IMAGEON™ 2250 (W2250)
Technical Reference Manual

Revision 2.0
Technical Reference Manual
P/N: CHS-215W2250-2.0

CONFIDENTIAL MATERIAL

All information contained in this manual is confidential material of ATI Technologies Inc. Unauthorized use or disclosure of the information contained herein is prohibited.

You may be held responsible for any loss or damage suffered by ATI for your unauthorized disclosure hereof, in whole or in part. Please exercise the following precautions:

- Store all hard copies in a secure place when not in use.
- Save all electronic copies on password protected systems.
- Do not reproduce or distribute any portions of this manual in paper or electronic form (except as permitted by ATI).
- Do not post this manual on any LAN or WAN (except as permitted by ATI).

Your protection of the information contained herein may be subject to periodic audit by ATI. This manual is subject to possible recall by ATI.

Table of Contents

Chapter 4: Power Management

Chapter 5: Electrical and Timing Specification

Chapter 6: Programming Interface

Chapter 7: Mechanical Characteristics

This page intentionally left blank.

List of Figures

List of Tables

Chapter 1
Introduction

1.1 About this Manual

This manual is part of a set of reference documents that provide information necessary to design the IMAGEON™ 2250 (W2250) graphics controller into a graphics subsystem.

1.2 ATI Component Part Number Legend

This manual covers IMAGEON™ W2250 part 215W2250AFA11 and will be updated periodically to include the latest component revisions and respective additional/changed specifications.

Figure 1-1 below shows how to read the coded information contained in the branded ATI component part number.

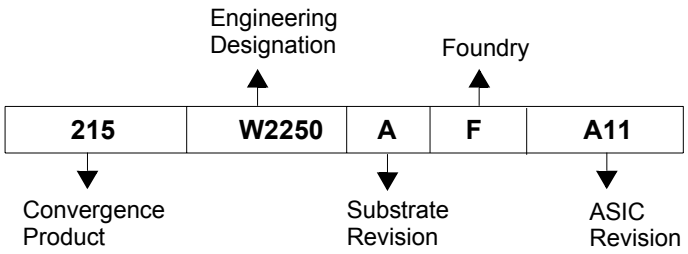

Figure 1-1 IMAGEON™ 2250 **Branded Part Number Coding Scheme**

The following describes each field shown in *Figure 1-1* above:

Convergence Product

All parts are identified by the 215 code.

Marketing Brand Name

These five digits identify the marketing brand name of the component.

Substrate Revision

This digit tracks the substrate revision for the component. The substrate revision is designated in alphabetical order: A = revision 1, B = revision 2, and so forth.

Foundry

This digit identifies the foundry. *Table 1-1* below lists the letters assigned to different foundries.

Table 1-1 Foundry Codes

Foundry Code	Description
C	TSMC Fab3
S	TSMC Fab4
F	TSMC Fab5
G	TSMC Fab6
W	TSMC-WSMC
Q	TSMC-WSMC(8B)

Table 1-1 Foundry Codes (Continued)

Foundry Code	Description
T	TSMC WaferTech
E	SSMC
U	UMC 8B (USC)
D	UMC 8D (USC2)
J	UMC 8E (Utek)

ASIC Revision

The three digits identify the ASIC revision.

1.3 Conventions and Notations

1.3.1 Pin/Signal Names

Mnemonics are used to represent pins. For example, the Gate Clock pin and the Interrupt request pin are represented by GCLK and INTb respectively.

Note: All active-low signal names are identified by the suffix b, e.g., INTb.

1.3.2 Pin Types

The assigned codes for the various pin types based on operational characteristics are listed in *Table 1-2* below.

Table 1-2 Pin Type Code

Code	Pin Type / Operational Characteristics
I	Input
O	Output
I/O	Bi-Directional
Pwr	Power
Gnd	Ground

1.3.3 Numeric Representation

Hexadecimal numbers are appended with “h” (Intel assembly-style notation) whenever there is a risk of ambiguity. Other numbers are assumed to be in decimal notation.

When the same pin name (except the following running integer) is used for pins that have identical functions, e.g. AD0, AD1, etc, a short-hand notation is used to refer to all of them, i.e., AD[22:0] refers to AD0, AD1,..., and AD22.

Note: The above shorthand notation is not to be confused with the notation used to indicate bit occupation in a register. For example, in a PCI device, SUBSYS_VEN_ID[15:0] refers to the Product Type Code field that occupies bit positions 0 through 15 within the 16-bit vendor ID register in PCI configuration space.

1.3.4 Acronyms

Standard acronyms used in the literature are presumed known and will not be explained. Less frequently used or ATI-specific acronyms will have the full definition alongside in parenthesis when they appear for the first time in the document.

Chapter 2
Overview

2.1 Introduction

IMAGEON™ 2250 is a member of ATI's family of media co-processors for handheld devices specifically tailored for mobile phone applications. It integrates an advanced graphics engine, a video decoder as well as a camera sub-system processing engine. Powered by the highly optimized IMAGEON architecture, handheld devices will benefit from ultra low power dissipation and, therefore, a much longer battery life.

IMAGEON 2250 features result in the creation of the ultimate visual experience for the end-user.

2.2 System Block Diagram

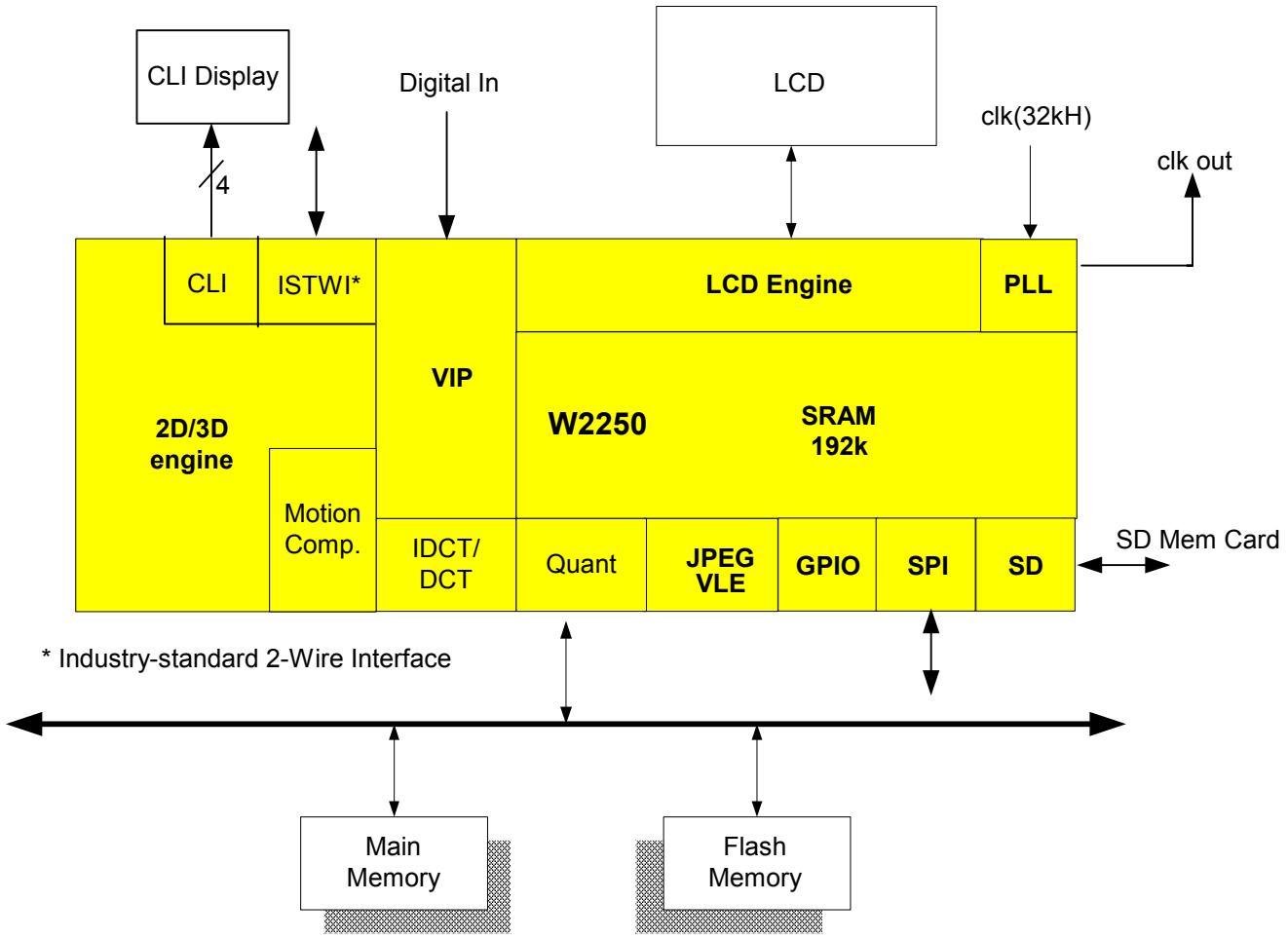

Figure 2-1 W2250 System Block Diagram

2.3 Features List

2.3.1 Advanced 2D Graphics

- BitBLT, ROP3, and ROP4
- DrawLine, Sprites
- Font caching
- Scaling
- Alpha blending
- 90 degree rotation
- Gouraud shading

2.3.2 MPEG/JPEG decoder

- iDCT engine
- Motion Compensation
- Scaling
- Color Space Conversion (CSC)
- CIF and QCIF decode at 30-fps

2.3.3 JPEG ENCODER

- DCT
- Quantization and variable length encode
- Produces CCITT T.81 compliant baseline sequential JPEG scan
- Encode up to 30 fps VGA images

2.3.4 Video Capture Port

- Connect video peripherals (such as a camera) to the handheld device
- CCiR 656, YCrCb 4:2:2 with 8-bit data bus
- Embedded control signals into data stream
- Separate Vsync, Hsync

2.3.5 Host Interface

IMAGEON 2250 interfaces seamlessly to the following CPUs:

- Intel® XScale
- Motorola MX
- TI OMAP
- Other ARM-based System-on-Chips
- A number of proprietary CPUs and baseband chipsets

2.3.6 Bus Configurations Supported

- Direct Addressing Mode: up to 23-bit address bus and 16-bit data bus
- Indirect Addressing Modes:
  - 1-bit address bus and 16-bit data bus
  - 1-bit address bus and 8-bit data bus
  - 1-bit address bus and 1-bit data bus

2.3.7 Integrated Frame Buffer

- Double-buffer support for QCIF+ resolution (220 x 176)

2.3.8 Display Support

- 8-bit monochrome, 12/15/16/18 bit color STN
- 12/15/16/18 bit TFT
- Maximum resolution 320x240 at a color depth of 18-bpp[1]
- Partial display refresh
- Frame modulation
- Panel rotation (90°, 180°, 270°)
- Decimate or scale down 2:1, 4:1 input image up to 1024x768 and preview using overlay

2.3.9 Clock Source

Three options available

- Direct use of an available clock
- Low-frequency crystal
- RC circuit

2.3.10 SD Memory and I/O Controller

- Designed for read-only or read/write cards
- Compatibility with Multi-Media Cards (MMC)
- Transfer support for any byte size
- Up to 10 Mbytes/s read/write rate (using 4 parallel lines)

2.3.11 Operating Voltage

- Range from 1.5V to 3.3V CPU Interface
- 1.2V ~ 1.8V Core Supply
- 1.5V/ 1.8V / 2.5V / 3.3V LCD Interface

2.3.12 Power Management

- Dynamic power down for inactive blocks
- Low power / low voltage process
- Power consumption
  - Suspend < 0.1 mW
  - Display refresh < 1mW
  - Average active mode ~ 10mW

2.3.13 Package

Many package options available

- BGA, COF, COB, Flip-Chip, etc

1. The frame buffer data format is 16-bit, and data extension/dithering techniques are used for 18-bit display support.

2.4 Development Support

- Reference Design Kit
- Diagnostic utilities
- Demonstration software

2.4.1 API Interface

- ATI Core Driver
  - hardware abstraction layer

2.4.2 Supported Operating Systems

- Microsoft® Pocket PC, Pocket PC Phone Edition and SmartPhone
- Windows® CE
- Symbian EPOC
- Java
- Linux (Qt)

Chapter 3
Functional Description

This chapter describes the major subsystems and interfaces of the W2250. To go to a topic of interest, use the following list of linked cross references:

3.1 Display Controller

3.1.1 Display Features

- Maximum resolution (double buffered): 176 x 220
- Maximum refresh rate: 75 Hz
- Portrait mode to rotate image by 90 deg, 180 deg, or 270 deg
- Overlay rotation is supported
- Video can be inverted horizontally (mirror) or vertically (rotate 180 deg)
- Partial display mode for power saving
- Second graphics source (supported through video overlay channel)
- ICON (supported through video overlay channel)
- Low power mode. Only primary graphics mode can be supported. Video mode and portrait mode are not supported due to bandwidth limitation

3.1.2 Frame Buffer Format

- 8 bpp for both mono and color
- 16 bpp (1555)
- 16 bpp (4444)
- 16bpp (565)

3.1.3 Panel Type

- STN mono (single pack 4, pack 8)
- STN color (single pack 4, pack 8, pack 16)
- TFT 333
- TFT 444
- TFT 555
- TFT 666
- Both Generic and HR type of TFT

3.1.4 Display Filter

- Frame modulation on STN producing a maximum of 16 gray level (mono) or 4K color (color)
- Frame modulation on TFT to boost color level by 2/4 times per color component, depending on the frame buffer format and the panel type
- YUV to RGB conversion and brightness control for VIDEO overlay data
- Color keying and masking for video overlay data
- Color to grey scale conversion

3.1.5 Video Overlay

- Supports 4:2:2 and 4:2:0 formats
- Supports QCif 176(h) x 144 (v)
- Maximum size: 320 x 480

3.1.6 Hardware Cursor and Hardware Icon

- 2 bpp, maximum size is 32 x 32
- 00 – color 0, 01 – color 1, 10 – transparent, 11 – inversion
- 1 hardware ICON (16 bpp, maximum size is 240 x 64) (supported through video overlay channel)

