux ( .foo_sdi_1_in (foo_sdi_1_in),
                      .foo_sdi_2_in (foo_sdi_2_in),
                      .foo_sdi_3_in (foo_sdi_3_in),
                      .foo_sdo_1_out (foo_sdo_1_out),
                      .foo_sdo_2_out (foo_sdo_2_out),
                      .scan_foo_sdi_1 (scan_foo_sdi_1),
                      .scan_foo_sdi_2 (scan_foo_sdi_2),
                      .scan_foo_sdi_3 (scan_foo_sdi_3),
                      .tcm_scan_divergence (tcm_scan_divergence),
                      .scan_foo_div_in (scan_foo_div_in)
                    );

back_mux back_mux ( .foo_sdo_1 (foo_sdo_1),
                    .foo_sdo_2 (foo_sdo_2),
                    .foo_sdo_3 (foo_sdo_3),
                    .foo_sdo_1_out (foo_sdo_1_out),
                    .foo_sdo_2_out (foo_sdo_2_out),
```

```
                    .foo_sdo_3_out (foo_sdo_3_out),
                    .scan_foo_sdi_1_bypass (scan_foo_sdi_1_bypass),
                    .scan_foo_sdi_2_bypass (scan_foo_sdi_2_bypass),
                    .scan_foo_sdi_3_bypass (scan_foo_sdi_3_bypass),
                    .scan_mode (scan_mode)
                  );

endmodule

module front_mux (foo_sdi_1_in, foo_sdi_2_in, foo_sdi_3_in,
                  foo_sdo_1_out, foo_sdo_2_out,
                  scan_foo_sdi_1, scan_foo_sdi_2, scan_foo_sdi_3,
                  tcm_scan_divergence, scan_foo_div_in);

input scan_foo_sdi_1;
input scan_foo_sdi_2;
input scan_foo_sdi_3;
input foo_sdo_1_out;
input foo_sdo_2_out;
input tcm_scan_divergence;
input scan_foo_div_in;

output foo_sdi_1_in;
output foo_sdi_2_in;
output foo_sdi_3_in;

//This section is to create the mux in front of the scan chain on sdi

assign foo_sdi_1_in = tcm_scan_divergence ? scan_foo_div_in : scan_foo_sdi_1;
assign foo_sdi_2_in = tcm_scan_divergence ? foo_sdo_1_out : scan_foo_sdi_2;
assign foo_sdi_3_in = tcm_scan_divergence ? foo_sdo_2_out : scan_foo_sdi_3;

endmodule


module back_mux (foo_sdo_1, foo_sdo_2, foo_sdo_3,
                 foo_sdo_1_out, foo_sdo_2_out, foo_sdo_3_out,
                 scan_foo_sdi_1_bypass, scan_foo_sdi_2_bypass,
                 scan_foo_sdi_3_bypass, scan_mode);

input foo_sdo_1_out;
input foo_sdo_2_out;
input foo_sdo_3_out;
input scan_foo_sdi_1_bypass;
input scan_foo_sdi_2_bypass;
input scan_foo_sdi_3_bypass;
input scan_mode;

output foo_sdo_1;
output foo_sdo_2;
output foo_sdo_3;

//This section is to create the mux at the back of the scan chain on sdo

assign foo_sdo_1 = scan_mode ? foo_sdo_1_out : scan_foo_sdi_1_bypass;
assign foo_sdo_2 = scan_mode ? foo_sdo_2_out : scan_foo_sdi_2_bypass;
assign foo_sdo_3 = scan_mode ? foo_sdo_3_out : scan_foo_sdi_3_bypass;
```

endmodule **(Need to add figs for rev 2)**

NOTE:

The gates representation will be updated in version 2 of the Neptune chip.

B.7 Neptune DFT Core Scan Requirements

Integration of a core into the Neptune must follow the same basic methodology as required in the “Patriot DFT Methodology” document. Additional core requirements for proper integration are included in the following sections. This section will be improved in the next release of this specification.

B.7.1 Core Requirements

Requirements to integrate the S-OnyxU 56600S DSP core and ARM7TDMI-S core into the Neptune DFT methodology involves the following:

- **Versions of the code**
- **Analyzing the DFT architecture of the embedded cores**
- **SRS deliverables for the core**

verifying software and versions, supporting special test modes, design requirements, and specific deliverables.

The requirements for cores are the same as previously described in the earlier sections for peripherals unless specified in this section.

B.7.1.1 Software (versions must be synchronized with Neptune requirements)

- Verilog XL
- Fastscan
- Test Compiler Plus
- IMS - DVT

B.7.1.2 Design Requirements for Integration

- 100% compliance with Patriot DFT methodology
- Verilog model - STUB for cores and s-pmbif
- Min shift speed should be 30MHz & Max capture speed should be greater or equal to the system clock rate.
- Maximum Scan Chain Length should be around 100-150 elements per chain, with only one exception: For N=Number of Scan Chains allowed, if #FF/150 > N, then it is possible to increase the number of scan elements per chain. Please contact AWDC DFT team if the scan elements per chain is greater than 150 as required.
- Power estimates for scan operation

B.7.1.3 Test Modes

Table 2.1 shows the required test modes and a brief description of each mode of operation for the S-OnyxU DSP56600S core and ARM7TDMI-S core in Neptune.

Table B-2.

Mode of Operation	S-OnyxU 56600S DSP Core	ARM7TDMI-S MCU Core	Descriptions
Normal	Yes	Yes	Functional Mode, this is the chip normal operating mode.
Core Scan	Yes	Yes	Both S-OnyxU and ARM7 cores must be tested separately from the rest of the chip and the outputs will be controlled during other scan based testing.
Peripheral Scan	Yes	Yes	Peripheral scan test is done through a dedicated test bus containing scan input and scan output signals. See Module Scan Requirements section
Burn-in / Scan Divergence	Yes	Yes	All scan chains are configured into one long chain which will be exercised during the burn-in / scan divergence process. This enables the ability to stop a functional test when a failure is observed, stop all clocks, enter scan mode and shift out the data in all registers.
Bus Scan	Yes	Yes	DSP Bus Interface (PMBIF) and MCU Bus Interface (MIG and JIG). All bus interface are scannable
IDDQ	No	Yes	Quiescent Current testing, this is a form of testing where the chip is placed in a OFF state and after the combinational switching has settled, the current drawn is measured. This mode may be required if the core does not have the tristate buses, analog modules, and internal memories.

B.7.1.4 Deliverables

- Verilog design file with SDF back annotated files
- DFT integration guide
- SonyxU/ARM7 test guide
- Test Compiler scan insertion script and log files
- Scan chain order description file
- Fastscan ATPG Scripts and control files (dof files, procedure file, timeplate file)
- Fastscan library cells in addition to the standard HIP7 library
- Patterns: Stuck-at fault coverage > 98%, Stuck-at fault guaranteed coverage > 95%. Please note that appropriate documentations required if the stuck-at fault coverage goal of 98% is not achieved. IDDQ coverage > 90%, transition coverage > 80%, path delay as required
- Fastscan full reports with coverage and log files
- Parallel and serial Verilog patterns and serial WGL pattern
- DVT environment and pattern debugging help upon request

B.7.2 S-OnyxU 56600S Specifics

The S-OnyxU 56600S is a fully synthesizable DSP core with full scan testing capabilities. All flip flops of the core are included in the scan chains. There are no embedded memories and no multicycle paths in the core. There are no tristate in the core.Figure B-25 shows the block diagram between the S-OnyxU 56600S DSP core, DSP peripherals, and the S-PMB interface.

B.7.2.1 Testing DSP Core in Neptune

The S-OnyxU 56600S DSP core consists of 28 scan chains, each with maximum scan chain length between 100-150 scan elements. The DSP scan patterns are delivered from the DSP group with the core along with the required documentations and scripts. The Neptune DFT team will utilize the FastScan ATPG scripts as a reference and re-run ATPG for the core at the Neptune chip level. This will eliminate the post-processing stage of WGL pattern modifications to the original WGL files delivered by the DSP core group. This post-processing stage must add vectors to configure the Neptune into scan mode. This stage also modifies the pin declarations from the core ports to the chip pins. Any bus(s) connected to the core can also be tested via scan based testing.

The test interface signals for the DSP are the same general signals defined in the chip level overview section of the DFT methodology.

B.7.2.2 DSP Synthesizable PMBIF

The Synthesizable Peripheral Module Bus (S-PMB) includes a group of global signals that are used for control and data transfer between the Core/DMA and the Neptune modules. The S-PMB interface is a fully DFT compliant synthesizable module. It is designed to use the flip-flops instead of latches contain no tristate signals or bi-directional buses. All bi-directional signals were split into 2 signals. It does not generate any clock signals to be used by the Customer Specific logic, the global clock signal (gclkw) of the chip is input directly into the Customer Specific logic.

SPMBIF will isolates the peripheral from the timing issues. The DSP clock speed is aggressive and is totally controlled by the interface. The same is true for Address and Data buses. The system clock (gclkw) is supplied to the S-PMB interface and muxed with the test_clk to generate the gating functionalities. Figure B-25 describes the implementation of the clock multiplexing and gating within the S-PMB Interface.

Figure B-25 shows the S-PMBIF control during the scan test.