Table 3-1 Number of H/W Cursor and Icon versus Display Modes

Operation Mode	Max Number of Active Hardware Cursors	Max Number of Active Hardware Icons	Max Number of Graphics Data Sources	Max Number of Video Data Sources
Graphics and Video	2	0	1	1
Graphics only	2	1	1	0
Full screen Video	2	0	0	1
Multiple Graphics sources	2	0	2	0
Low Power Mode with Reduced Display	2	0	1	0

3.1.7 Panels, Color Data, and Frame Buffer Formats

Table 3-2 Valid Combinations of Panels and Frame Buffer Format

Panel Format	Frame Buffer Format	Output Color Available	Output Data Format
STN Mono	4, 8 bit	FM to 8/16 level gray scale	Packed 4, 8
STN Color	4444, 1555, 565	4444: FM to 2K or 4K color 555, 1555: Same as 4444	Packed 4, 8, 16
TFT 333	4444, 1555, 565	4444: FM to 444 using 2 level 565, 1555: FM to 444/555 using 2/4 level	333
TFT 444	4444, 1555, 565	4444: 444 565, 1555: FM to 555 using 2 level	444
TFT 555	4444, 1555, 565	444: expands to 555 565, 555: 555	555
TFT 666	4444, 1555, 565	Expanding to 666	666

Note: FM = frame modulation

Table 3-3 Color Data And Frame Buffer Mapping

Frame Buffer Format	Byte used in Frame Buffer	Data Assignment
4 bpp (mono)	1	Data(7:4) = Data(3:0)
8 bpp (mono)	1	Data(7:0)
4444	2	Data(15:12) = K, Data(11:8) = R, Data(7:4) = G, Data(3:0) = B
1555	2	Data(15) = K, Data(14:10) = R, Data(9:5) = G, Data(4:0) = B
565	2	Data(15:11) = R, Data(10:5) = G, Data(4:0) = B

3.1.8 Packing Formats

Table 3-4 Mono STN Single Pack 8

Mono STN Single Pack 8			
LCD Shift Clock	**0**	**1**	**2**
oD(17:8)	0	0	0
oD(7)	iR1	iR9	iR17
oD(6)	iR2	iR10	iR18
oD(5)	iR3	iR11	iR19
oD(4)	iR4	iR12	iR20
oD(3)	iR5	iR13	iR21
oD(2)	iR6	iR14	iR22
oD(1)	iR7	iR15	iR23
oD(0)	iR8	iR16	iR24

Table 3-5 Color STN Single Pack 4 and 8 (Format 2)

Color STN Single Pack 4				Color STN Single Pack 8 (Format 2)			
LCD Shift Clock	0	1	2	LCD Shift Clock	0	1	2
oD(17:8)	0	0	0	oD(17:8)			
oD(7)	iR1	iG2	iB3	oD(7)	iR1	iB3	iG6
oD(6)	iG1	iB2	iR4	oD(6)	iG1	iR4	iB6
oD(5)	iB1	iR3	iG4	oD(5)	iB1	iG4	iR7
oD(4)	iR2	iG3	iB4	oD(4)	iR2	iB4	iG7
oD(3)	0	0	0	oD(3)	iG2	iR5	iB7
oD(2)	0	0	0	oD(2)	iB2	iG5	iR8
oD(1)	0	0	0	oD(1)	iR3	iB5	iG8
oD(0)	0	0	0	oD(0)	iG3	iR6	iR8

Table 3-6 Color STN Single Pack 8 (Format 1)

LCD Shift Clock	0	1	2	4	5	6
oD(17:8)	0	0	0	0		
oD(7)	iR1	iG1	iG6	iB6	iB11	iR12
oD(6)	iB1	iR2	iR7	iG7	iG12	iB12
oD(5)	iG2	iB2	iB7	iR8	iR13	iG13
oD(4)	iR3	iG3	iG8	iB8	iB13	iR14
oD(3)	iB3	iR4	iR9	iG9	iG14	iB14
oD(2)	iG4	iB4	iB9	iR10	iR15	iG15
oD(1)	iR5	iG5	iG10	iB10	iB15	iR16
oD(0)	iB5	iR6	iR11	iG11	iG16	iB16

Table 3-7 Color STN Single Pack 16

LCD Shift Clock	0	1	2
oD(17:16)	0	0	0
oD(15)	iR1	iG6	iB11
oD(14)	iB1	iR7	iG12
oD(13)	iG2	iB7	iR13
oD(12)	iR3	iG8	iB13
oD(7)	iB3	iR9	iG14
oD(6)	iG4	iB9	iR15
oD(5)	iR5	iG10	iB15
oD(4)	iB5	iR11	iG16
oD(11)	iG1	iB6	iR12
oD(10)	iR2	iG7	iB12
oD(9)	iB2	iR8	iG13
oD(8)	iG3	iB8	iR14
oD(3)	iR4	iG9	iB14
oD(2)	iB4	iR10	iG15
oD(1)	iG5	iB10	iR16
oD(0)	iR6	iG11	iB16

Note: This pack is similar to STN Pack 8 (Format 1). It just combines two data into one clock.

Table 3-8 TFT Panels

LCD Shift Clock	TFT 333	TFT 444	TFT 555	TFT 666
oD(17)	0	0	0	B(5)
oD(16)	0	0	B(4)	B(4)
oD(15)	0	B(3)	B(3)	B(3)
oD(14)	B(2)	B(2)	B(2)	B(2)
oD(13)	B(1)	B(1)	B(1)	B(1)
oD(12)	B(0)	B(0)	B(0)	B(0)
oD(11)	0	0	0	G(5)
oD(10)	0	0	G(4)	G(4)
oD(9)	0	G(3)	G(3)	G(3)
oD(8)	G(2)	G(2)	G(2)	G(2)
oD(7)	G(1)	G(1)	G(1)	G(1)
oD(6)	G(0)	G(0)	G(0)	G(0)
oD(5)	0	0	0	R(5)
oD(4)	0	0	R(4)	R(4)
oD(3)	0	R(3)	R(3)	R(3)
oD(2)	R(2)	R(2)	R(2)	R(2)
oD(1)	R(1)	R(1)	R(1)	R(1)
oD(0)	R(0)	R(0)	R(0)	R(0)

3.1.9 Timing Diagrams

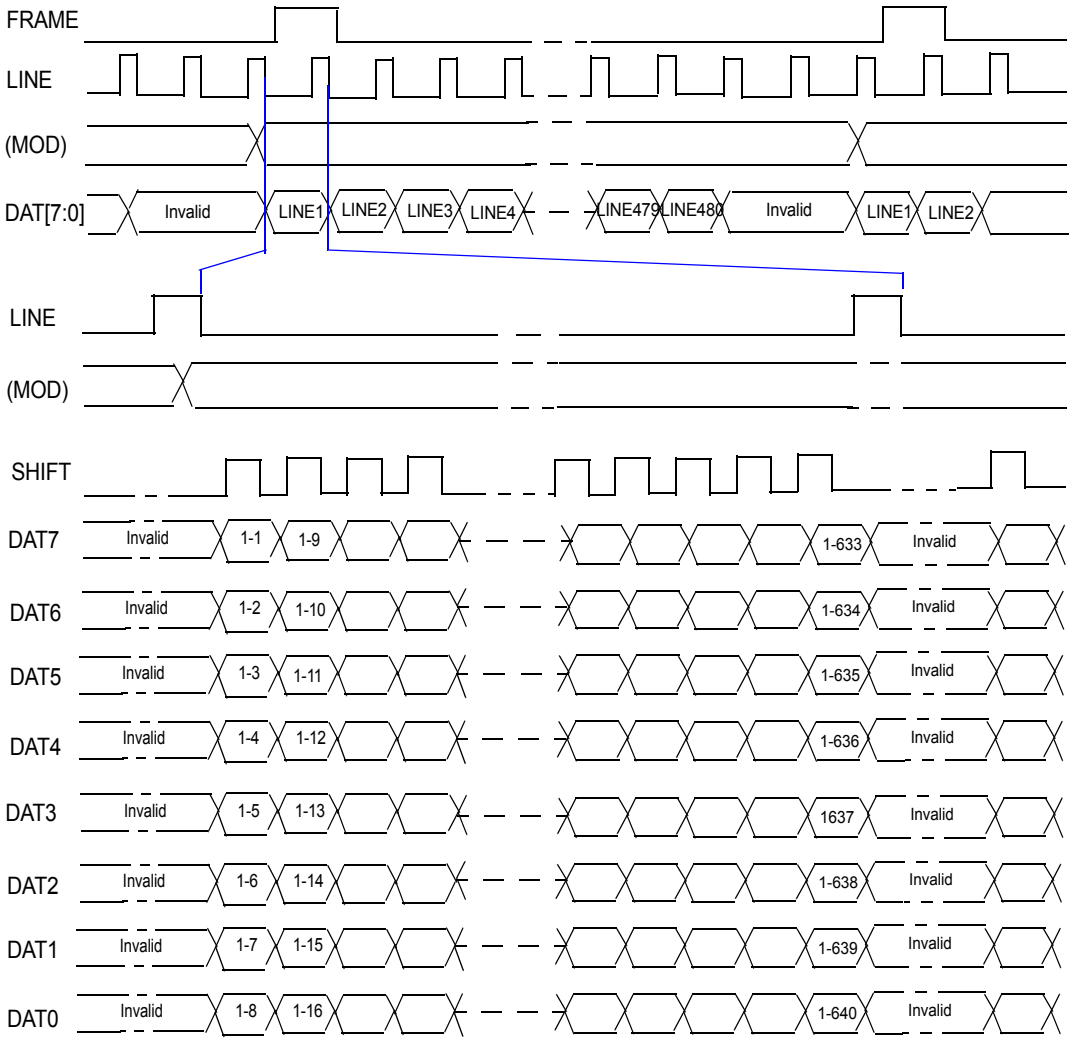

Figure 3-1 STN Mono Pack 8

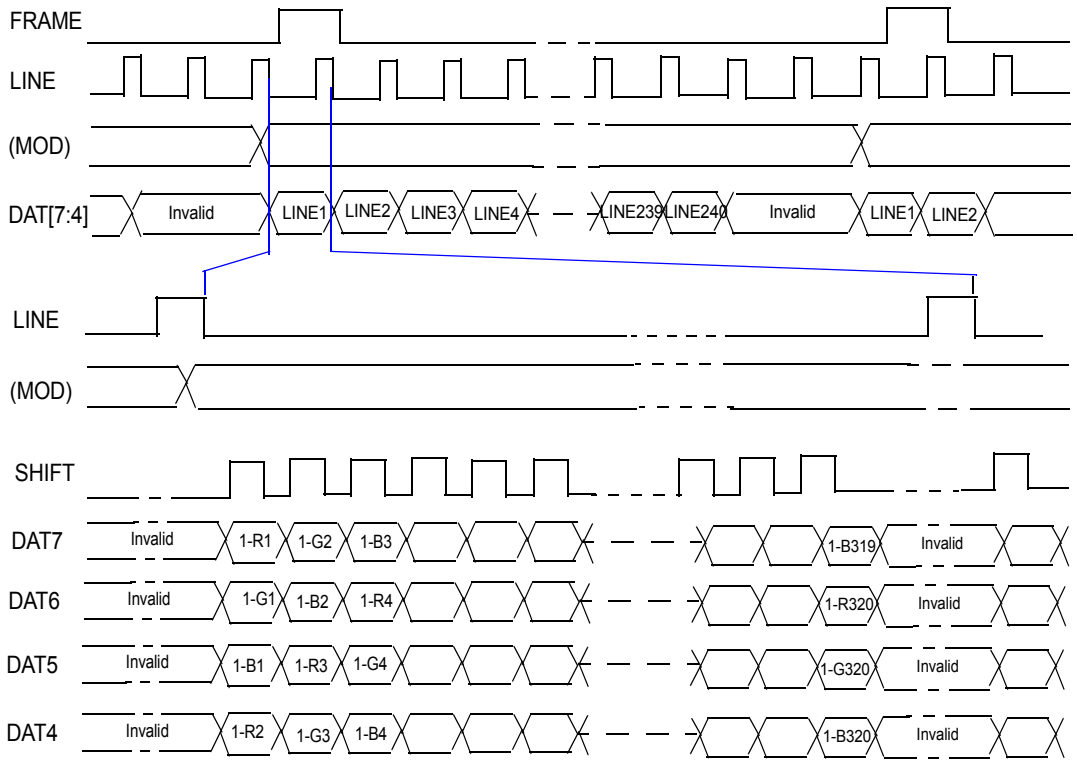

Figure 3-2 STN Color Pack 4

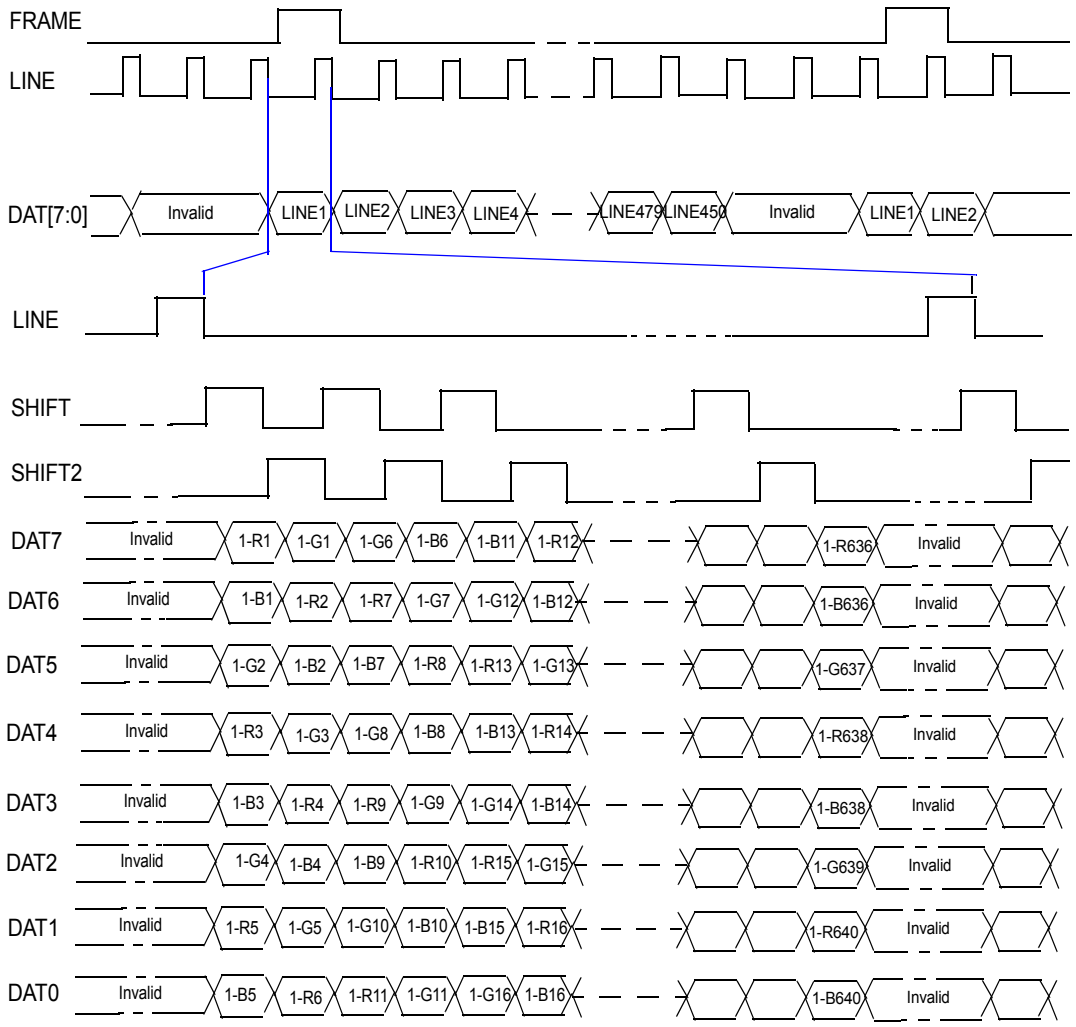

Figure 3-3 STN Color Pack 8 (Format 1)