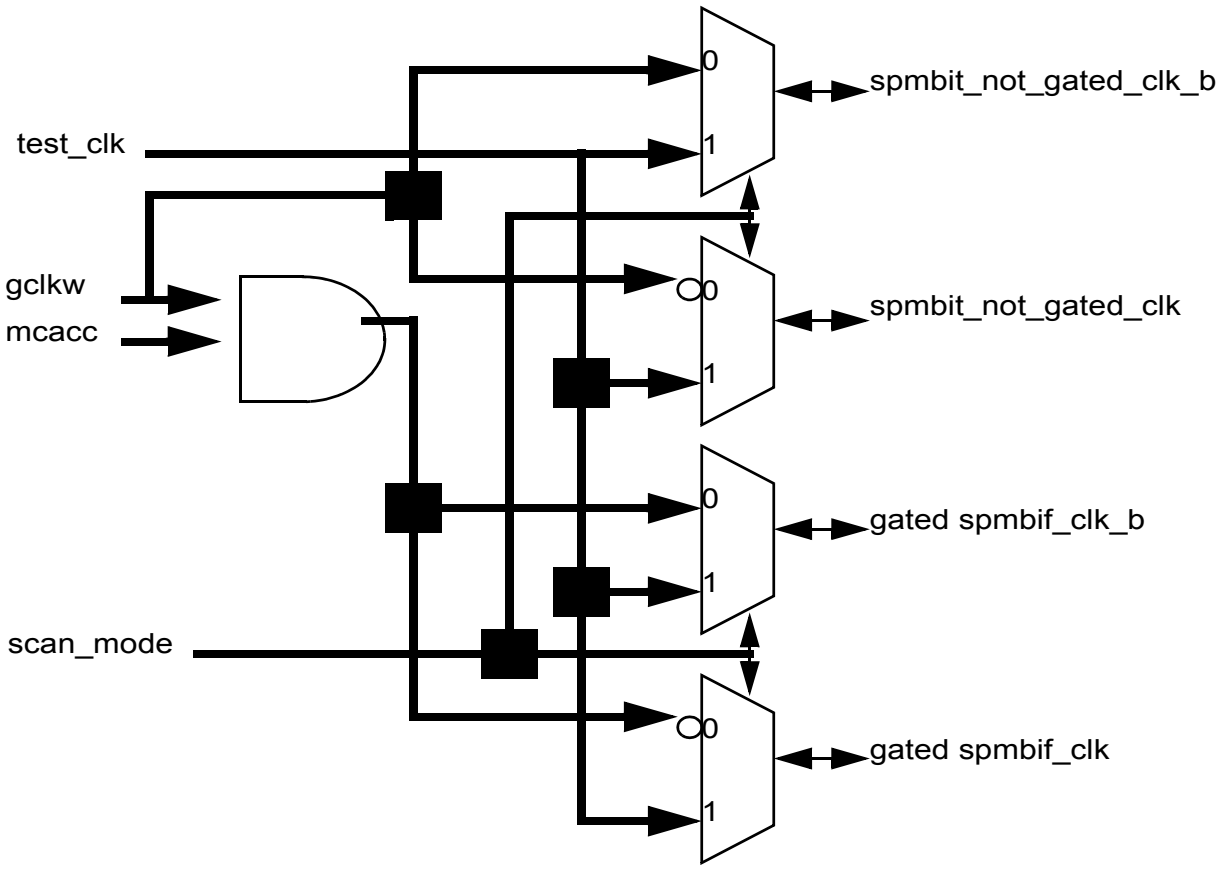


Figure B-25. S-PMB Interface Clock Logic

B.7.3 ARM7TDMI-S

The ARM7TDMI-S mcu core will be delivered as part of an ARM7 platform, rather than as a standalone core. The ARM7 platform and patterns will be delivered by EPS, along with the corresponding scripts and documentation. The Neptune DFT team will generate chip level patterns for the ARM7, using the EPS patterns and scripts as a reference. For Neptune there will be no WGL post-processing.

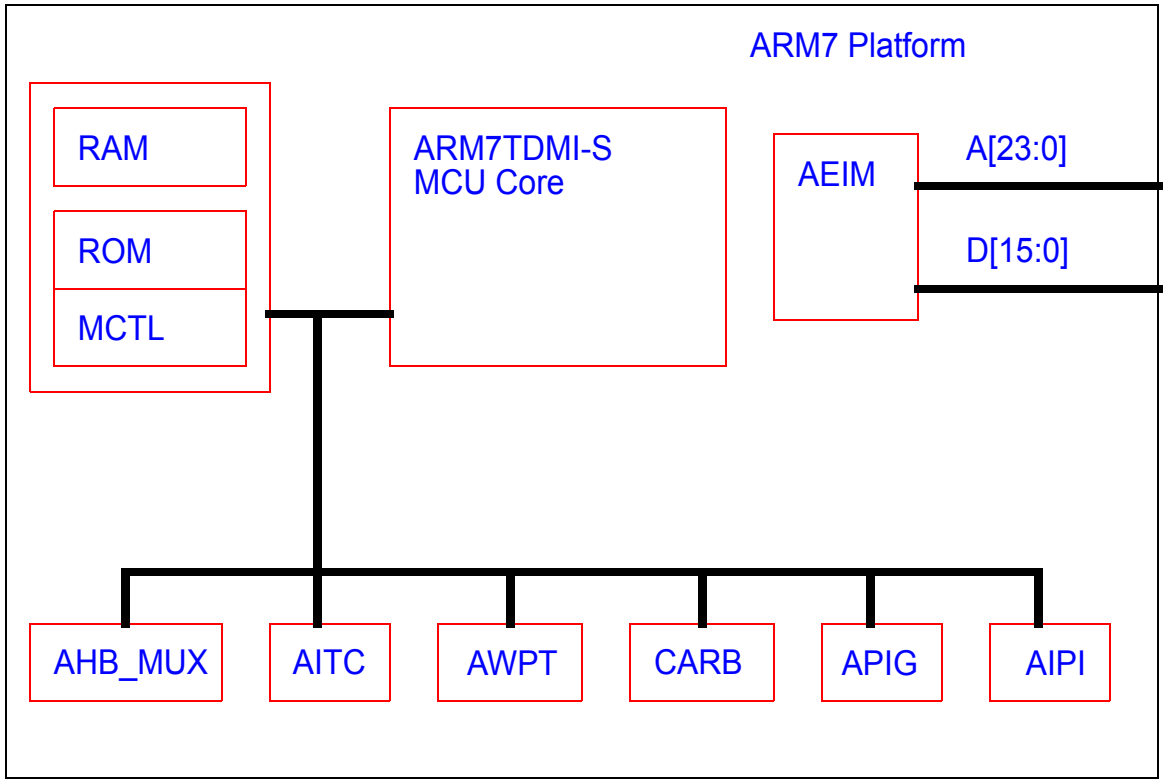


Figure B-26. ARM7 Platform

The number of ARM7 scan chains will be limited by the number of chains available for Neptune. Currently, Neptune has 27 scan chains available. The ARM7 platform will have 31 scan chains. Eight of these are dedicated to the ARM scan wrapper. 23 chains go to the core and peripherals that make up the ARM platform. When the ARM7 is running in **platform scan mode,** the 8 wrapper scan chains will be tied serially in pairs, making four wrapper scan chains. These four chains will run in parallel with the 23 ARM core and peripheral scan chains, totalling 27 parallel chains. When the Neptune is in MCU **ARM platform scan mode**, all eight of the ARM scan wrapper chains will be active, running in parallel. **The core scan chains are shut off**.

The limited number of scan chains available for Neptune requires most ARM7 scan chains to be longer than the maximum of 150 scan cells per chain. The exception to this is the eight scan wrapper scan chains. These will be held to 150 scan elements per chain to avoid slowing lengthening the MCU peripheral scan patterns. When running paired serially in ARM scan mode, the scan wrapper chains will be 300 scan elements long. The other 23 scan chains will average about 330 elements.

Table B-3. ARM7 Scan Chains

Scan Mode	Scan Wrapper Cells	ARM7 Platform Cells	Total Scan Chains
MCU Peripheral	8	23	31
MCU Core	4	23	27

Table B-4. ARM7 Platform Scan Cell Counts

Module	Instances in Arm Platform	Scan cells Per Instance	Total Scan Cells
ARM7TDMI-S	1	3749	3749
APIG	2	150	300
AITC	1	572	572
AWPT	1	2100	2100
AEIM	1	396	396
AHB_MUX	1	50	50
AIPI	1	100	100
MCTL	1	280	280
Wrapper	1	1200	1200
Total			**8747**

To allow for scan chains of fairly uniform length, scan chains for some of the ARM7 submodules will be connected serially with one another. The APIG1 and APIG2 will be connected in this way, as will the AHB_MUX and AIPI. Other scan chains will not be connected serially between ARM7 submodules inside the platform.

Table B-5. Proposed ARM7 Platform Non-Wrapper Scan Chains

Module	Scan Chains Available	Total Scan Cells	Average Chain Length
ARM7TDMI-S	11	3749	340
APIG1 & APIG2	1	300	300
AITC	2	572	286
AWPT	6	2100	350
AEIM	1	396	396
AHB_MUX & AIPI	1	150	150
MCTL	1	280	280
Total	**23**	**7547**	**330**

The test interface signals for the ARM7 platform will differ slightly from the SONYXU test signals. The ARM7 platform contains memories, in addition to the MCU core. To support IDDQ testing, the ARM7 must be able to turn off these memories. The tcm_iddq signal is required to accomplish this. See Figure B-27.