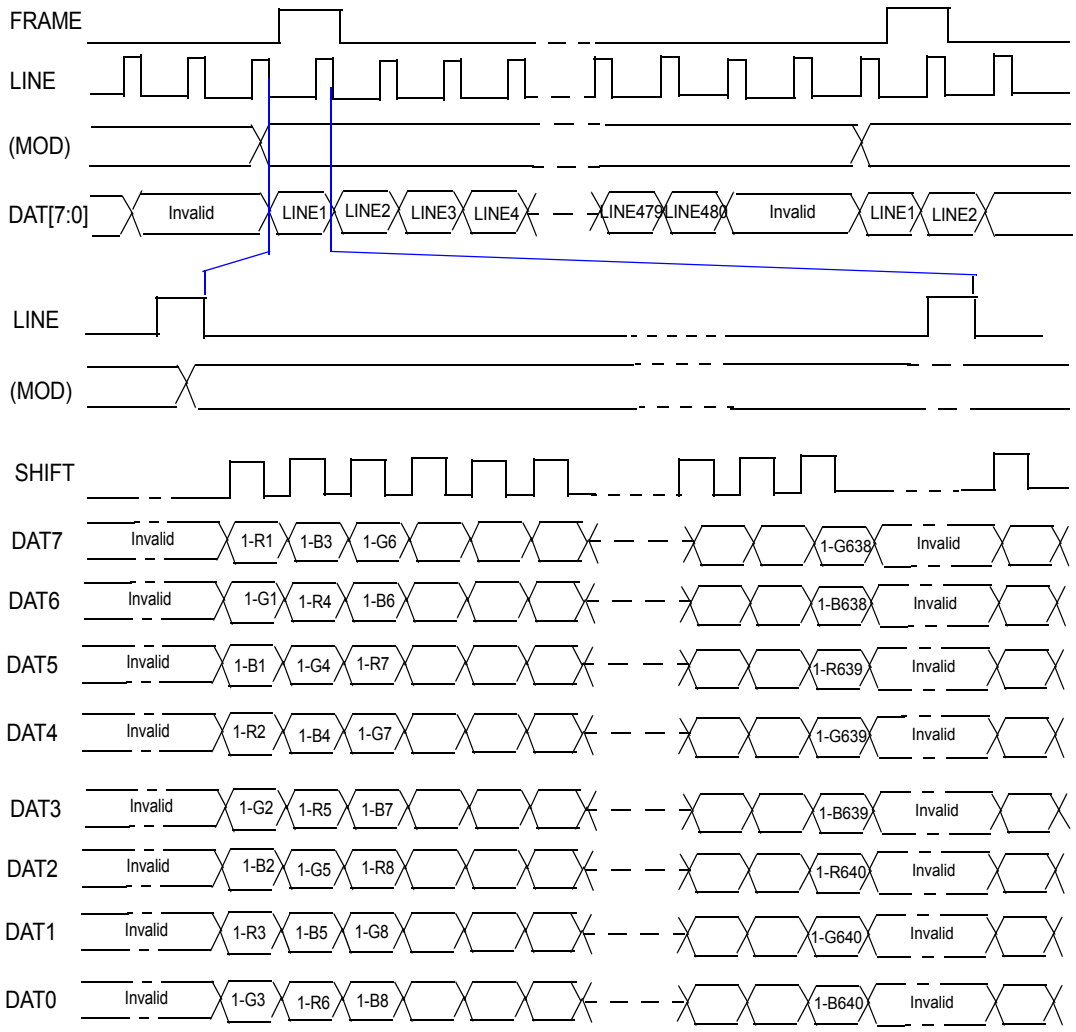

Figure 3-4 STN Color Pack 8 (Format 2)

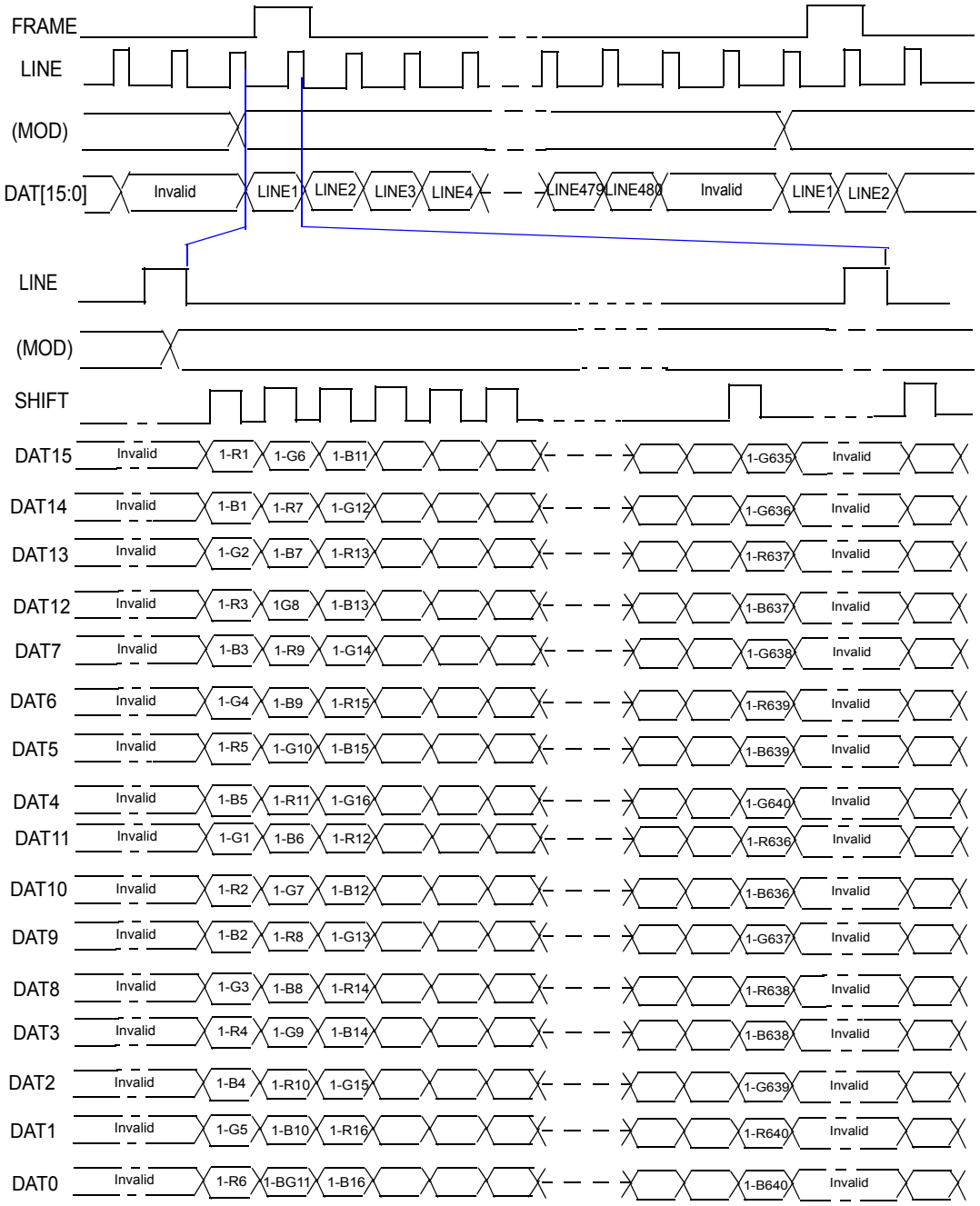

Figure 3-5 STN Color Pack 16

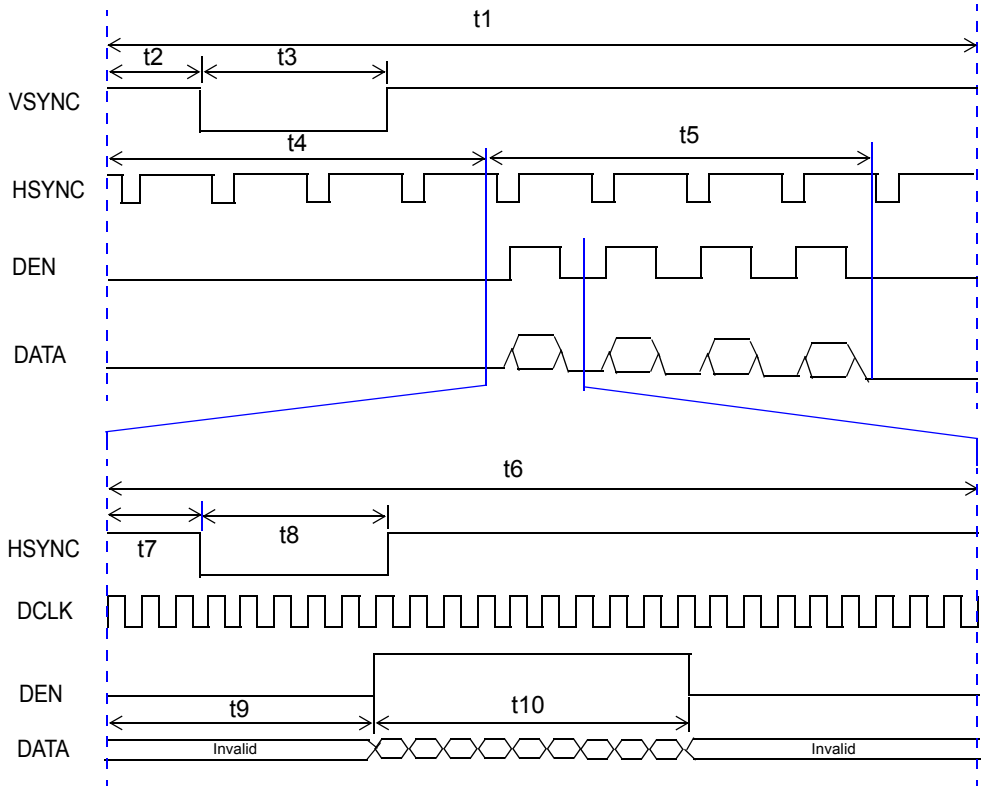

Figure 3-6 Generic TFT Panel Timing

Table 3-9 Generic TFT Panel Timing Parameters

Parameter	Description
t1	total scan line
t2	vsync start
t3	vsync width
t4	valid line start
t5	Total valid line
t6	Total pixel per scan line
t7	hsync start
t8	hsync width
t9	valid data start
t10	valid data width

Notes:

- t1 to t5 are programmable in number of lines
- t6 to t10 are programmable in number of pixel clocks

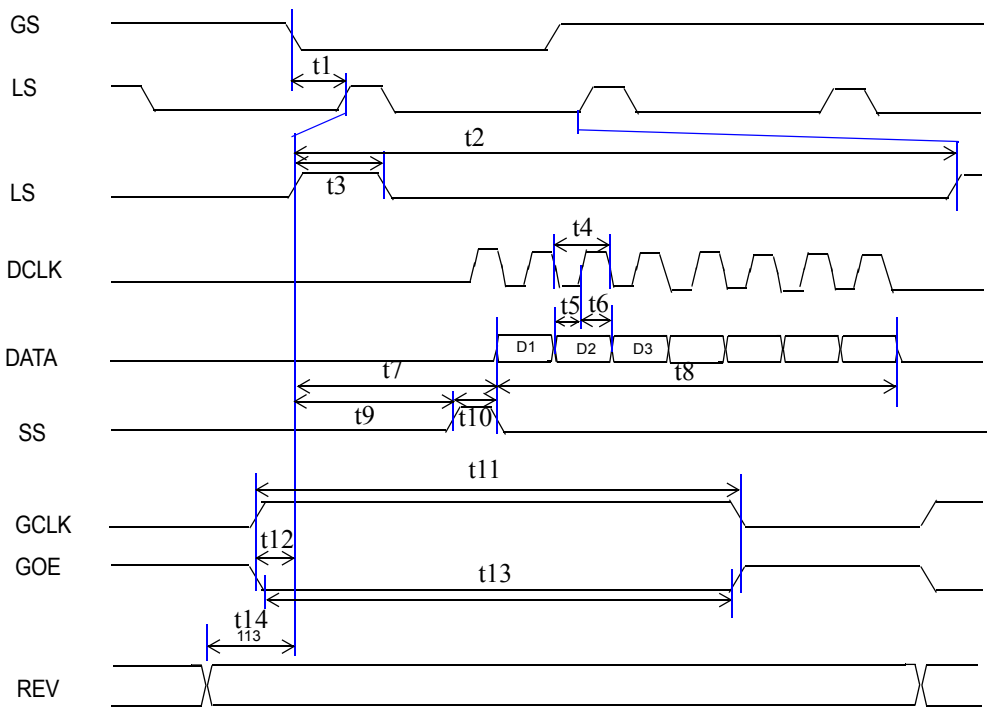

Figure 3-7 Direct LCD Interface

Table 3-10 Direct LCD Interface Timing Parameters

Parameter	Description
t1	latch setup time
t2	latch period
t3	latch width
t4	pixel clock period
t5	data setup time
t6	data hold time
t7	latch to first data
t8	valid data width
t9	latch to scan start
t10	scan start width
t11	gate clock high time
t12	gate output enable to latch
t13	gate output enable low time
t14	Rev to latch signal

Notes:

- t5, t6 are set to around half the pixel clock period.
- All other signals are programmable in number of pixel clocks

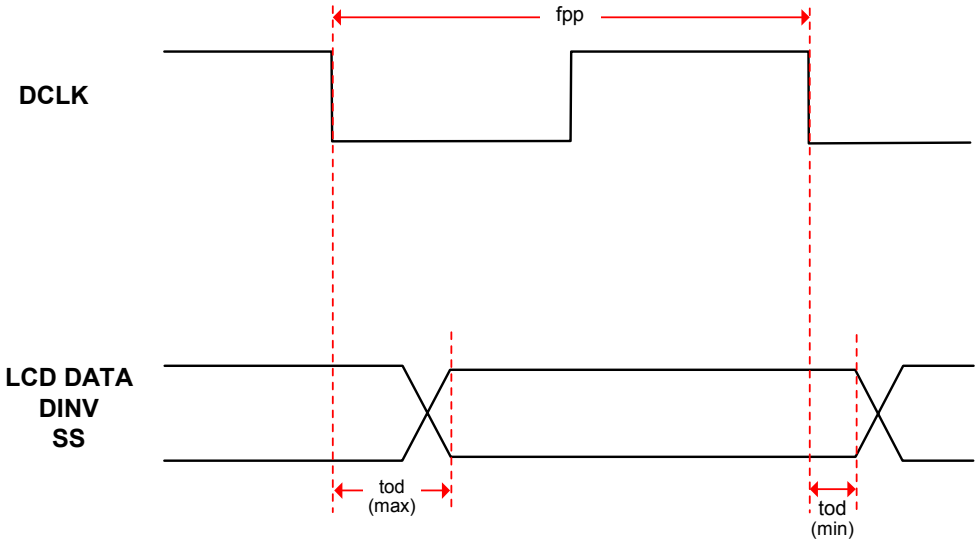

Figure 3-8 LCD Interface

Table 3-11 LCD Interface Timing Parameters

Parameter	Symbol	Minimum	Maximum	Units
Clock Frequency	fpp	0	10	MHz
Output (with respect to DCLK)				
SS	tod	2	4	ns
LCDD	tod	1	4	ns
DINV	tod	1	4	ns

3.1.10 LCD Interface and GPIO Pin Multiplexing

Table 3-12 LCD Interface and GPIO Pin Multiplexing

Pin	Generic TFT Interface	Direct LCD Interface	STN (Mono/Color) Interface
LCDD0 to LCDD5	R0 to R5	R0 to R5	LCD
LCDD6 to LCDD11	G0 to G5	G0 to G5	DATA
LCDD12 to LCDD17	B0 to B5	B0 to B5	
DCLK	data shift clock	data shift dclk	shift clock
SS	data enable (DEN)	Source start (SS)	
LS	horizontal sync	latch enable for source driver (LS)	Line (data latch)
GS	Vertical sync	Gate start signal for gate driver (GS)	Frame
GCLK		Gate clock for gate driver (GCLK)	
GOE		output enable for gate driver (GOE)	
DINV		data inversion bit	
GPIO0			
GPIO1		REV signal (reverse polarization)	
GPIO2			
GPIO5			
GPIO6			
GPIO7			

3.2 Graphics Engine

The W2250 graphics engine is a fixed-function subunit that runs concurrently with the host processor. It incorporates full ROP3 and ROP4 (Pattern/Src/Dst) support and is capable of drawing both rectangle and line draw primitives. A sophisticated pixel datapath allows monochrome to two-color expansion, fast solid color fills, patterned fills, and host-to-screen transfers.

2D Engine Features:

- Support up to 8MB virtual address
- Full ROP3 and ROP4 support (Pattern/Src/Dst)
- Patterns (8 x 8 color/mono brushes, 32 x 1 mono pens for lines)
- Rectangle and line trajectories
- 1/4/8/15/16 bpp support with 16 bits datapath
- Colorcmp on both src/dst channels
- Colorcmp flip
- Interrupt support
- Multimedia event triggers
- Quick engine setup
- A four-function simultaneous source/destination color compare for transparent blits, bit masking, and scissoring
- Stretch blit

2.5D Enhanced 2D Features:

- Alpha blending source and destination
- Alpha can be chosen from source, dst, or independent alpha plane
- Scaling up/down with bilinear filter

3.2.1 Graphics Engine Block Diagram

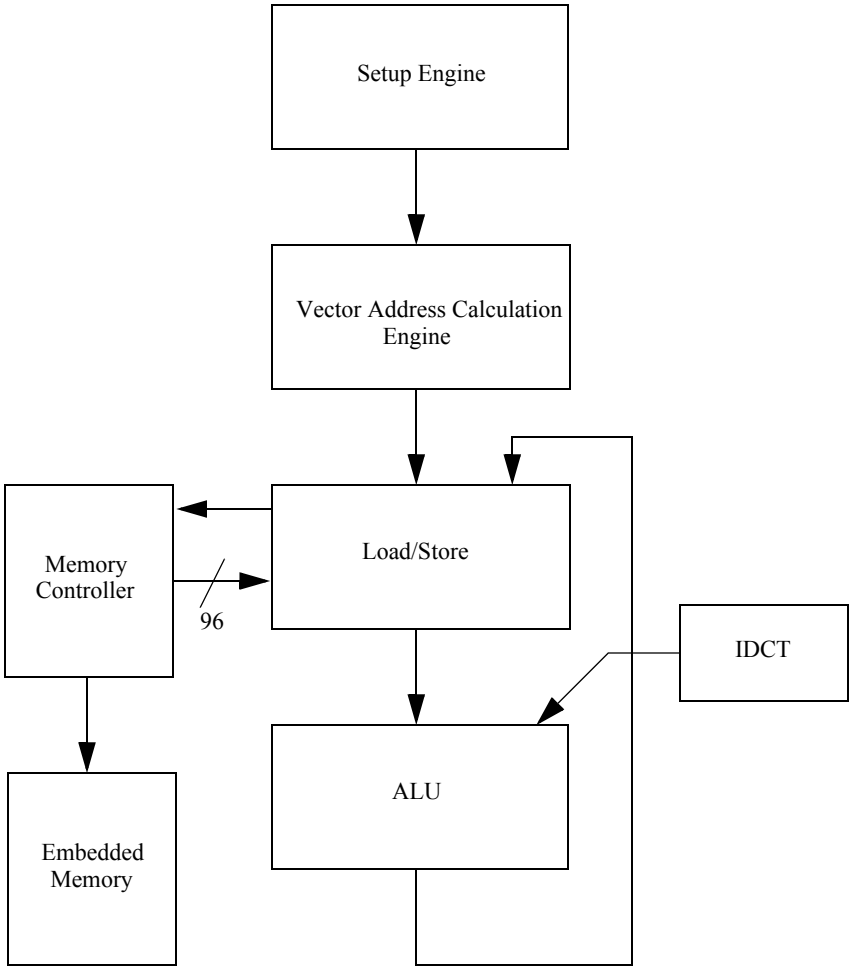

Figure 3-9 Graphics Engine Block Diagram

3.2.2 Graphics Engine Architecture

The Graphics Engine (GE) is a specialized stream/vector processor. The GE programming model is based on the execution operation that is set by an operation code that is part of the render state. The defined operation is applied to all stream elements.

Instruction:

Render State:

setSrcAdd(offset,pitch,w,h)
setDstAdd(offset,pitch,w,h)
setScissorsSrc();
setScissorsDst();
setBpp;
setSrcMono – source data are 1bpp and are used for to selecting one of the scalar register: fg/bg
setBrush
setScalarAlpha

Operations:

2D

FILL(Src)
SRCCPY(Src,Dst)
ROP3/ROP4 (Src, Dst, Brush)

2.5D

add/sub src, dst
alphaBlend(src,dst,ScalarAlpha)
alphaBlend(src,dst,srcAlpha)
addS src, ScalarR //add scalar value to the each element of the //stream
subS src, ScalarR
scale(src,srcXInc, srcYInc, dst) // bilinear filter
motion compensation

Source Stream:

rectangle(xs,ys,w,h);
span (xs,ys) width length of the span w*h, and ignoring pitch

Destination Stream:

rectangle(xd,yd,w,h);
line(xs,ys,xe,ye);

The GE can be divided into two main sections: Setup Engine and Vector Engine. The Setup Engine is responsible for converting the complex input into spans. It is a small scalar processor with several arithmetic units (AUs).

The other blocks are the Vector Address Calculation, Load and Store, and Execution Units. All these stages are pipelined.

Memory width is 96 bits, which allows enough bandwidth for graphic operations.

3.2.3 2.5D Operation

Alpha blending involves combining previously written pixel data with current data to create a level of transparency defined by the earlier calculated alpha component. The available functions for alpha blending are the following:

- Zero
- One
- Source Alpha
- Inverse Source Alpha
- Destination Alpha
- Inverse Destination Alpha

These blend factors are calculated for the source and destination pixels and the results are multiplied by the source and destination components.

3.2.4 2D Operation

The W2250 contains a BitBLT engine, a hardware cursor, and an extensive set of 2D registers and instructions.

The W2250 BitBLT engine provides hardware acceleration for many common WinCE and Java operations. The BitBLT engine is made up of two engines: Fixed BitBLT (BLT) and Stretch BitBLT (STRBLT). The term ‘BitBLT’ refers to block transfers of pixel data between memory locations. The term ‘Fixed’ is used to differentiate from the Stretch BLT engine.

The BLT engines can be used for the following functions:

- Move rectangular blocks of data between memory locations
- Pixel format conversion
- Data Alignment
- Perform logical operations

The W2250 has instructions to invoke BLT and STRBLT operations, permitting software to set up instruction buffers and to use batch processing.

Note: These instructions replace the need to do PIO directly to BLT and STRBLT registers.

3.2.4.1 Fixed Blt Engine

The rectangular block of data does not change while it is transferred between memory locations. Data to be transferred can consist of regions of memory, patterns, or solid color fills. A pattern is either an 8 x 8 brush pattern or a 32 x 1 line pattern.

The W2250 has the ability to expand monochrome data into a color depth of 8 or 16 bits. BLTs can be either opaque or transparent. Opaque transfers move the data specified to the destination. Transparent transfers compare destination color to source color and write according to the selected mode of transparency.

Data is horizontally and vertically aligned at the destination. If the destination for the BLT overlaps with the source memory location, the W2250 can specify which area within memory to begin the BLT transfer. Using the BLT engine accelerates the Graphical User Interface (GUI) of WinCE and Java. Hardware is included for all 256 Raster Operations (Source, Pattern, and Destination) defined by the operating system, including transparent BitBLT.

While the BitBLT engine is often used simply to copy a block of graphics data from the source to the destination, it also has the ability to perform more complex functions. The BitBLT engine is capable of receiving three different blocks of graphics data as input as shown in *Figure 3-10 BLT Engine Block Diagram on page 3-18* (source data, destination data, and pattern data). The source data may exist either in the Frame Buffer or it may be provided by the host CPU. The pattern data always represents an 8 x 8 block of pixels that must be located in the Frame Buffer, usually within the off-screen portion. The data already residing at the destination may also be used as an input, but this data must also be located in the Frame Buffer.

The BitBLT engine may use any combination of these three different blocks of graphics data as operands, in both bit-wise logical operations to generate the actual data to be written to the destination, and in per-pixel write-masking to control the writing of data to the destination. It is intended that the BitBLT engine will perform these bit-wise and per-pixel operations on color graphics data that is at the same color depth to which the rest of the graphics system has been set. However, if either the source or pattern data is monochrome, the BitBLT engine has the ability to put either block of graphics data through a process called “color expansion”. This process converts monochrome graphics data into color. Since the destination is often a location in the on-screen portion of the Frame Buffer, it is assumed that any data already at the destination will be of the appropriate color depth.

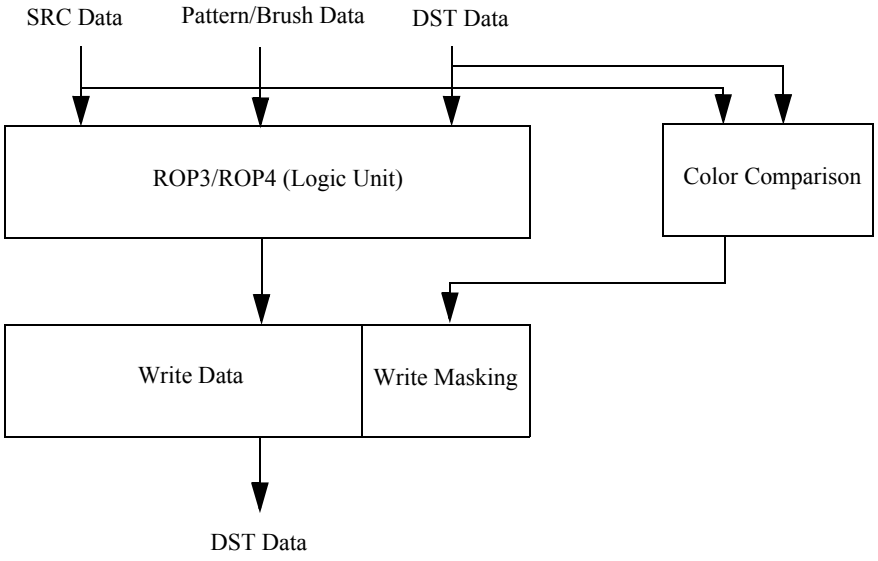

Figure 3-10 BLT Engine Block Diagram

3.2.4.2 Stretch BitBLT Engine

Stretch BitBLT can stretch source data in the X and Y directions to a destination larger or smaller than the source. Stretch BitBLT functionality expands a region of memory into a larger or smaller region using replication and interpolation.