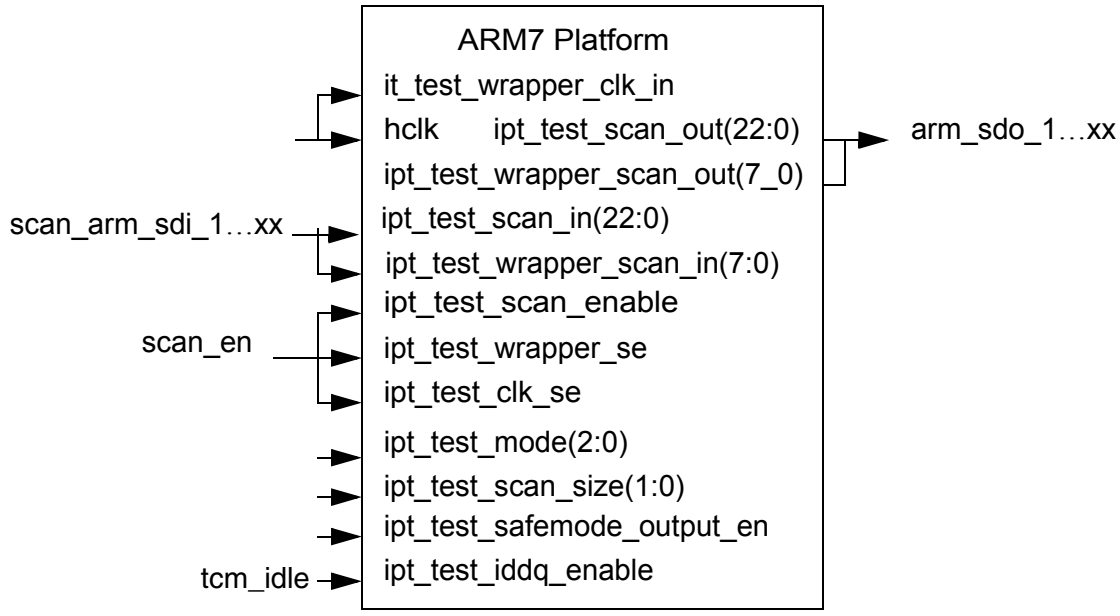


Figure B-27. ARM7 Platform Test Interface Signals

ipt_test_mode(): .These pins are used to control the operational mode of the ARM7 platform.

ipt_test_safemode_output_en(): This pin will put all put all ARM7 platform outputs in a safe state, and force all 3-state outputs to high impedance state.

ipt_test_iddq_enable(tcm_iddq): Disables any internal pullup/pulldown resistors for IDDQ test.

ipt_test_scan_size(): Determines the configuration of the ARM7 platform scan chains. These pins will be tied to a known value during integration.

ipt_test_scan_in(scan_arm_sdi_1…xx): These are the input pins for the ARM7 platform scan chains.

ipt_test_wrapper_scan_in(scan_arm_sdi_1…xx): These are the input pins for the ARM7 wrapper scan chains.

ipt_test_scan_out(arm_sdo_1..xx): Output pins for the ARM7 platform scan chains.

ipt_test_wrapper_scan_out(arm_sdo_1.xx): Output pins for the ARM7 wrapper scan chains.

ipt_test_scan_enable(scan_en): This pin provides the scan_enable signal for the ARM7 platform scan cells.

ipt_test_wrapper_se(scan_en): This pin provides the scan_enable signal for the ARM7 wrapper scan cells.

ipt_test_clk_se(scan_en): This pin provides the scan_enable signal to the clock control circuits of the ARM7 platform.

hclk(): This input provides the scan clock signal for the ARM7 platform scan chains.

ipt_test_wrapper_clk_in(): This input is used to generate the scan clock signal for the ARM7 wrapper scan chains.

To facilitate scan operations in Neptune, the ARM7 platform will include a scan wrapper. This allows the ARM7 inputs and outputs to be held at known states. The scan wrapper will use 8 scan chains to perform these functions. Additionally, test control logic and a scan path router are provided to work with the scan wrapper. See Figure B-28.

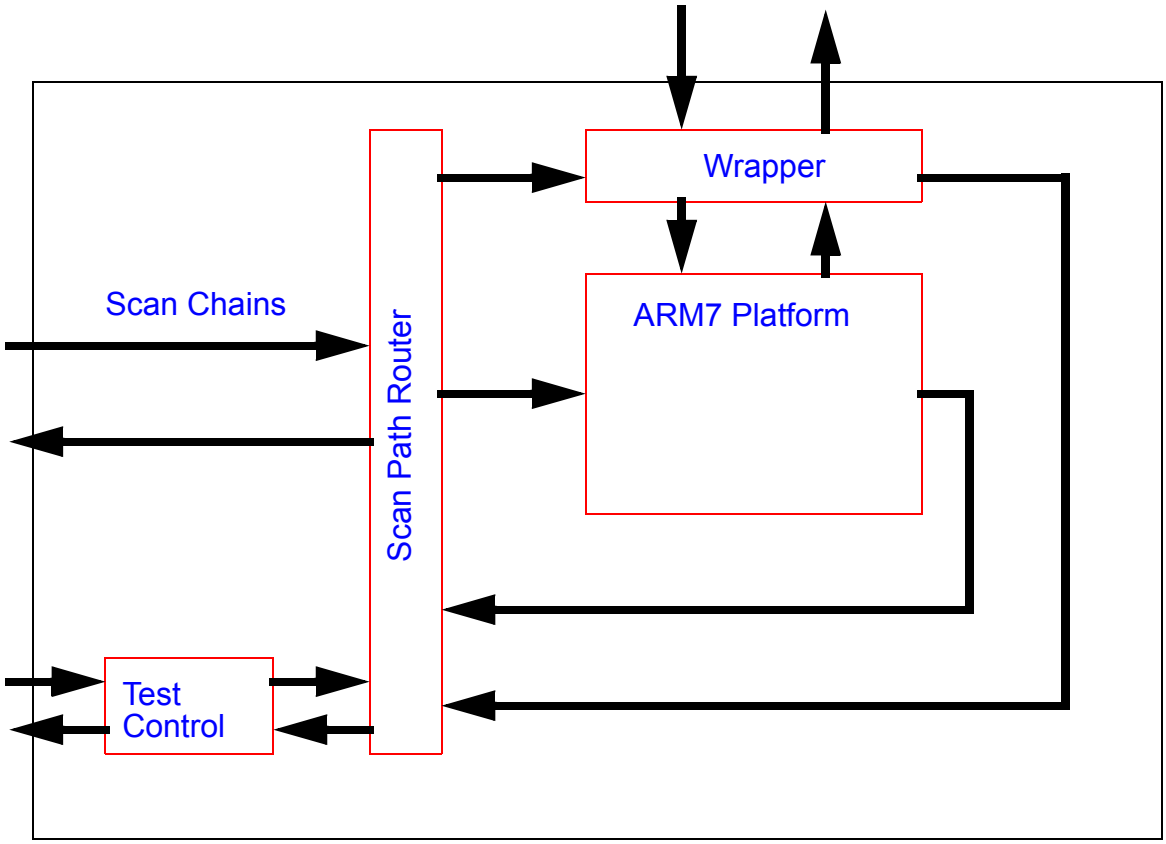


Figure B-28. ARM7 Platform

B.8 Analog and Mixed-Signal DFT

B.8.1 Introduction

Various DFT methodologies, such as scan, built-in self test (BIST), and JTAG, have been widely used and become standard in digital IC chips. The well defined DFT architectures and various ATPG tools are the driving forces in modern test technology. System-on-chip (SOC) technology is widely used in different chip design and manufacturing, and has been posing new challenges in test design. In such chips, analog and mixed-signal are becoming common. This is especially true for most of the communications and electronic control chips. Traditionally, testability is a low priority and the test development is a manual process for analog and mixed-signal ICs. DFT methodology is not well defined for analog / mixed-signal ICs, but recently some new development and process has been made.

While analog/ mixed-signal DFT is gaining more and more attentions and importance due to its a vital part of a SOC chip, the new development and progress is inevitable. This include in different areas, such the advent of certain BIST solutions on mixed-signal ICs and the acceptance of the IEEE 1149.4 standard for mixed-signal test bus. However, our goal and test design strategy is not chasing the very new methodology that is still in development. but rather to use current proven best available DFT methodologies to implement in our chip. This will reduce test time and test cost, and increase test coverage in the production process with less product development time, and meet project schedule. So our approach and focus will be on utilizing full scan, BIST, IDDQ, JTAG, and functional test wherever possible in the Neptune project.

The analog modules and / or sections or blocks include, but not limited to, the followings:

- power analog section/ modules: regulators, level shifters, OpAmps.
- precision analog modules: delta-sigma modulators, ADCs, and DACs.
- oscillating analog modules: VCOs, PLLs, clock circuits.
- I/O pads, etc.

B.8.2 Analog and Mixed-signal Modules

The following is a list of the Analog and Mixed-signal modules in Neptune chip including the Analog to Digital Interface (A2DIGL), which is a digital module.

Table 1-1 shows the test features currently implemented or planned in the Mixed-signal / Analog and related modules. As the chip and module definitions and designs progress, further changes and modifications on test features may be made. Therefore, this document will be updated accordingly.

Table B-6. Analog / Mixed-signal and Related Modules

Module Name	Block Name	Scantest	Special Testmode	Notes
Receiver Coprocessor	RxCPROC	yes	yes	
Receiver Saturation Detection & SDM Dither Generation	RxSDG	yes	yes	
Transmitter	TX	yes	yes	
Fast digital DC adapt	DCADAPT	yes	yes	
Rx/Tx synthesizer	TRSYNT	yes	yes	
Voice Codec	VOCOD	yes	yes	
General Purpose ADC	GPADC	yes	yes	
Power Amplifier Control	PAC	yes	yes	
Tuning Circuit	TUNEC	yes	yes	
Regulators	REGUL	NA	yes	4 regulators
Receiver Analog Front End	RxAFE	NA	yes	
Reference PLL	REFPLL	NA	yes	
Clock Amplifier	CAMP	NA	yes	
Clock Monitor	CMON	NA	yes	
I/O Pads and Sinewave Generator control	PADIO TOSW	NA NA	no no	
Analog Test Interface	ANATEST	yes	no?	
Analog to Digital Interface	A2DIGL	yes	yes	digital control

NOTE:

scan test: mean full scan here.

special testmode: testmode for special functionality and/or signals besides scan testmode.

NA: not applicable or not planed

No scan test signals from analog and mixed-signal modules will be tested through the analog test module, ANATEST. A block diagram is shown in the ANATEST Chapter. For more information, refer to the Analog Test Module specification document.

NOTE:

The output pads pad0 and pad1 of Anatest have a output capacitance of 25pf. An input buffer is needed on the test board interface to test the signals from these two pads.

B.8.3 Analog Modules Requirements

In a pure analog module, a test mode should be defined and implemented so that the critical signals can be controlled and observed through the Analog Test Module. A low power sleep mode and / or a bypass mode is required for scan and IDDQ test.

It has been a common practice to provide a separate power supply for analog modules. This is achieved by either design consideration or functional requirements in a chip. On-chip PLLs, VCOs and other clock circuits should have a mode that disables or disconnects them from the rest of the chip. This is especially important to the IDDQ testing oscillators in a circuit cause increased quiescent current.

In an analog module, there is no scan involved, so special testmodes / signals may be required. There should be a low power mode. The low power mode should be enabled during IDDQ testing. All test signal outputs need to be routed to the analog test module ANATEST.

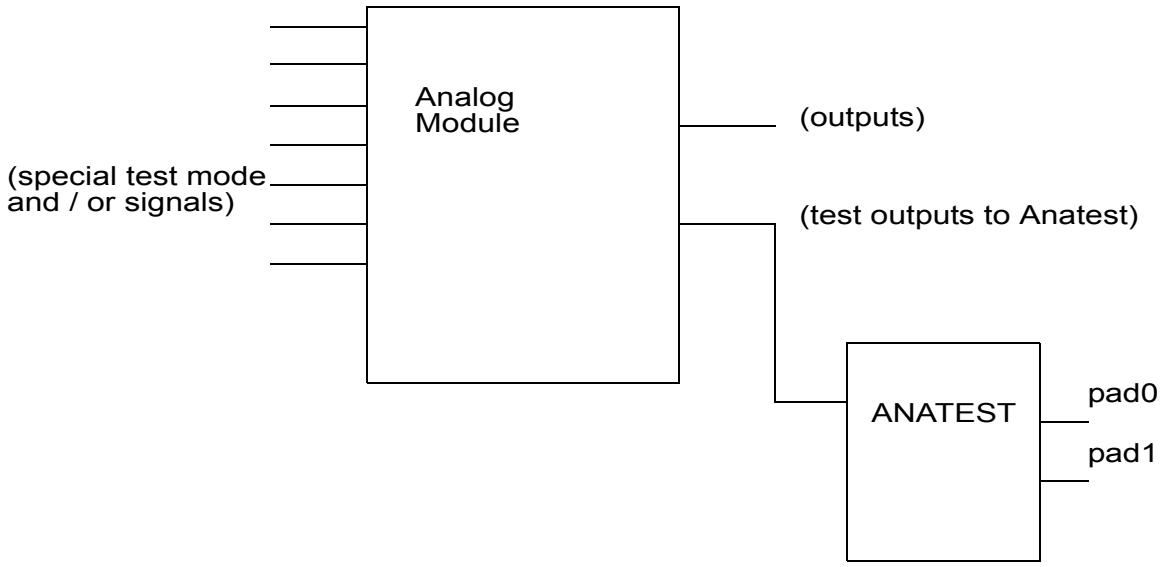


Figure B-29. Analog Module Test Through Anatest

B.8.4 Mixed-Signal Modules Requirements

In a mixed-signal module, the analog section should adopt the same considerations and requirements as described in the previous section of this document for analog modules.

In the digital section of the mixed-signal module, it is recommended the scan test is implemented, where possible. BIST should be implement in memory sections.

A low power mode or a bypass mode should be implemented for chip IDDQ testing and control at the chip level.

If scan and BIST test techniques are not feasible, a well defined functional test mode should be implemented to test the critical signals and registers. Controllability and observability of those signals is required to ensure a high test coverage. This is extremely important for any debugging activities and failure analysis on the silicon after the design completed.

Each module design team should document the test strategy and implementation in a Test Integration Guide for the module. This includes the scan and BIST test strategy, and functional test mode description.

In a mixed-signal module, full scan test of the digital section is required whenever possible. In addition, special or functional testmodes / signals may be required. All test signal outputs need to be routed to the analog test module, ANATEST, to get tested. A low power, IDDQ test mode, is needed for IDDQ test. Refer to the scan signals in the overview section for specific signal naming conventions and usage for digital sections.

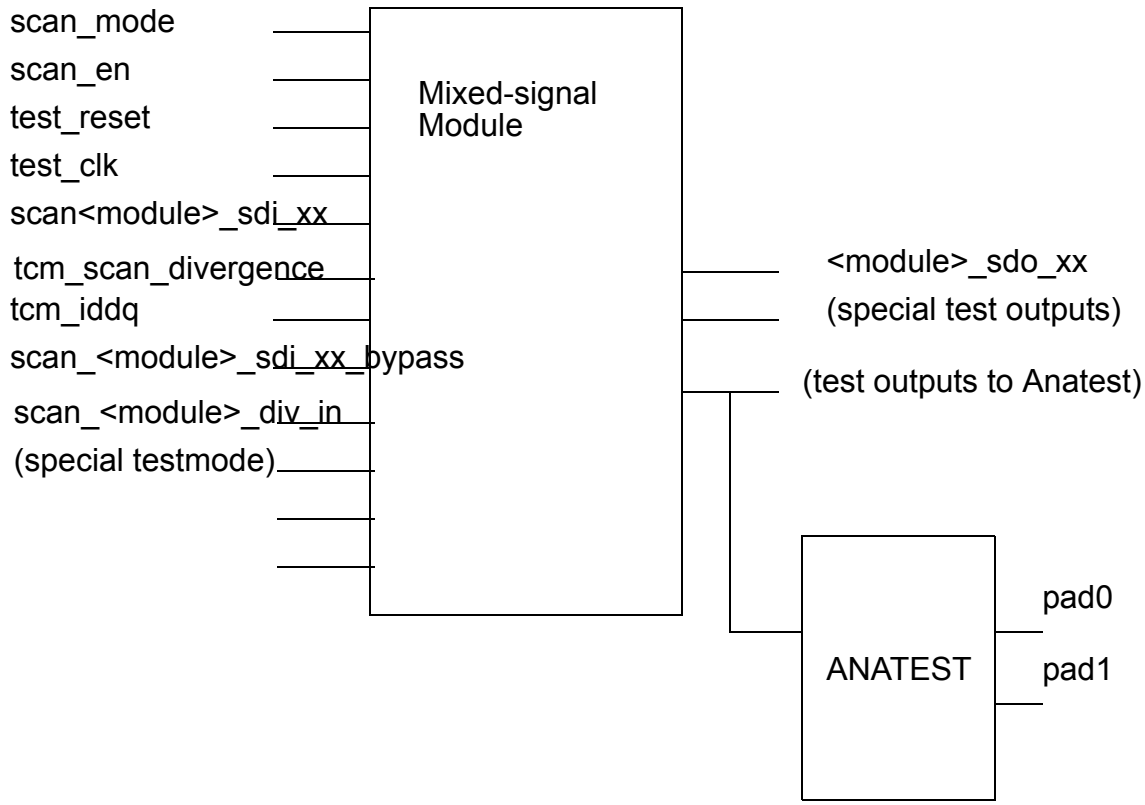


Figure B-30. Mixed-Signal Module Test in General

B.8.5 I/O Pads Requirements

I/O pads generally have internal pullups and pulldowns which normally draw a quiescent current, but can be put in zero current mode using a separate pin. These pins all pullups and pulldowns turned off when the IDDQ measurement is performed. During an IDDQ testing, one should be able physically disconnect or tristate(3-state) device output pins. The design of I/O pads have critical affect to the success of IDDQ test. The followings are the basic requirements to I/O pads:

- Put IO pads on a separate power rail if possible.
- If IO pads and core logic share the same power rail, use IO pads that have controllable pullups rather than pads that have passive pullups: so that the pullups can be gated-out during IDDQ testing.

- Slew control for IO pins must be disabled or IO pins must be put on a separate rail. If IO pads and core logic share the same power rail, all dc path from power to ground, like slew control must be disabled during IDDQ testing.

There are two strategies to achieve this, the first one is preferred:

1. Use controllable pullups and pulldowns so that they can be gated-out during IDDQ testing.
2. Drive pads so that pullups and pulldowns not active (i.e. drive a pad with pullup VDD). Pads that have both pullups and pulldowns should be driven to VDD (VSS if measuring ISSD) by the testbench.