Stretch BitBLT also provides format conversion and data alignment. Through a hardware DDA algorithm, expansion and shrinking can occur in both the horizontal and vertical directions in both integer and non-integer values.

3.3 Hardware Accelerated Video Processing

The W2250 incorporates an integrated DCT (Discrete Cosine Transform) engine as well as motion compensation (MC) support for the acceleration of MPEG decoding (see *Figure 3-11 Hardware Accelerated Decoder Flow*). The acceleration hardware is fed directly from an advanced packet parsing engine, enabling high processing rates with minimal software overhead.

Using the advanced packet parsing engine also increases the parallelism between the hardware and software which ensures that the most is made of the host processor's power. This combination of hardware acceleration decreases the loading on the host processor to significantly below that of past MC-only engines, and enables quality software MPEG-4 player playback for QCif or Cif image size.

The following operations are handled in hardware:

- 8X8 IDCT (JPEG/MPEG decode)
- 8X8 FDCT (JPEG encode)
- normal and unrestricted Motion Compensation (MC) for MPEG1/2/4 & H.263
- four zig-zag pattern support plus non-zigzag
- transparent (pre-computed coefficients) support

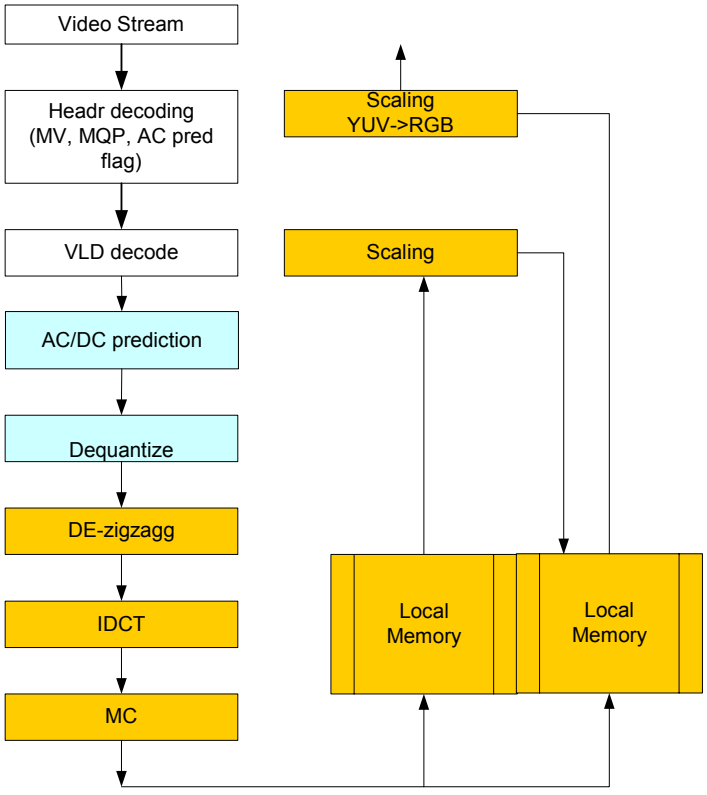

Figure 3-11 Hardware Accelerated Decoder Flow

3.3.1 IDCT Engine

The IDCT engine implements an IEEE 1180 compliant IDCT algorithm which, when combined with a run/level and dezig-zag formatter, off-loads a significant portion of the MPEG decoding process from the host processor. Run/level codes are combined with control words by the host processor in a packetized format, which the advanced packet parsing engine bus masters down from system memory over the host bus. The run/level codes are expanded, clamped and dezig-zagged within the IDCT engine. Software is responsible for de-quantizing and packing the data into runs and levels. Dezig-zagging is selectable through the register interface. There is also selection for intra/non-intra block support. The IDCT engine takes the dezig-zagged coefficients and performs a 2-D IDCT before buffering the result and sending it to the motion compensation hardware.

3.3.2 Motion Compensation

The motion compensation hardware allows for the implementation of all motion compensation modes required for MPEG-4 support. It fetches data from the frame buffer based on control information supplied by the advanced packet parsing engine on a macroblock basis, it combines the data from the frame buffer with the output of the IDCT engine through a full 9-bit signed adder, and it then writes the result back to the frame buffer.

YCrCb 420 → RGB color conversion is done on the fly in the display.

3.3.3 Portrait Mode Support

The following figure shows portrait mode support. The CPU/Host can queue MPEG commands to internal frame buffer with data, and check the status to see active frame number. To support temporal scalability, with enhanced layer, W2250 needs one additional buffer, 38Kbyte.

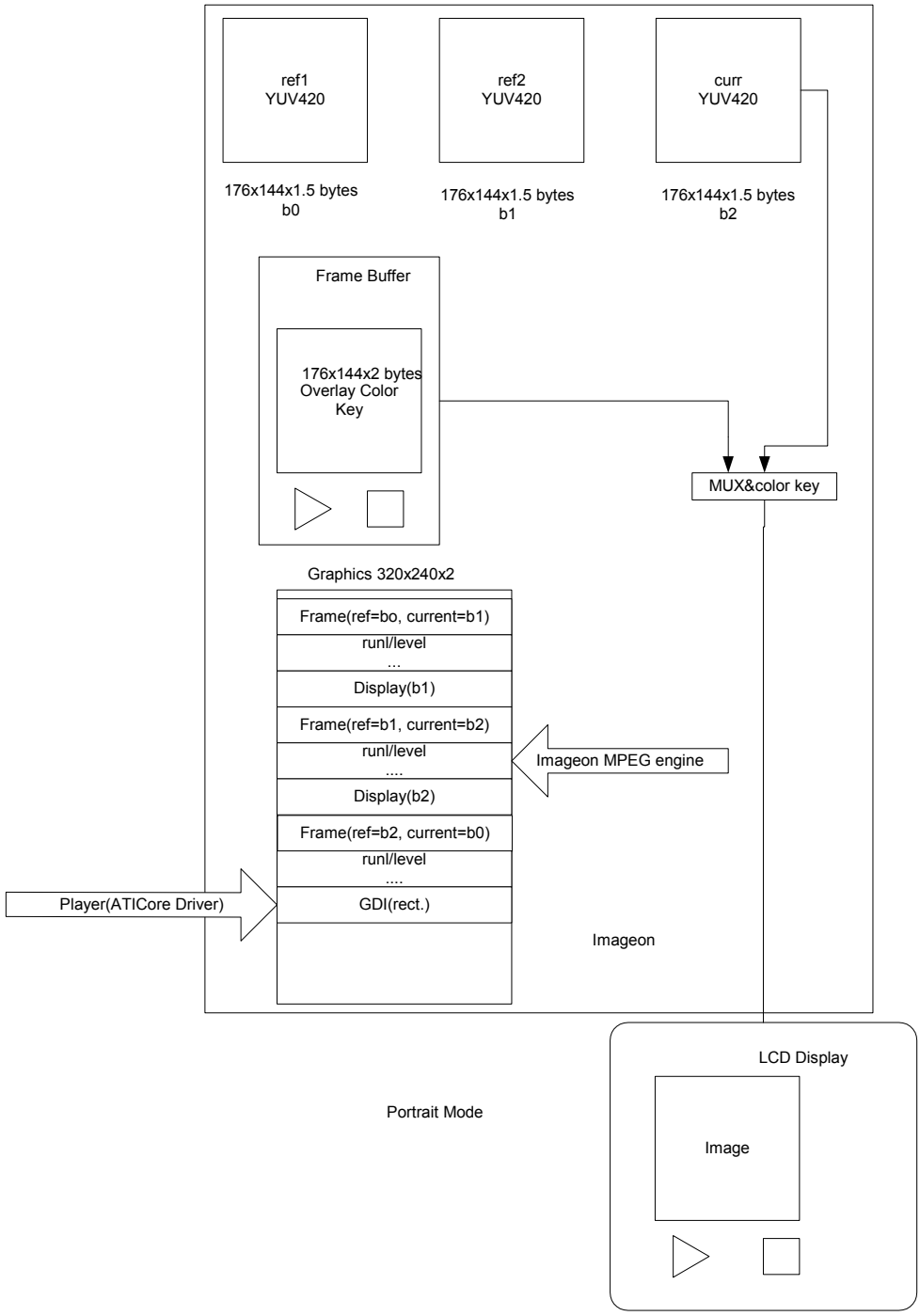

Figure 3-12 Portrait Mode Support

In the portrait mode, if panel is 176x220, the total memory requirements for QCif and temporal scalable profile is 114 Kbytes. There is also graphics surface for menu etc., which needs 68 Kbytes.

If panel is 240x320, there is enough memory to support (I, P) buffers for QCif with overlay always on top of the graphics.

3.4 Video Capture Port

The Video Capture Port has become the de facto standard for a multimedia video interface. It is based on the simplified ITU-656 standard. The W2250 has an 8-bit video capture port, it is designed to bring multimedia processing capabilities to a wide range of portable devices from digital cameras to pocket PCs and 2.5G and 3G wireless networks terminals. It provides the following set of features.

Table 3-13 Video Capture Port Features

Feature	Description
Input video formats	CCIR656 interlaced and non-interlaced video, Zoom video and I port
Input color modes	4:2:2 YCrCb, 4:2:2 YUV
Frame buffer color modes	4:2:2 packed, 4:2:2 plane and 4:2:0 plane
Color conversion	Internal
Maximum supported resolution	800x600 pixels (Larger resolution possible with limitations)
Multiple buffering scheme support	Single, double and triple buffering
Hardware decimation (picture downsizing)	Hardware decimation or down scaling 2:1, 4:1
Hardware image mirroring	Full hardware support
Picture clipping	Full hardware support
Interrupts	Upon completion of frame and/or field buffering
Synchronization	Video and display synchronization, automatic buffer flipping
Operation modes	Continuous and one-shot

3.4.1 Video Port Interface (ITU-R)

In ITU-R mode, Video Capture expects standard video stream as described in ITU-R Recommendation 656. Only F, V and H bits are analysed in timing reference code (SAV and EAV), the rest of the bits are not evaluated. As an extension of ITU-R 656, optional programmable invalid pixel code recognition can be turned on.

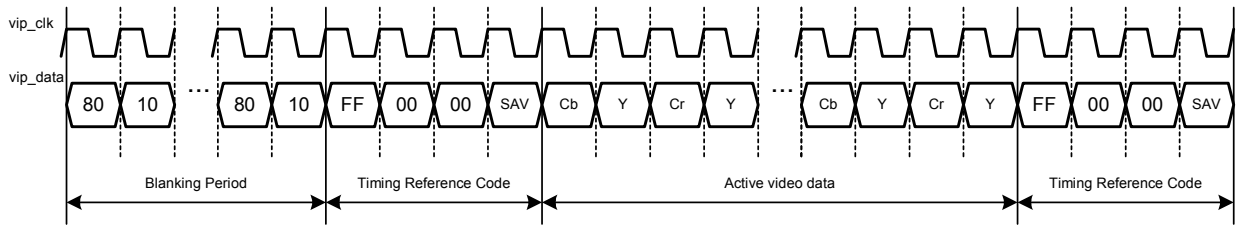

Figure 3-13 Video Port Timing Diagram (ITU-R Mode)

3.4.2 Video Port Interface (Zoom-Video Mode)

In Zoom-Video mode, VSYNC and HREF signals are used as video frame timing reference instead of timing reference code. HREF signal is used as horizontal active video qualifier. VSYNC signal is asserted at the beginning of each field. The shape of the signal is different for odd and even fields. VSYNC going low when HREF signal is high denotes the start of an even field. HREF high means that the next video line is the first line in an odd field.

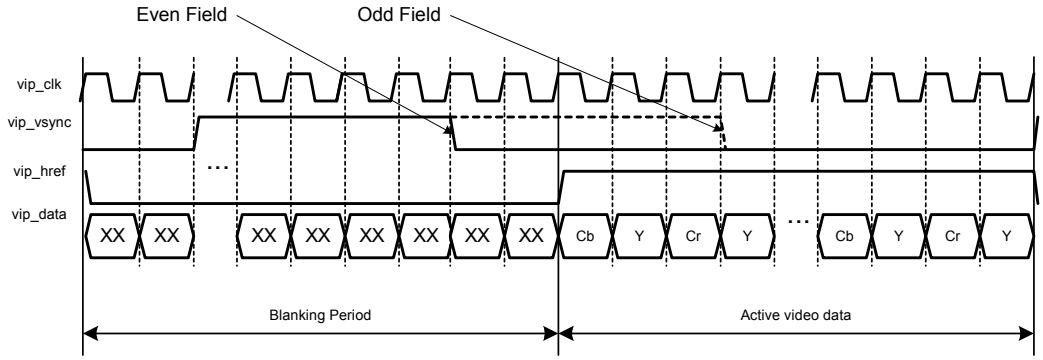

Figure 3-14 Video Port Timing Diagram (Zoom-Video Mode)

3.4.3 Video Port Interface (I-Port Mode)

Video capture supports both 4:2:2 and 4:2:0 I-Port video data formatting.

I-Port mode is designed to support video data stream generated by Philips SAA7114 and SAA 7118 video decoders. This video format is similar to ITU-R, but features some extensions. Some of them are:

- SAV and EAV codes are only present in those lines, where data is to be transferred, i.e., active video lines, or VBI-raw samples, no codes for empty lines.
- There may be more or less than 720 pixels between SAV and EAV.
- Data content and number of clock cycles during horizontal and vertical blanking is undefined, and may be not constant.
- Data stream may be interleaved with not-valid data codes, 00H, but SAV and EAV 4-byte codes are not interleaved with not-valid data codes.
- There may be an irregular pattern of not-valid data, or IDQ, and as a result, 'CB-Y-CR-Y -' is not in a fixed phase to a regular clock divider.