B.9 Introduction[1]

The purpose of this chapter is to describe the BIST requirements and interface for Neptune. The objective is to have one BIST engine for RAM and another for ROM. The BIST engine will be flexible and utilize generics in RTL to make the design reconfigurable for all memory sizes in Neptune.

It maybe enhanced in the future to add additional features for redundancy correction.

All information in this section should be considered PRELIMINARY. This section will be improved in the next version of this specification. AWDC is working with ASP to define the requirements for this section. ASP will then generate a design specification. ASP will design the BIST logic which will be fully sythesizable using the standard cell library.

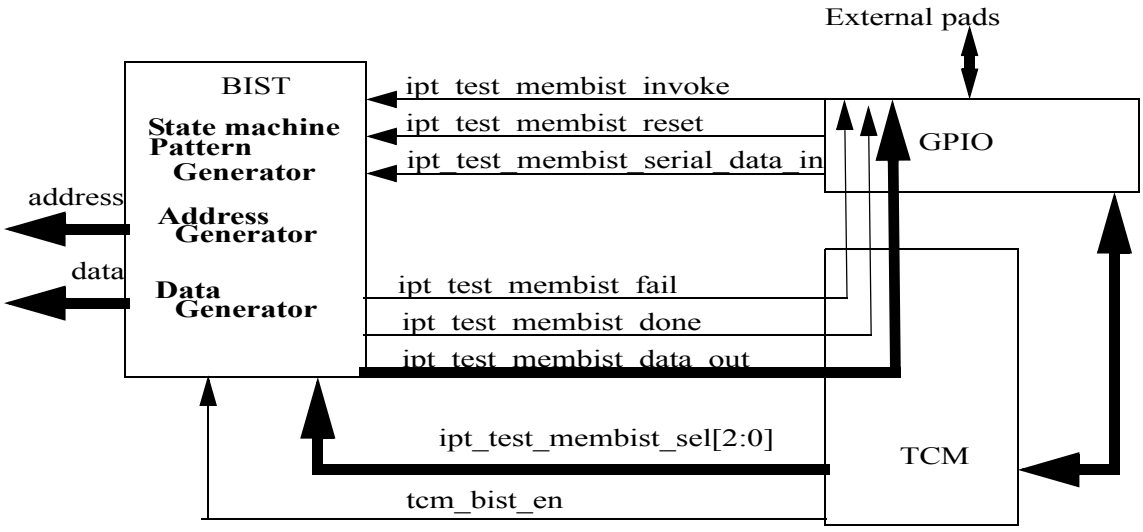


Figure 0-1. RAM BIST Interface

B.9.1 Algorithm

An algorithm must be constructed to cover all faults in the fault model and be easy and efficient to implement in silicon. The algorithm should be memory array architecture independent where changes to the array would not require changing the BIST. March tests are designed to be array structure independent and execute efficiently providing maximum coverage.

1. This specification is a "living" document, which will be updated as necessary during Neptune evolution. It contains hidden text comments which should be turned off for any printing that is to be distributed. If this sentence and the previous one appear, then you have comments turned ON.

The final algorithm has not been selected yet. This will be reflected in the next version of this specification.

A march algorithm will be used. If the memories are word oriented, the march sequence must be extended to cover all IO. The simplest method entails running the entire march algorithm multiple times with different data backgrounds on each execution. The number of data backgrounds is dependent on the width of the IO. For devices where the IO is a power of 2, the formula is: $K = \log_2(m) + 1$, where m is the number of IO and K is the number of patterns required. This is not the most efficient method but is generally easy to implement in a simple BIST controller. A more efficient approach is to modify the march algorithm to cover multiple IO related faults. This may not be feasible without increasing the complexity of the BIST.

B.9.2 Late Write Memory

1. Writes are always buffered. write to the array occurs on the next write operation.
2. Read operations get data from the buffer if the read address matches the last write address otherwise data is from the memory array.

Table B-7. Example RAM Operation with Late Write

Operation	buff	a[0]	a[1]	Comments
pwr up	?	?	?	data state unknown on power up
w0 a[0]	0	?	?	first write data goes in buffer
w1 a[0]	1	0	?	next write pushes previous write into array
r a[0]	1	0	?	read returns 1 which is still in the buffer
w0 a[1]	0	1	?	write to a[0] is pushed into array
w1 a[1]	1	0	0	previous write pushed into array
w1 a[1]	1	1	1	
w0 a[1]	0	1	1	
r a[0]	0	1	1	read 1 stored in array
r a[1]	0	1	1	read 0 stored in buffer

B.9.3 ipt_test_membist_sel[1:0]

Input signals. These signals select the BIST and determine the operating mode for BIST. These signals are static during BIST operation.

0 - Production BIST mode (Fail bit is sticky)

1 - DEBUG mode (Fail bit is not sticky)

2 - DEBUG bit map mode

3 - Data Retention Mode

B.9.4 ipt_test_membist_invoke

Input signal. Assertion starts BIST operation. BIST continues to run while this signal is asserted and the algorithm is not complete. If this signal is deasserted while the algorithm is not complete, then the BIST must pause in the current state where asserting this signal resumes the BIST operation from the pause point and completes the algorithm.

B.9.5 ipt_test_membist_reset

Input signal. Assertion forces the BIST to stop the current operation and place all signals in a known safe state.

B.9.6 ipt_test_membist_fail

Output signal. This signal asserts when a fail comparison occurs. In test mode 1 this signal is sticky so the signal remains asserted until ipt_test_membist_reset is asserted. In test mode 2 this signal is dynamic and asserts when a fail is detected and deasserts for passes.

B.9.7 ipt_test_membist_done

Output signal. This signal asserts when the selected BIST algorithm is complete. In test mode 1 this signal is sticky so the signal remains asserted until ipt_test_membist_reset is asserted.

B.9.8 ipt_test_membist_bitmap_out

Output signal(s). This is at least a single output which is enabled when bitmap mode is selected and the BIST is running. The result of each read comparison is presented on this signal (0 for pass, 1 for fail). For memory with more than one IO, this signal presents the data for each IO by first running the entire BIST sequence, then incrementing to the next IO and repeating the sequence until all IO have been presented.

B.10 Neptune Testing in Additional to Scan

B.10.1 Functional Test Introduction

Neptune has faults will be structurally verified by using internal scan. There are several classes of logic that fall outside of the typical combinational logic that is not scan testable in the Neptune. Currently, the vectors needed have been identified as falling within a few sections: functional speed and non-scanned logic verification tests, scan based IDDQ test, memory test, and analog type tests. Bus master mode will be used to isolate each major block in the design to allow functional tests to be performed without interaction with other blocks. Any on chip PLL's or VCO type circuits will need to be address individually.

B.10.1.1 JTAG

The JTAG logic will need to be verified for functionality (compliance to the IEEE 1149.1 standard) and for manufacturing correctness. Since this test logic is not inherently scannable it must be verified using functional test vectors applied through the JTAG Test Access Port (TAP) and parallel chip inputs (to ensure proper connection of the Boundary Scan Ring) **for DSP**.

It maybe possible to generate such test vectors via a internally generated program from the SSDT group called Jtool. Refer to "Jtag ATPG Software" documentation dated 5/95 version 1.0.

B.10.1.2 Scan Controller (Test Control Module)

The Scan Controller logic will need to be verified for functionality and manufacturing correctness before conducting Scan Testing. Since the scan controller is not inherently scannable it must be verified using parallel functional test vectors applied to the scan test interface pins. Also, since the internal scanned blocks can be scan tested without using any parallel vectors applied to the functional pins, the interface between the chip inputs/outputs and needs to be verified in a pseudo-functional test (values must be placed onto the parallel pins and sampled into scan chain and then scanned out). Refer to the "Scan Specification" section for more details.

B.10.1.3 BIST

BIST will be mandatory starting with HiP7. Once a BIST is invoked it will operate on its own with no required input from the tester except the CLK and a few control signals. Upon completion of the BIST, the value of the BIST need to be assessed.

There needs to be some level of debugging capability on the circuitry with is tested by the BIST. Observability is typically harder to achieve on a BIST.

The Memory Arrays, associated comparators and other critical logic will be tested using the one of the following approaches: Direct access or BIST architecture. If the BIST architecture is chosen, a debugging mode MUST also be supplied.

In the direct access method, each array will be reachable directly from the tester. The chip data, address busses and a designated read/write~ pin will be defined for each memory array. This allows the memories to be tested directly by the tester with standard algorithmic test routines such as walking-ones, checkerboards, and marches. These types of vectors are tester code that will cause the tester to apply repeated wavesets that will algorithmically act on a single memory array (at a time). Vectors that act across several memories at once for verification that the data placed in one is not disturbed by time or by placing data in other arrays are defined as Memory Retention Tests.

In the BIST method, a built in self test designed in logic on the chip will be generated to exercise the memory array at speed using a similar algorithmic test routines such as walking-ones, checkerboards, and marches. This approach will be necessary if the designer decides that isolating the memory and routing the data inputs and outputs will effect its functionality. In using a BIST, debugging an actual fail location is difficult. The designer will be expected to supply a debug mode which will give bit mapping capability. The debug mode does not need to run at speed.

A BIST strategy will be defined in the later versions of this document.

B.10.2 Functional Speed and Non-Scanned Logic Verification Vectors

Certain speed paths and functional areas may not be testable using the scan and BIST test architectures. These areas must be verified for functionality, speed, and manufacturing correctness using manually generated parallel functional vectors applied to the chip while it is in functional mode. Ideally these functional tests will need to be fault graded and pass the faults covered to the ATPG tool.