3.4.4 Video Port Timing Parameters

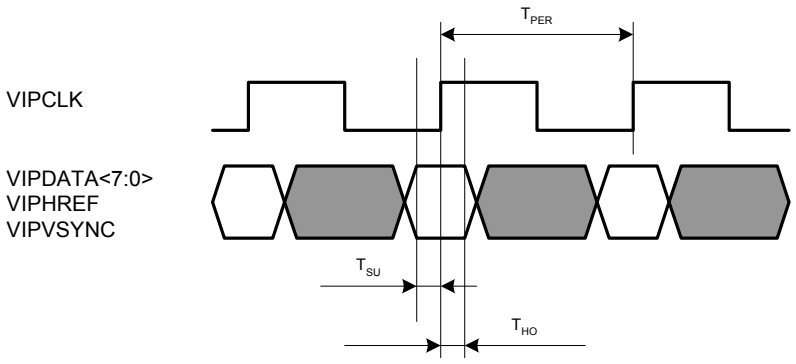

Figure 3-15 Video Port Interface Timing Diagram

Table 3-14 Video Port Timing Parameters

Parameter	Description	Type	Value (ns)
T_{per}	Clock Period of VIPCLK	min	30
T_{su} (VIPDATA)	Setup time of VIPDATA with respect to VIPCLK.	min	0
T_{ho} (VIPDATA)	Hold time of VIPDATA with respect to VIPCLK.	min	4
T_{su} (VIPHREF)	Setup time of VIPHREF with respect to VIPCLK.	min	0
T_{ho} (VIPHREF)	Hold time of VIPHREF with respect to VIPCLK.	min	6
T_{su} (VIPSYNC)	Setup time of VIPVSYNC with respect to VIPCLK.	min	6
T_{ho} (VIPSYNC)	Hold time of VIPVSYNC with respect to VIPCLK.	min	0

3.4.5 Connecting Video Capture to Video Decoders

Video capture can be connected to a wide variety of digital video decoders and single-chip digital video cameras. An example follows:

Single-chip Camera Module
Video capture supports a camera module. The camera module must be programmed to provide YCrCb output with byte ordering compatible with ITU-R 656 recommendation, i.e., CbYCrY.

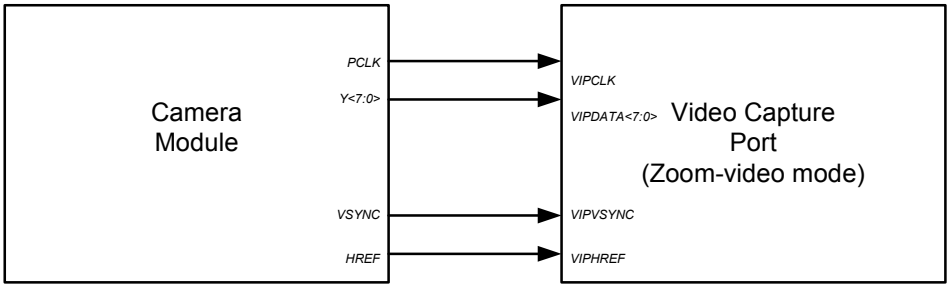

Figure 3-16 Connecting Video Capture Port to Digital Module

3.4.6 Applications

Video capture port can be used in a variety of applications. Below are just a few most common examples

Still-Image Cameras

Figure 3-17 Still-Image Camera Application on page 3-25 below shows how the video capture port is used in a still-image camera. The video capture port is programmed to operate in a one-shot mode. Start of capturing is controlled by the CPU. The captured image may be stored in either internal or external (depending on the size and resolution of the picture) frame-buffer memory for being shown on the LCD display. The frame buffer memory is accessible by the CPU, so the image then may be read, compressed and stored in the flash memory for non-volatile storage until archived. The CPU is notified when capture is complete by asserting an interrupt.

Maximum supported resolution of 800x600 pixels makes it possible to use IMAGEON™ in low or mid-end digital cameras.

There are two options as to how to deal with captured data.

In the first, the capture data will be transferred directly to the external memory in YUV4:2:0 format, using 460kBytes and ready for compression. For preview, data can be subsampled or scaled down to 320x240 using 115kbytes of internal memory.

In the second, capture data will be encoded on the fly to produce a baseline sequential JPEG scan, and the CPU will be signalled via interrupt to indicate that data is ready. JPEG encoding includes DCT, quantization using user defined table

of q values, ZigZag scan and Variable length Encode. Preview is done by decimating the input video stream 4:1 by dropping pixels or filtering.

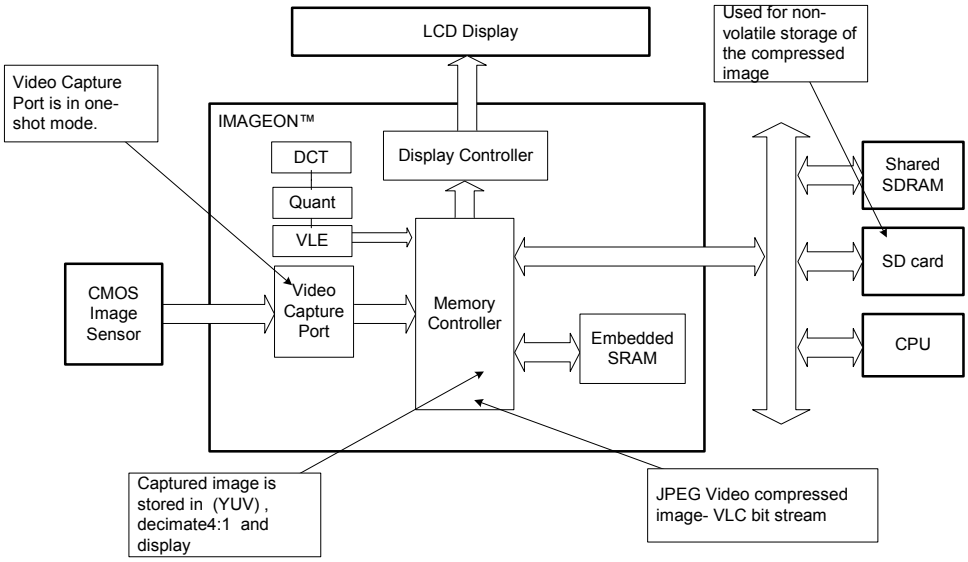

Figure 3-17 Still-Image Camera Application

Figure 3-18 Embedded Memory Allocation - Picture Preview Mode on page 3-26 and *Figure 3-19 Embedded Memory Allocation - JPEG Compression Mode on page 3-26* show the embedded memory allocation for picture preview and JPEG encoding modes, respectively, in the case of progressive 4:2:2 formatted input video stream.

Picture preview mode requires an allocation of memory buffer of a maximum of 16 lines (two 8-line ping-pong style). Buffer of the maximum size is needed when 4:2:2 to 4:2:0 format conversion is turned on and the scaling factor is 1:4. It is also assumed that progressive video input will require double video buffer scheme, i.e. one frame of video is being displayed while the next one is being captured into a separate buffer.

For JPEG encoding, VIP buffer is organized in the same ping-pong manner, but the buffer size will increase to 32 lines of input video if 4:2:2 to 4:2:0 format conversion is turned on, and remains the same if 4:2:2 to 4:2:0 format conversion is off. The display controller will use only one video buffer if the image being displayed is still. The second display video buffer will be used to store a JPEG-compressed output image.

The overall memory allocation requirements are given in *Table 3-15 on page 3-27.*

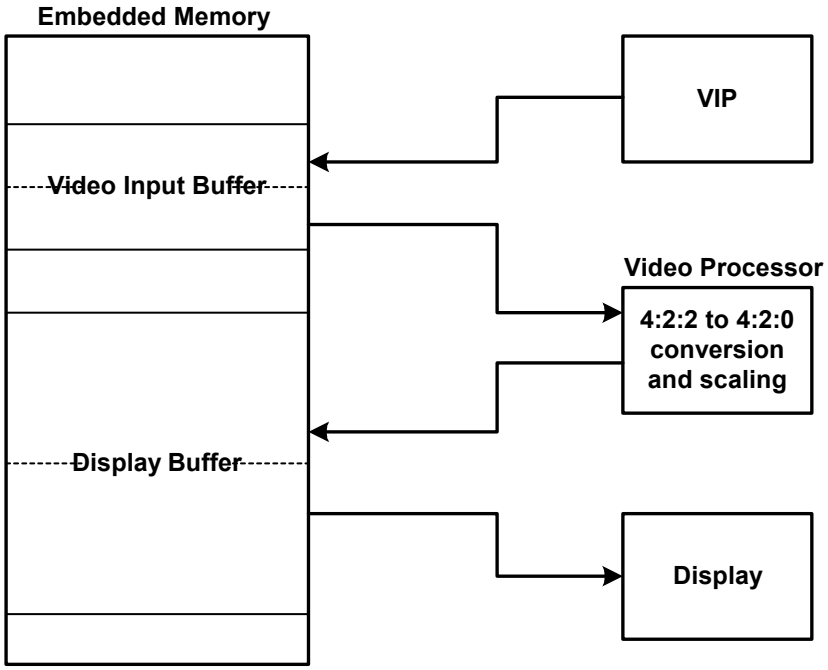

Figure 3-18 Embedded Memory Allocation - Picture Preview Mode

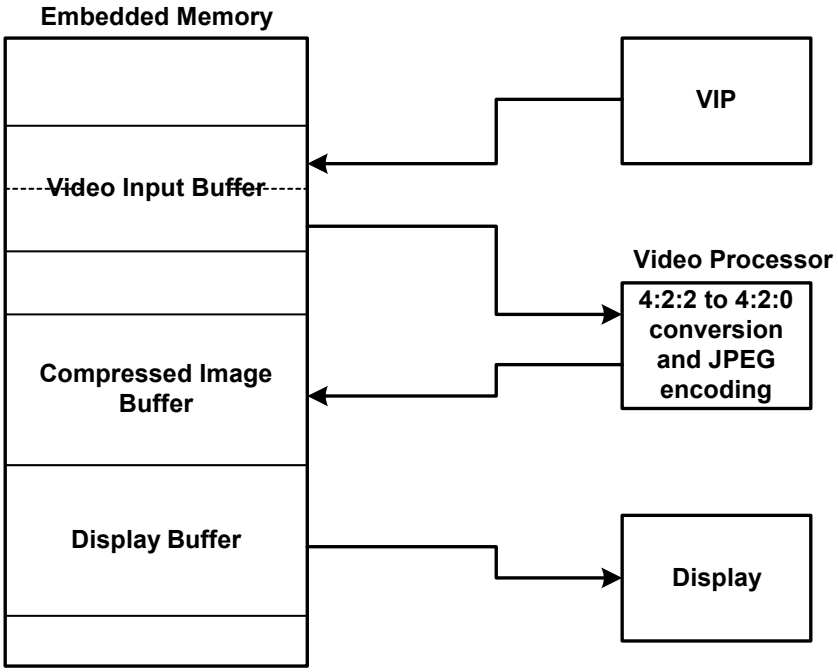

Figure 3-19 Embedded Memory Allocation - JPEG Compression Mode

Table 3-15 Memory Size Requirement For Video Application

Input Video Resolution	Scaling for preview	Format	Video input buffer size, bytes	Display buffer size, bytes	Total memory allocated, bytes
320x240	1:2	4:2:2	10240	76800	87040
		4:2:0	20480	57600	78080
	1:4	4:2:2	10240	19200	29440
		4:2:0	20480	14400	34880
640x480	1:2	4:2:2	Not supported		
		4:2:0	Not supported		
	1:4	4:2:2	20480	76800	97280
		4:2:0	40960	57600	98560
800x600	1:2	4:2:2	Not supported		
		4:2:0	Not supported		
	1:4	4:2:2	25600	120000	145600
		4:2:0	51200	90000	141200
1024x768	Not supported[1]				

Note 1: Resolution of 1024x768 pixels is still supported if JPEG compression is done on a 1:4 downsized image in 4:2:0 format. Memory size allocated for video buffers is 180224 bytes.

Note 2: If JPEG encodes a scaled image, the scaling and writeback to memory must happen first (i.e. scaling and JPEG encode on the fly is not supported).

3.4.7 VCLK Frequencies

The following illustrates how to achieve different VCLK frequencies:

For VCLK = 14.3 Mhz, closest = 14.3 Mhz
XCLK Frequency = 4.576

SYSTEM PLL Frequency = 57.2

OUTPUT Frequency = 14.3

PLL 32 FeedBackDivider = 143

SYSTEM FeedBackDivider = 12

SYSTEM FractionDivider = 0.5

SYSTEM ReferenceDivider = 1

PostDivider = 4

For VCLK = 14.2 Mhz, closest = 14.2 Mhz
XCLK Frequency = 4

SYSTEM PLL Frequency = 71

OUTPUT Frequency = 14.2

PLL 32 FeedBackDivider = 125

SYSTEM FeedBackDivider = 17

SYSTEM FractionDivider = 0.75

SYSTEM ReferenceDivider = 1

PostDivider = 5

For VCLK = 13.39 Mhz, closest = 13.4 Mhz
XCLK Frequency = 4

SYSTEM PLL Frequency = 67

OUTPUT Frequency = 13.4

PLL 32 FeedBackDivider = 125

SYSTEM FeedBackDivider = 16

SYSTEM FractionDivider = 0.75

SYSTEM ReferenceDivider = 1

PostDivider = 5

3.4.8 YCrCb Color Conversion

The equations used in the W2250 for YCrCb Color Conversion are based on the ITU_R BT.601 (CCIR 601) standard where:

- Y is in the range of 16..235
- Cr,Cb is in the range of 16..240, with 128 equal to zero.

R=Y+1.37(Cr-128)

G=Y-0.698(Cr-128)-0.336(Cb-128)

B=Y+1.73(Cb-128)

These equations will produce RGB values with a nominal range of 16..235 (black-white).

To provide gamma-corrected RGB data with a range of 0..255, the following equations are used:

R = 1.164*(Y-16) + 1.596*(Cr-128)

G = 1.164*(Y-16) - 0.813*(Cr-128) - 0.392*(Cb-128)

B = 1.164*(Y-16) + 2.017*(Cb-128)

3.5 SD Memory and IO Controller

The SD (Secure Digital) implementation in the W2250 is a memory/IO card interface serial controller.