B.10.2.1 Critical Speed Paths

Certain Speed Paths such as bus-to-memory transfers will have to be verified using parallel functional vectors while the chip is in functional mode, since these paths do not exist while the chip is in the scan mode. It will be the system designers responsibility to identify critical paths and interface with test design to assure proper methods of testing are in place.

B.10.2.2 Bus Specification

Patterns to verify the various bus specifications as to timing, speed, and functionality will have to be applied while the chip is in functional mode. Such functions as speed operation, ability to tri-state, and voltage and current parametrics will have to be accomplished.

B.10.2.3 Non-Scanned Logic

Any non-scanned combinational or sequential logic that is not tested by scan must be tested with functional vectors while the chip is in functional mode. There should be no logic of this type on the Neptune with the exception of the Onyx core.

B.10.3 IDDQ Scan Vectors

IDDQ tests - The ATPG tool which is being used for the Neptune project does support the IDDQ fault model. It has been proven that IDDQ testing is a reliable method of early detection of many different kinds of faults. It does require some design consideration to achieve good IDDQ scan patterns. Refer to the “Scan Design Guidelines” for more information.

IDDQ test will be applied in characterization to determine if this is a feasible means of determining faults in the Neptune.

B.10.4 Analog Test Vectors

The Neptune will be primarily digital with a few specific analog type blocks. These blocks will need to be controllable and observable via test modes which mux in or out the appropriate signals to the package pins to adequately tests these blocks.

These type of blocks must also be controlled during scan testing. Low power mode should be used for all of these type of block while scan testing is performed. Any signal that interfaces to the or from the an analog section should be controlled during scan testing.

B.11 Failure Analysis

Several challenges arise when trying to do a comprehensive failure analysis of a system on a chip (SOC) such as Neptune. One of the strongest tools for failure analysis for a chip this complex is a full scan methodology. However, full scan by itself is not sufficient to allow complete diagnosis of all possible failures. Also if a defect is within the scan control logic or in the scan path, scan will be useless in the failure analysis effort.

Several modules are required to be functional to get into scan mode operation. The TCM, CCM, WDOG, and GPIO are required to get Neptune into scan test modes. If any of these modules have a functional issue, using scan based testing could be useless in failure analysis. Scan divergence mode will be utilized to solve this issue. Additional care is also being taken because Neptune is a process driver for HIP7.

B.11.1 Full Scan

As stated in the opening paragraph, a full scan methodology is the foundation for other methods for a comprehensive FA approach. Scan can be used to identify and help pin-point failing logic. Quick and easy analysis can tell what module and section in that module has the failure. A more detailed analysis can help pin-point the defect to a small set of logic gates.

B.11.2 IDDQ

IDDQ testing is the measurement of a current when all clocks are off. This method helps identify if a node or nodes are incorrectly pulling current because of a defect. Ideally cmos technology will have very low quiescent current if all buses, I/O, memory, and analog sections are off and the chip is in sleep mode. Ideally the only current during IDDQ testing will be in the leakage of normal transistor. However, as more and more transistors are in a design, the total leakage can be significant.

IDDQ vectors can be generated using a selection of functional vectors or allowing an ATPG tool to generate scan based IDDQ vectors. Each method is valid is performed correctly.

Failure analysis can utilize IDDQ vectors with a liquid crystal immision type test to identify sections of the chip which are pulling excessive current. This type of technique can be a very powerful tool in the FA process.

Today, Neptune needs additional test modes to support IDDQ testing fully. This will be a goal in the future. Hopefully some success will be found using limited scan based IDDQ based testing today.

B.11.3 MEMORY BIST with Debugging Capability or at Full Speed Access

The ability to run a memory array at its full speed and have bit mapping capability is essential to successfully analyzing a failing memory. There are two methods that can allow this to happen. Memory Built in Self Test (BIST) with debugging capability or at speed direct access of the array(s).

Bit mapping of the failure to the physical topological location of the defect is a must.

While at speed direct access of the memory is much more straight forward to analyze, multiplexing all of the pins out of the chip and running at speed usually is not possible. In Neptune, the gpio pins are utilized in most test modes. The gpio pins may not be able to run at the memory speed. Also, area impacts of muxing these signals out and routing would be significant.

Memory BIST is another possible solution to obtaining FA data on arrays. It can be run at speed. It has the advantage of having a very simple interface. It only needs a clock signal to run at speed. Since a very few pins have to get out of the chip for analysis, routing overhead is minimal. However, analysis will be more difficult and not as straight forward as the direct access approach.

A memory BIST must be carefully designed to have the proper debugging capability built in. Usually if a BIST fails, the data is internally registered and the failing data and address are serially shifted out of the chip at a slow rate. Some post-processing of the data will have to be done to bit map the entire array. This can make analysis of the data to determine if a row or a column failure has occurred tedious.

B.11.4 Scan Divergence

Sometimes called latch divergency or “Scan on the FLY”, scan divergency is a powerful tool in analyzing failures. Scan divergency is the ability stop a functional test when a failure is observed, stop all clocks, enter scan mode and shift out the data in all registers. This is effectively giving the FA engineer the untimate logic analyzer for the part.

Several challenges must be overcome to successfully implement scan divergence on Neptune or a sub-section of Neptune.

B.11.4.1 Entering Scan Mode Operation

Today, a complicated functional sequence must be used to enter scan mode. System reset, bus master mode, gpio configuration, and special commands into the TCM must be used to correctly enter scan mode operation.

To accomplish scan divergence, significant design changes must happen within the TCM and gpio modules. This requires the addition of two test pins. These pins have been requested. Final pin configuration has not yet been given. These pins are mandatory to support scan divergence and burn-in modes. In the future, these pins will be utilized for flash testing on the next version of Neptune.

B.11.4.2 Reconfiguring Scan Chains

Since all the data for all scan chains must be shifted out, combining of scan chains would also be required at the chip level. Neptune has 27 scan chains for Neptune.

In scan divergency, the scan chains need to be daisy chained together so all scan chains can be shifted out at the same time utilizing the 1 chain. This requires additional logic within each module to accomplish reconfiguring scan chains specifically for scan divergence.

An example of how modules can be daisy chained together to support both scan divergence and burn-in mode is illustrated in Figure B-31. Exact signal naming can be found in the overview section of this specification.

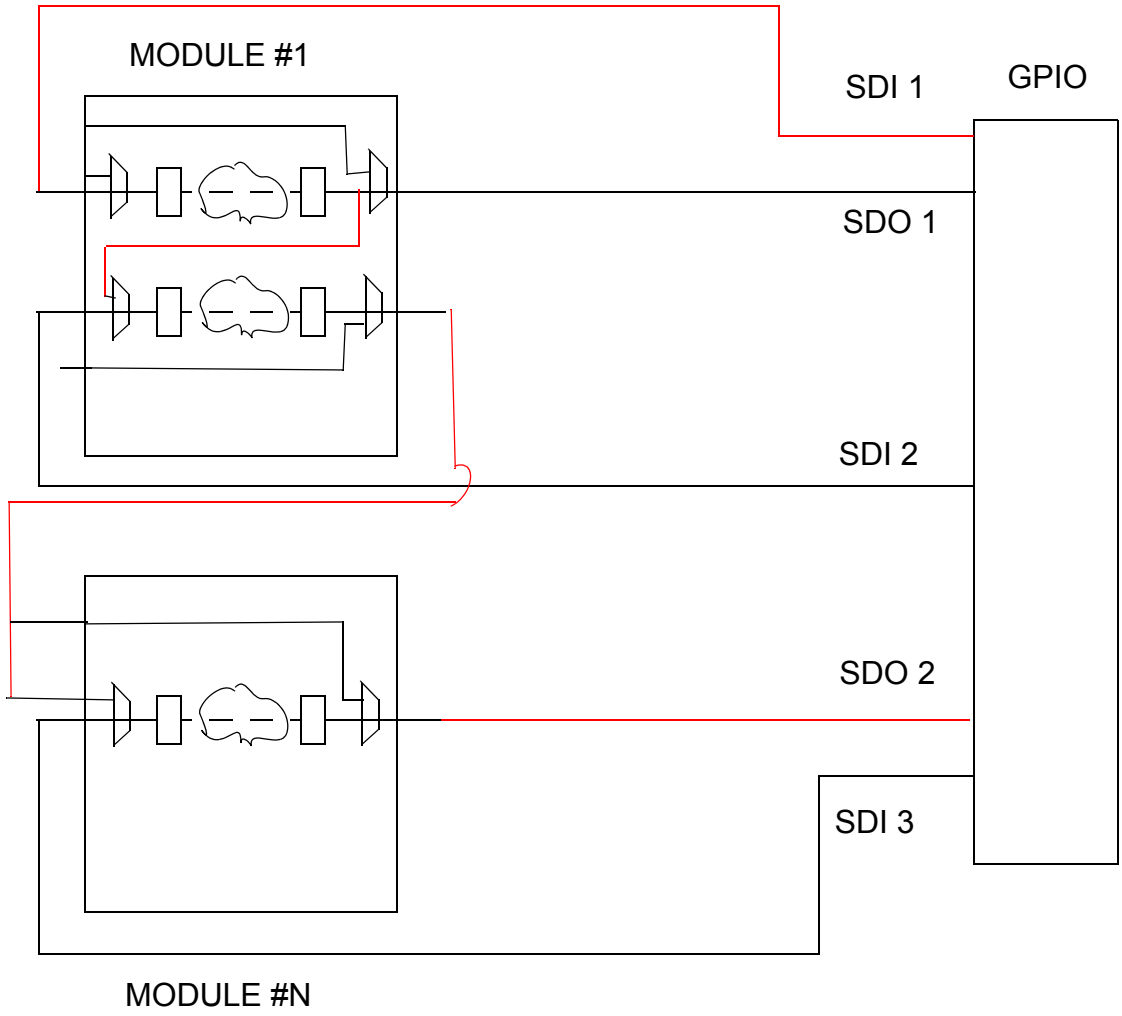


Figure B-31. Example Scan Divergence Configuration

B.11.4.3 Functional Clock Speeds Greater Than the Tester Frequency

The easiest way to clock the device when in scan divergence mode is to bypass the on board PLL(s). This allows the tester to drive the clock(s) and makes the tester synchronous to the internal clock edge. However, several different clock domains exists within the Neptune. The disadvantage of bypassing the clock is in debugging the clock generation circuit. However this is probably an acceptable trade-off.

The tester presently being scheduled for Neptune testing is the Teradyne Catalyst tester. This is a 100M hz tester. There are sections of the Neptune that run above 100 M hz. For these sections, at speed functional testing with scan divergence may not be possible. A possible alternative is to use the on board PLL(s) and use a programmable counter to synchronize the tester and the chip. The counter is 32 bits or more and will be programmable to count down at a certain number of clock cycles and disable the PLL(s).

B.11.4.4 Non-Destructive Memory During Scan Shift

Using scan divergence, it would be necessary to disable writing into any memory location during the scan shift process. It would also be necessary to have another mode to read out the contents of memory after scan divergence testing.

B.11.5 BURNIN Scan Mode

A simple burn-in mode will be supported. This mode will configure all scan chains into one long chain which will be exercised during the burn-in process.

B.11.6 FA Diodes

Some level of Failure Analysis diodes will be required to help identify manufacturing defects using a IDS2000 backside prober. The specification and location of these diodes will be defined by the Neptune Design Methodology or the failure analysis team.

B.11.7 Functional Test Suite

Functional test suites will be necessary for a comprehensive failure analysis to be performed. Using functional tests with scan divergence E-BEAM and/or back side probing of FA diodes, significant debugging can be performed.

Chip level functional tests of the cores must be supplied by the integration team.

B.11.8 Micro-Probing

TBD

B.11.9 E-BEAM Probing

E-BEAM probing from the top side of the die maybe performed on the top layer metal using an IDS5000 type prober. Backside probing using an IDS2000 will be performed utilizing FA diodes.

B.11.10 Test Point Insertion

Test Point insertion utilizing unused areas to push signal to top layer metal will be done using a script developed in MSIL.

B.12 Neptune Scan Guidelines and Rules

B.12.1 General Comments about Rules and Guidelines

Peripherals Designers will have to follow the rules that we have described in this chapter to make the scan test possible and efficient. All exceptions to any of these rules or guidelines will have to be approved by the DFT team.

It is not important to have scan FF inserted after synthesis and before scan insertion. The scan insertion script can handle designs using or not using the compile -scan option.

B.12.2 Clock Rules

For ATPG purpose, one of the major clock rule is the C1 rule. All the clocks must be totally controllable. So, gated clocks are particularly very sensitive to that rule.

PMBIF takes care of the clock rules. But if we don't reroute an asynchronous gated clock to the PMBIF we have to be aware of these rules.

When all the clocks are at their off-state, all clock inputs of scan memory elements must be at their inactive or initial state of a capturing transition.

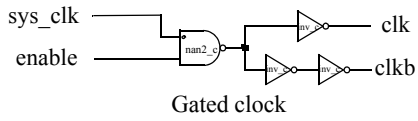


Figure B-32. Gated Clock

Because we don't know anything about the initial state of "enable" signal and also about his behavior during scan, clk and clkb cannot be controlled directly by sys_clk. A solution to avoid this problem is to gate the "enable" signal with the scan_mode signal which is always set during scan test.

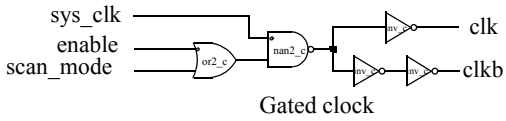


Figure B-33. Gated Clock for Scan

B.12.3 RULE: No Gated Clocks Must Exist During Scan Shifting

Any clock gating circuits shall be bypassed or controlled externally during scan shift process if they exist. There shall be no logic other than inverters and buffers on clock nets during scan shifting. A clock as it relates to scan testing is any signal that affects or controls the state of a flip-flop or latch not in the data path (clk, set, reset, hold, enable).

Gated clocks are allowed as long as the gating signal is disabled during scan shifting and only enabled during the capture cycle. This can be accomplished by using the scan enable signal in the gating logic. **Figure B-32** shows scan enable NOT logically AND'd with the functional clk gate enable as an active HI enable signal to the gated clock circuit.

It is also acceptable if the gating signal of the gated clock circuit can be directly driven externally. This will allow the ATPG tool to generate the proper signals to enable the clk gating at the proper times.

Gating of the clk during the shift process will result in corrupted data into or out of scannable elements. This will result in significantly lower test coverage or possibly an untestable design.

B.12.4 RULE: Define All Scan Ports

In rtl, define all signals described in the overview section of this chapter (scan_mode, scan_en, sdi, sdo, test_reset).

B.12.5 RULE: No Internally Generated Clocks Used During Scan Testing

The clocks used for scan elements shall come from an external package pin to be accessed directly by the tester. This rule applies to PLLs, VCOs, or any other structure that generates an internal clock. These structures shall be bypassed while the device is in scan mode. It is recommended that each clock domain generated has separate bypass control.

This rule also applies to ripple counters, prescalers, clock dividers, one-shots, etc. These structures must be disabled during scan testing and run off of the externally supplied scan clock.

B.12.6 GUIDELINE: Multiplexers Instead of 3-Statable Elements

Several design trade-offs need to be considered in determining a multiplexer logical implementation vs. a 3 statable bus structure. A scan implemented design can handle both of these approaches if design for test considerations are factored in at the design phase. In general, multiplexers are safer and easier to design for scan.

Multiplexers specify an operation in a complete and mutually exclusive way, where a 3 state implementation may not. ATPG tools will attempt to exercise all combinations, either creating vectors that cause contention, or turn off the 3 states and reporting a reduced fault coverage.

3 state elements or busses are acceptable if they are designed to be mutually exclusive during scan testing (see Testing 3 state elements). The use of a 3 state elements enable needs to be handled with care when generating “AT-SPEED” timing type tests (see “AT Speed” testing).

B.12.7 RULE: Testing 3 State Elements

The 3 state elements shall be controllable during scan testing. No X-state generators shall exist during scan testing. One and only one driver shall be active at a time. 3 state busses must not have bus contention during scan testing! Busses should be designed to be mutually exclusive when possible if internal to a module.

Significant loss of test coverage will result if busses are not managed during scan testing. Bus contention will result in loss of all scan patterns generated while contention is possible.

B.12.8 RULE: Testing Dynamic Logic

No dynamic logic shall be active while in scan mode testing (Pre-charge, pullups, keepers, etc.).

Gated pullups and pulldowns are allowed if the pullup/pulldown is gated off during the scan test mode.

Loss of coverage of the pullup/pulldown’s effect during scan test mode since the pullup will not even be described for the ATPG tool.

Dynamic logic which is active during scan mode will affect IDDQ measurements (see SCAN IDDQ Guideline).

Pullups/Pulldown keepers should be disabled and signal should be driven during scan shifting.

B.12.9 GUIDELINE: Types of Storage Elements

The most common forms of scan based design are MUX D and LSSD based scan design. The most important factor in selecting the types of storage elements is to avoid mixing different scan methodologies (avoid adding latches to a flop based design and flops to latch based designs).

FLIP FLOP BASED DESIGN - Storage elements should be positive edge triggered D-type flip-flops. The use of the following storage elements should be avoided:

a. Latches (including transparent latches)

b. Negative edge triggered D-type flip-flops

Implementation of any of the listed elements may result in the loss of test coverage and increased complexity of the test vectors. Additional logic will be required to allow proper shifting of the scan chain.

a. Latches can be implemented to be transparent while performing scan based tests or be substituted as a scan latch during scan insertion. Care must be taken with transparent latches to avoid combinational feedback paths while the latch is transparent. A violation of this rule must be carefully documented. Also, the path must meet timing while in transparent mode. Using scan latches will result in supplying a scan order file prior to scan insertion.

b. For a full scan design, all flip-flops driven by the same external clock must be sensitive to the same edge of the clock. If an inverter is inserted in the clock signal path to provide the inverted clock signal to opposite-edge triggered flip-flops, these flip-flops must be handled carefully in the scan insertion process. Also, the circuitry in the neighborhood of such cells might have reduced fault coverage. (See One Clock Domain per chain)

B.12.10 RULE: Combinational Feedback Paths

No combinational feedback paths shall exist during scan testing. Combinational feedback paths should not exist in most designs. Only a few design techniques intentionally generate combinational feedback paths. State elements such as latches should not be generated with primitive logic gates but should use a standard latch selected from the standard library.

If a combinational feedback path is essential for functionality of the device, additional logic must be added to break the feedback path while scan testing is being performed. This will result in lower test coverage for the additional logic.