3.5.1 Functional Overview

- Data transmission/reception in frame units
- Compatibility with MultiMediaCard
- Error check: CRC7(for commands), CRC16(for data)
- SD Memory/IO Card Interface: COMMAND(1 line), Data/INT(4 lines)
- 1k byte data buffer (Synchronous single port SRAM)
- Card detection/removal support, either through a dedicated pin for each card SDC[1:0]CD or through SD[1:0]DAT3 data lines.
- Write Protect feature using mechanical switch
- SDbuffer underflow/overflow, timeout (response, other), END, CRC, CMD
- Sector counter for multiple Read/Write operations
- Any byte size transfer support. Sector size can be set from 1 byte to 512 bytes.
- General specification:
  - Operational voltage 3.3V + 5%
  - Variable clock rate 0 to 25Mhz
  - Up to 10MByte/sec read/write rate (using 4 parallel data lines)
  - Two internal SRAM buffers: 256x16x2
  - Low power consumption due to implementation of gated clocks

3.5.2 SD Host Interface

Note: C0 indicates the SD card/port number.

Table 3-16 SD Host Interface

Description	Pin	SD mode	Description	SDIO mode	Description
SDC0CLK	O	CLK_C0	Clock	CLK_C0	Clock
SDC0CMD	I/O	CMD_C0	Command / Response	CMD_C0	Command / Response
SDC0DQ0	I/O	DAT0_C0	Data 0 or READ_BUSY	DAT0_C0	Data 0
SDC0DQ1	I/O	DAT1_C0	Data 1	DAT1_C0	Data 1 or IRQ
SDC0DQ2	I/O	DAT2_C0	Data 2	DAT2_C0	Data 2 or READ_BUSY
SDC0DQ3	I/O	DAT3_C0	Data 3	DAT3_C0	Data 3
SDC0CD	I	Card detection for card C0			
SDC0WP	I	Position of Write Protect Mechanical Switch for card C0			

3.5.3 SD Block Diagram

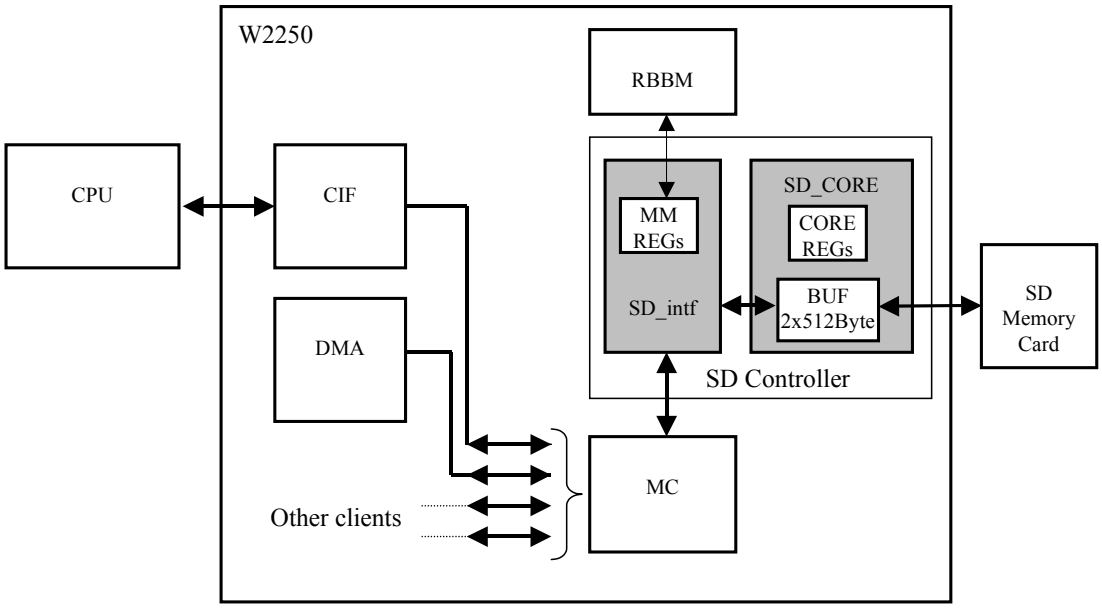

Figure 3-20 SD Block Diagram

The **Memory Mapped registers** (MM REGs) are accessed through the RBBM path, and are used for:

- Controlling the SD Controller pads, and
- Returning auxiliary STATUS information.

The **SD_CORE registers** (CORE REGs) are accessed through the Memory Controller (MC) path, using address [6:0] mapped to [8:2] on the CPU interface (CIF).

CORE REGs provide the required interface control for efficient data transfer between the W2250 and the SD Memory/IO card. Data transfer can be driven by any MC client, mainly by CIF or DMA.

3.5.4 Data Transfer Diagram

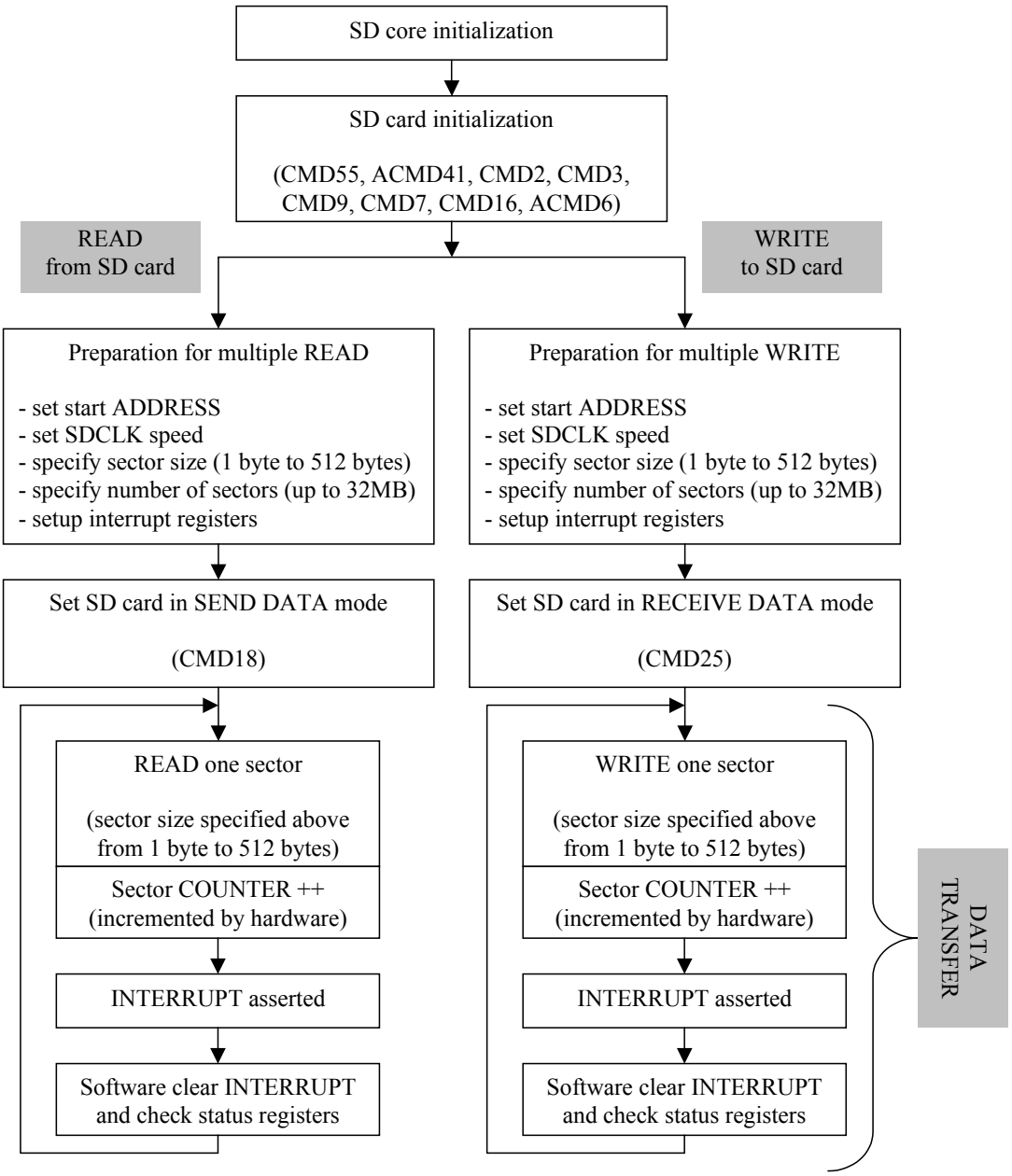

Figure 3-21 DataTransfer Diagram

Notes:

- Data transfer process is continuous until specified number of sectors is reached.
- Data transfer operation can be driven in two ways:
  - by HOST: System Memory -> CPU -> SD BUF (1K) -> SD card
  - by DMA: System Memory -> SD BUF (1K) -> SD card.

3.5.5 SDIO Timing

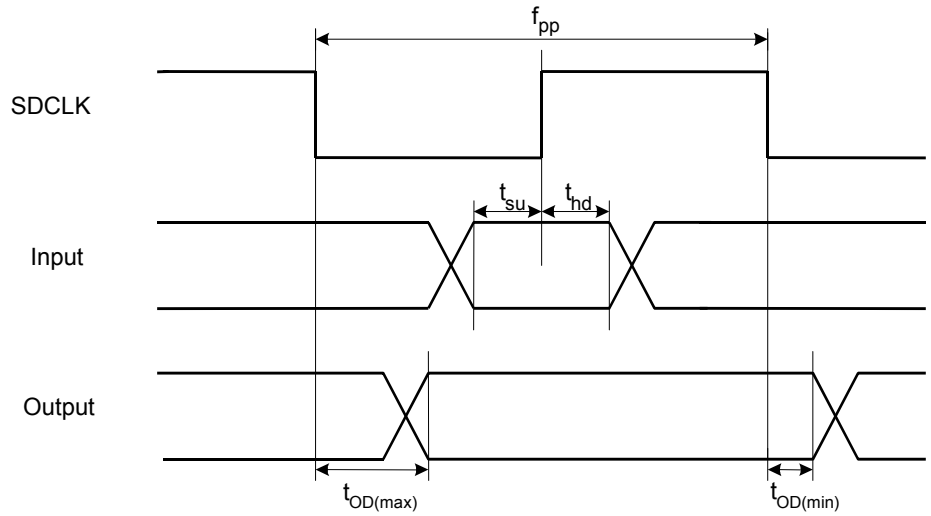

Figure 3-22 SDIO Setup and Hold Timing

Table 3-17 SDIO Timing Parameters

Parameter	Symbol	Min	Max	Units
Clock Frequency	fpp	0	25	MHz
Input (CMD, DAT with respect to SDCLK)				
Setup Time	tsu	5		ns
Hold Time	thd	5		ns
Output (CMD, DAT with respect to SDCLK)				
Output Delay	tOD	0	5	ns

3.6 CPU Interface

3.6.1 Supported Features

- Endian Swapping
  - FIX SWAPPING : support for BIG endian platforms is assured by swapping/unswapping data directly, during READ and WRITE operations
  - DYNAMIC SWAPPING : address decoding defines three swapping apertures (0-4, 4-8, 8-16MB), each of them applying an individual swapping scheme, selected by register
- ADDRESS WRAP AROUND, during READ or WRITE operations, is possible by defining an wrapping aperture outside the Frame Buffer
- BURST READ from Frame Buffer
  - PRE-FETCH from Frame Buffer : 16 dwords are fetched in advance by auto-incrementing the current address that is being READ. This process is continuous as long as address comes in incremented by one dword, otherwise the process is terminated and pre-fetched data is invalidated.
  - PRE-FETCH from SDIO : READ data from SD buffer is being fetched in advance until the full SECTOR size is reached.
- TUNE-UP driving strength of output pads : WAITb, DQb for READ DATA, and INTERRUPTb
- COHERENCY cases:
  - READs will flush WRITEs.
  - READs will NOT pass WRITEs.
  - WRITEs to Memory Mapped registers will NOT pass WRITEs to Frame Buffer.

3.6.2 CPU Interface

3.6.2.1 Parallel CPU Interface (powered on VDDR1)

Table 3-18 Parallel CPU Interface (powered on VDDR1)

Pin Name	Type	Description
DQ[15:0]	I/O	Data bus
AD[22:0]	I	Address bus
CSb	I	Chip Select active LOW
OEb	I	Output Enable active LOW
WEb	I	Write Enable active LOW (optional active HIGH by STRAP_INV_DMAC_CLK)
BE[1:0]	I	Byte Enable during Write Operation
INTb	O	INTERRUPT active LOW (optional active HIGH by programming)
RESETb	I	System RESET active LOW
MR	I	Chip Select active HIGH (indirect mode) or R/Wb for early wait (direct mode)
WAITb	I	Used in Direct mode only. Wait for Data ready (read cycle) or accepted (write cycle). Selected by strap to be active LOW (default) or HIGH.
SRCCLK	I	32kHz clock input

Notes:

- The data bus is available in either 16-bit or 8-bit configuration, determined by packaging.
- INDIRECT mode requires only AD[0] to be driven by CPU. In this case, all the rest of AD pins are internally pulled down.