B.12.11 GUIDELINE: Handling Memory Array Elements During Scan Testing

Several options on handling memory arrays might be appropriate depending upon the memory type and configuration. A memory strategy should be planned for scan testing. The safest way to deal with memory arrays is to add a boundary of scannable registers around the periphery of the array. When this is not possible because of size or timing issues, care needs to be taken to have controllability of the read and write functions of the array; preferably externally controlled from a chip level input pin. The IP bus specification defines two memory inhibit signals to use for this purpose.

Scan based testing only addresses possibly testing the ports of the memory array. Traditional scan based testing can not be used to test the internal array structure.

BIST (Built In Self Test) or functional direct access methods must be used to gain access and verify the internal array structures.

To avoid possibly significant reduction in fault coverage for the surrounding circuitry.

Scan based IDDQ testing will be significantly impacted if memory elements are not disabled.

Power Consumption

B.12.12 GUIDELINE: Non-Scannable Analog, Core, and Embedded Memory Test Techniques

All non-scannable block including analog, core, or embedded memory blocks should have the ability to be tested from the pin interface of a chip or BIST'd (Built In Self Test). This requires that inputs to the block are accessible between the internal design input and an input at the top level of the module. Outputs from the block need to be brought up to the top level of the module. This can typically be done using bus master mode on Neptune.

If a design has sections of scan based and non-scannable structures, the interaction of these interfaces during scan testing must be dealt with. The safest way to deal with these non-scan structures during scan testing is to add a boundary of scannable registers around the periphery of the structure. When this is not possible because of size or timing issues, care needs to be taken to understand the impacts and interactions with the scan logic.

All of these non-scan modules must have another form of testing which has been simulated and proven, including a way of accessing the module, controlling it to apply test vectors, and observing the output to determine proper operation. A debugging mode should also be available.

This provides a mechanism to control and monitor the functionality of a non-scannable embedded block.

B.12.13 GUIDELINE: Scan Enable Network

The scan enable network should be treated as a clock in the clock tree insertion phase of the design and shall be laid out in a similar fashion (using a tree type structure, although not as rigorous as a full clock tree).

The amount of care that needs to be used in controlling the buffering of the scan enable signal depends upon the timing requirements and goals of the scan testing speed. (If AC type scan patterns will be applied, careful scan enable buffering may be required to meet the scan enable timing requirements).

B.12.14 RULE: SDI (Scan Data-In) Port Hook-Up During Scan Shifting

The only legal connection to SDI port of a scan element is a single fan in from Q,Q BAR, SDO port of a scan cell, or a buffered connection from another scan element during scan shifting or a primary input.

A mux may be inserted in the scan-chain in some cases where the scan chain is modified to support other test structures such as the BIST scan vector insertion point. Thus this register can now serve a dual purpose in different modes. This mux enable must select the path from a single Q,Q BAR, SDO port of a scan cell, or a buffered connection from another scan element during scan testing. (IE the mux is logically transparent during scan testing.)

Some modules in the design have latches which have been configured to be in transparent mode while during scan testing. This will result in slightly lower test coverage because the true function of the latch is not exercised.

Head registers may be inserted for delay paths testing purposes.

B.12.15 RULE: Control Asynchronous Cells During Scan Testing

No asynchronous cells shall be used by any scan tested sequential logic without being disabled during scan shifting. SE must take priority during scan testing.

Asynchronous set or resets should controlled by test_reset or be driven from an external package pin during scan testing.

If asynchronous state elements are required, these functions must be controlled so that they do not toggle during the shifting of the scan chain. This will ensure that the shift process delivers the vector that the ATPG tool generates (the scan data is not corrupted).

Toggling of a controlling signal to a scan element can cause the shift process of the scan chain to be compromised. This can result in significant test coverage loss or possible untestable module.

B.12.16 GUIDELINE: No Multi-Cycle Paths During Scan Testing

No multiple clocked paths must exist during combinational ATPG execution. All multi-cycle paths must be broken and disabled during scan testing.

Combinational ATPG will not generate sequential patterns sets efficiently.

B.12.17 GUIDELINE: Scan Chain(S) Lengths

100-150 elements per scan chain.

A reasonable number of scan chains should be used and all scan chains should be as close to the same length as possible. (BALANCED)

Execution of scan chains in parallel reduces test time. Factors like available package pins and the manufacturing tester memory should also be considered. This will allow the maximum use of the tester memory.

B.12.18 GUIDELINE: Scan Chain Clocking

Scan clocks should be able to run at or above the functional system clock frequency.

This allows the scan chain to be shifted at fast rate to reduce test times. Also there might be other fault picked up other than traditional stuck-at faults. Additional quality from these type of faults are difficult to quantify.

B.12.19 GUIDELINE: Scan In and Scan Out Pad Data Frequency

Shared functional I/O pads for scan in, scan out, enables, sets, clears, and test mode select when doing scan debug modes at speed should be able to run at or above the functional system clock frequency.

B.12.20 GUIDELINE: Control of 3 State Bus Enables

The enable signal of a 3 state bus during path delay testing needs to be properly timing verified.

RULE: Timing verification of scan operation

All “AT-SPEED” scan tests need to be verified for proper operation.

Appendix C
MAP BGA Substrate (PKGROUTE)

Revision History Table

Revision	Date	Author	Changes
0	12/13/01		Initial Release. From Version 2.4 of Neptune IC Baseband Spec.
0.1	05/07/03		Updated for LTE specification release.

C.1 Neptune Package Substrate Design Priorities

This appendix lists the requirements on the MAP BGA package substrate. These requirements must be followed in any new package to be developed.

This is priority ordered list of the critical signals for the Neptune package. Lower numbers are higher priority.

1. TX_CP, RX_CP and associated supplies (VDD_CP, VSS_CP, CP_BIAS) signals have short traces, isolated by it's supplies from others and outside balls. This is a sensitive signal.
2. VCO1_P, VCO1_N and associated supplies (VSS_PRE, REG_BYP_PRE) have short traces, isolated by it's supplies from others and near outside balls - 1GHz input from radio (algae). This is a noise generator and is susceptible. VCO1_P and VCO1_N have reduced ESD protection and must not be on corner balls or the outside row. We desire one row back from the edge.
   - VCO1_P, VCO1_N and associated supplies (VSS_PRE, REG_BYP_PRE) traces and balls shall be surrounded on top and bottom metal with a metal trace as wide as possible connected to VSS_PRE.
3. RX_I, RX_IX, RX_Q, RX_QX, REF_HI, REF_LO and associated supplies (VSS_TXRX, REG_BYP_TXRX_I, REG_BYP_TXRX_O, VDD_TXRX, VAG_RX) have short traces, isolated by it's supplies from others and outside balls. - analog received signal from radio (algae). These are sensitive signals.
4. RX_I, RX_IX, RX_Q, RX_QX and VAG_RX should be adjacent and routed together as they are differential signals and differential mode noise should be reduced by nearby routing. Within the cluster, keep RX_I with RX_IX and keep RX_Q with RX_QX.
5. PA_DET, PA_REF, and PA_OUT signals have short traces and outside balls.

6. PA_DET and PA_REF are a pseudo differential pair and should have adjacent balls and nearby routing.
7. ADC_N, ADC_P, DAC_N, DAC_P, REF_CODEC_N, REF_CODEC_P and associated supplies (VSS_CODEC, VDD_CODEC, REG_BYP_CODEC, VAG_CODEC, VAGOUT_CODEC) have short traces, isolated by it's supplies from others and outside balls - analog voice signal from microphone & to speaker amps. These are sensitive signals.
8. ADC_N and ADC_P be routed together as they are a differential pair.
9. DAC_N and DAC_P be routed together as they are a differential pair.
10. LVDD and LVSS be routed together, have short traces and outside balls.
11. XTAL and EXTAL and it's associated supplies (VDDA, VSSA) have short traces, isolated by it's supplies from others and outside balls. This is a noise generator.
12. QVDD, QVSS, REG_BYP_CORE, VDD, and VSS pin have short traces and outside balls. This is the power supply for the digital portion of the chip.
13. Keeping EIM signals(A[23:0], D[15:0], R/W*, EB[3:0], CS[5:0], DB_SHOW_RW, DB_STRB, QSTAT[4:0], DB_SZ[1:0], BURSTCLK, OE*, CK0, AVDDx7, AVSSx7) away from the higher priority signals and grouped together. Multiple groups are acceptable, for example, one group for D* and one for A*. A23 and A22 are backup pins, not to be used in the system, so they are low priority.
14. Keypad GPIO. This is JVDD, JVSS, ROWx, COLUMNx, and ADC_DATA routed together and have short traces. These are the quietest digital pins in the system.
15. Other GPIO. These are the supply groups BVDD, CVDD, DVDD, EVDD, and FVDD on the pad ring spreadsheet.
    - Any production package: (currently only the 13x13 design) MOD, MCU_DE*, DSP_DE*, TRST, TMS and TCK should not be outside balls. The farther away the better. This is so that a hacker must depopulate the package to access the signals that give access to control the MCU and DSP.
    - Corner balls should only be used for power pads. The new Charge Device Model (CDM) for ESD testing requires that corner balls be tested more stringently. And our power pads contain extra strong ESD devices.
    - VDDx and VSSx traces, particularly those in higher priority groups, shall have the widest allowable trace.
16. TOUT10, INT4, A23, and A22 are spare digital signals and are therefore lowest priority, as described in above items.

C.2 Other Considerations

For test, analog pins should be on the northwest or southwest corner of the package, assuming pin 1 is in NW corner. We currently are in the Northwest corner.