3.6.3 Configuration Registers

Table 3-19 Configuration Registers

Field Name	Offset	Bits	Default	Description
IND_ADDR_A_0	0x0	7:0	0x0	This register will read back incremented byte address, if AD_INC_A=1. Also, see ACCESS_IND_ADDR_A
IND_ADDR_A_1	0x1	7:0	0x0	
IND_ADDR_A_2	0x2	7:0	0x0	
IND_DATA_A_0	0x3	7:0	0x0	Access Registers and Frame Buffer trough channel A; In Indirect addressing mode, IND_DATA[3]=RD_DATA_RDY_A if AD[0]=1
REG_BASE	0x4	7:0	0x1	Default start of MM register space is 0x010000 (64KB)
INTF_CNTL	0x5	7:0	<see below>	Note: RD_FETCH_TRIGGER/RD_DATA_RDY fields are to be used only when CPU does NOT support WAITb. SEE DETAILED DESCRIPTION BELOW
STATUS	0x6	7:0	<see below>	SEE DETAILED DESCRIPTION BELOW
CPU_DEFAULTS	0x7	7:0	<see below>	SEE DETAILED DESCRIPTION BELOW
IND_ADDR_B_0	0x8	7:0	0x0	This register will read back incremented byte address, if AD_INC_B=1. Also, see ACCESS_IND_ADDR_B
IND_ADDR_B_1	0x9	7:0	0x0	
IND_ADDR_B_2	0xA	7:0	0x0	
IND_DATA_B_0	0xB	7:0	0x0	Access Registers and Frame Buffer trough channel A; In Indirect addressing mode, IND_DATA[3]=RD_DATA_RDY_B if AD[0]=1
PM4_RPTR(Access: R)	0xC	7:0	0x0	Used in combination with RD_FETCH_PM4_RPTR. Note: UNPACK_RD_DATA setting does take effect when reading from this field; POSSIBLE VALUES: 00 - blk_size=4Kb (32 bytes/unit) => 128 units 01 - blk_size=8Kb (32 bytes/unit) => 256 units 02 - blk_size=16Kb (64 bytes/unit) => 256 units 03 - blk_size=32Kb (128 bytes/unit) => 256 units 04 - blk_size=64Kb (256 bytes/unit) => 256 units
BUS_CNTL	0xD	7:0		Defines HardWare configurations for BUS protocol SEE DETAILED DESCRIPTION BELOW
PM4_WRPTR_0	0xE	7:0	0x0	PM4_WRPTR[7:0]
PM4_WRPTR_1	0xF	7:0	0x0	PM4_WRPTR[13:8]

Table 3-20 Configuration Registers - Detailed description of INTF_CNTL field, Offset 0x5

Field Name	Bits	Default	Description
AD_INC_A	0	0x1	POSSIBLE VALUES: 00 - access only the location defined by current IND_ADDR_A[23:0] 01 - increment address starting from current IND_ADDR_A[23:0]
RING_BUF_A	1	0x0	Used in combination with MM registers: OFFSET_ADDR_A, BLOCK_SIZE_A. Note: when RING_BUF_A=1, all READs and WRITEs are forced to Frame Buffer (to disable this feature, set DIS_RING_BUF_TO_FORCE_DEC) POSSIBLE VALUES: 00 - wrap around disabled 01 - enable address wrap around, when accessing the Frame Buffer
RD_FETCH_TRIGGER_A(Access: W)	2	0x0	Writing repetitively to this field, does not have any effect if RD_DATA_RDY_A is still '0'. Could be disabled by CIF_READ_DBG/DIS_REG_RD_FETCH_TRIG POSSIBLE VALUES: 01 - trigger data fetching from current IND_ADDR_A
RD_DATA_RDY_A(Access: R)	3	0x0	Polling this bit generates a fetch trigger, similar with RD_FETCH_TRIGGER_A POSSIBLE VALUES: 00 - WAIT, read data is not available 01 - one dword is available

Table 3-20 Configuration Registers - Detailed description of INTF_CNTL field, Offset 0x5 (Continued)

Field Name	Bits	Default	Description
AD_INC_B	4	0x1	POSSIBLE VALUES: 00 - access only the location defined by current IND_ADDR_B[23:0] 01 - increment address starting from current IND_ADDR_B[23:0]
RING_BUF_B	5	0x0	Used in combination with MM registers: OFFSET_ADDR_B, BLOCK_SIZE_B. Note: when RING_BUF_B=1, all READs and WRITEs are forced to Frame Buffer (to disable this feature, set DIS_RING_BUF_TO_FORCE_DEC) POSSIBLE VALUES: 00 - wrap around disabled 01 - enable address wrap around, when accessing the Frame Buffer
RD_FETCH_TRIGGER_B(Access: W)	6	0x0	Writing repetitively to this field, does not have any effect if RD_DATA_RDY_B is still '0'. Could be disabled by CIF_READ_DBG/DIS_REG_RD_FETCH_TRIG POSSIBLE VALUES: 01 - trigger data fetching from current IND_ADDR_B
RD_DATA_RDY_B(Access: R)	7	0x0	Polling this bit generates a fetch trigger, similar with RD_FETCH_TRIGGER_B POSSIBLE VALUES: 00 - WAIT, read data is not available 01 - one dword is available

Table 3-21 Configuration Registers - Detailed description of STATUS field, Offset 0x6

Field Name	Bits	Default	Description
WR_FIFO_AVAILABLE_SPACE(Access: R)	1:0	0x0	Note: use this field only when CPU does NOT support WAITb POSSIBLE VALUES: 00 - FULL 01 - 2 entries available 02 - 4 entries available 03 - EMPTY: 8 entries available
FBUF_WR_PIPE_EMP(Access: R)	2	0x0	See CIF_WRITE_DBG/DIS_MC_CLEAN_COND POSSIBLE VALUES: 01 - Write path to Frame buffer is empty
SOFT_RESET	3	0x0	The whole chip stays in RESET as long as this bit is set. During SOFT_RESET only this field could be written/read POSSIBLE VALUES: 00 - normal mode 01 - RESET whole chip (SOFT_RESET overwrites GLOBAL_RESET)
SYSTEM_PWM_MODE(Access: R)	5:4	0x0	POSSIBLE VALUES: 00 - FAST Clock mode 01 - NORMAL mode 02 - undefined 03 - SUSPEND mode
MEM_ACCESS_DIS	6	0x0	POSSIBLE VALUES: 00 - normal mode 01 - disable any access to Registers or Frame Buffer
EN_PRE_FETCH	7	0x1	POSSIBLE VALUES: 00 - Pre-fetch disabled 01 - enabled

Table 3-22 Configuration Registers - Detailed description of CPU_DEFAULTS field, Offset 0x7

Field Name	Bits	Default	Description
UNPACK_RD_DATA	0	0x0	Defines how READ data is being driven out: POSSIBLE VALUES: 00 - LSByte first 01 - MSByte first

Table 3-22 Configuration Registers - Detailed description of CPU_DEFAULTS field, Offset 0x7 (Continued)

Field Name	Bits	Default	Description
ACCESS_IND_ADDR_A	1	0x1	This feature applies during READ or WRITE POSSIBLE VALUES: 00 - H/W does NOT increment INDEX when accessing IND_ADDR_A_0/1/2 01 - S/W keep INDEX to IND_ADDR_A_0, while H/W increments INDEX when accessing IND_ADDR_A_0/1/2
ACCESS_IND_ADDR_B	2	0x1	This feature applies during READ or WRITE POSSIBLE VALUES: 00 - H/W does NOT increment INDEX when accessing IND_ADDR_B_0/1/2 01 - S/W keep INDEX to IND_ADDR_B_0, while H/W increments INDEX when accessing IND_ADDR_B_0/1/2
ACCESS_SCRATCH_REG	3	0x0	Note: when ACCESS_SCRATCH_REG=1, UNPACK_RD_DATA setting takes effect POSSIBLE VALUES: 00 - read only the lowest byte of SCRATCH_REG0 01 - read the whole dword SCRATCH_REG0
PACK_WR_DATA	4	0x0	Defines how WRITE data is expected from CPU: POSSIBLE VALUES: 00 - LSByte first 01 - MSByte first
TRANSITION_SIZE	5	0x0	Note: This is only related with IND_DATA_A/B WRITE operation. READ is NOT affected POSSIBLE VALUES: 00 - DWORD MODE 01 - NON-DWORD MODE
UPDATE_RECTANGLE_PARAM(Access: W)	6	0x0	After this bit is set, next two bytes written, are assigned to RECTANGLE parameters: first/lower byte to PITCH[9:2] and second/upper byte to LENGTH[7:0]. Note: this bit is cleared by hardware after second byte is written
RD_FETCH_SCRATCH	7	0x0	Writing ‘1’ to this field will trigger copying current content of SCRATCH_REG0, into SCRATCH configuration field. Poll this bit until is set, to find out when SCRATCH field contains valid data POSSIBLE VALUES: 00 - Data not available 01 - SCRATCH data available for READ

Table 3-23 Configuration Registers - Detailed description of BUS_CNTL field, Offset 0xD

Field Name	Bits	Default	Description
DIS_WAITb	0	0x1	00 - WAITb pad (output) is enabled 01 - disabled
SEL_BE	1	0x0	When SEL_BE=1, always assume BYTE writes. When SEL_BE=0, IND_ADDR_A/B bit[0] is ignored (in 16bit configuration) POSSIBLE VALUES: 00 - Byte Enable controlled by CPU_BE[1:0] 01 - use IND_ADDR_A/B[0] as BE
SWAP_BY_SIZE	2	0x0	POSSIBLE VALUES: 00 - disabled 01 - SWAP bytes within dword boundary, based in how many bytes are filled in each dword during WRITE operation

Table 3-23 Configuration Registers - Detailed description of BUS_CNTL field, Offset 0xD (Continued)

Field Name	Bits	Default	Description
RECTANGLE_ACCESS	3	0x0	Hardware will clear this bit after second write to IND_ADDR_A_0 is detected. Does not apply for channel B POSSIBLE VALUES: 00 - disabled 01 - IND_ADDR_A auto-incrementing, complies with a rectangle fill/dump
EN_SUSTAIN_DWORD_ADDR_A	4	0x0	IND_ADDR_A[23:2] will NOT increment. Hardware will clear this bit after second write to IND_ADDR_A_0 is detected POSSIBLE VALUES: 00 - disabled 01 - IND_ADDR_A does not increment beyond one dword boundary, during READ or WRITE
EN_SUSTAIN_DWORD_ADDR_B	5	0x0	IND_ADDR_B[23:2] will NOT increment. Hardware will clear this bit after second write to IND_ADDR_B_0 is detected POSSIBLE VALUES: 00 - disabled 01 - IND_ADDR_B does not increment beyond one dword boundary, during READ or WRITE
EN_4BIT_ADDR	0	0x0	Note: to switch to 4bit addressing mode, write to this bit using 1bit addressing protocol POSSIBLE VALUES: 00 - 1bit addressing mode 01 - 4bit addressing mode
PACK_QSPI	7	0x1	POSSIBLE VALUES: 00 - LSBit first 01 - MSBit first

3.6.4 READ Protocol - INDIRECT Mode

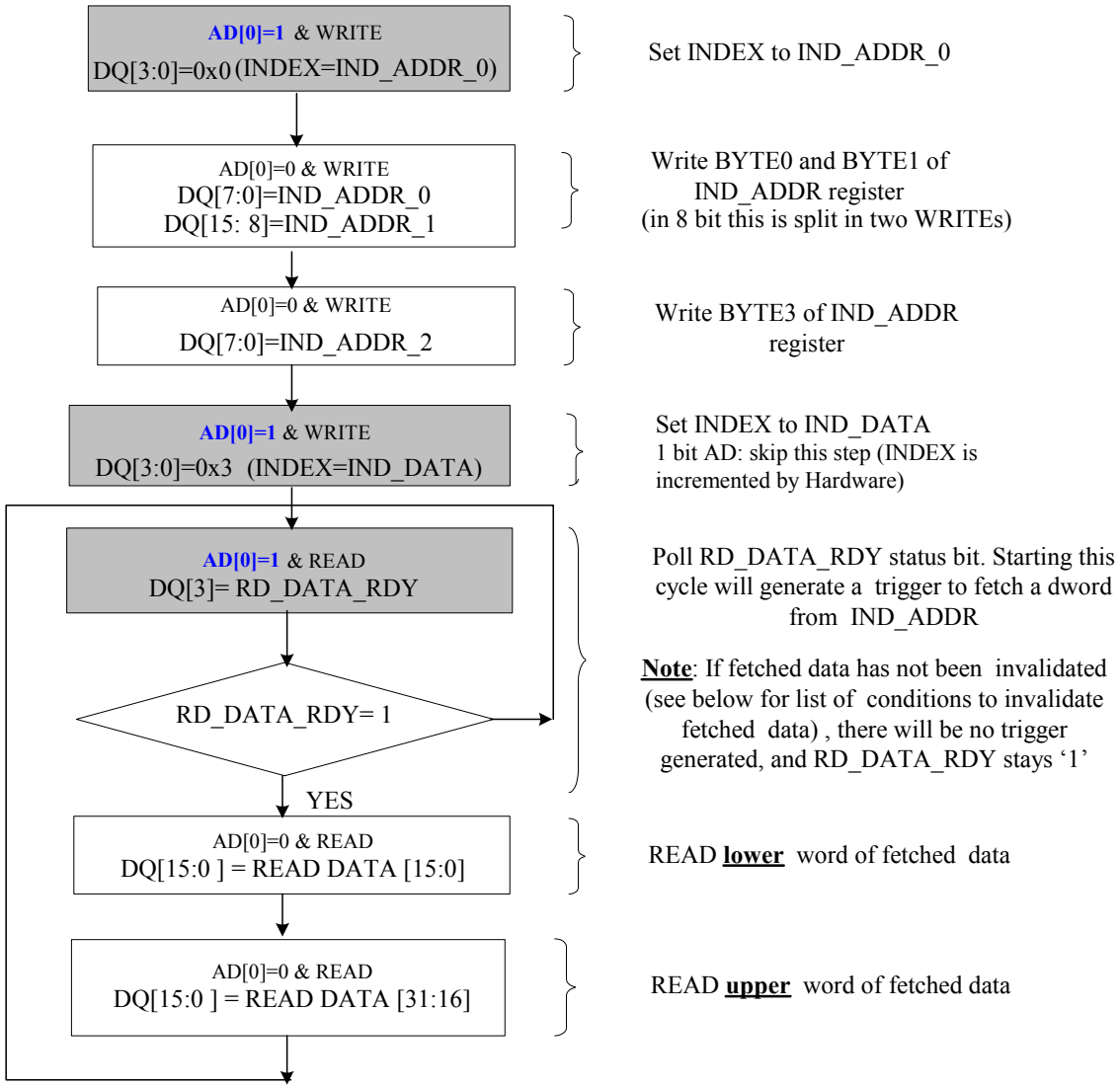

Figure 3-23 READ Protocol without Pre-fetch (ADDR=1 bit, DATA = 16 bit) - Indirect Mode

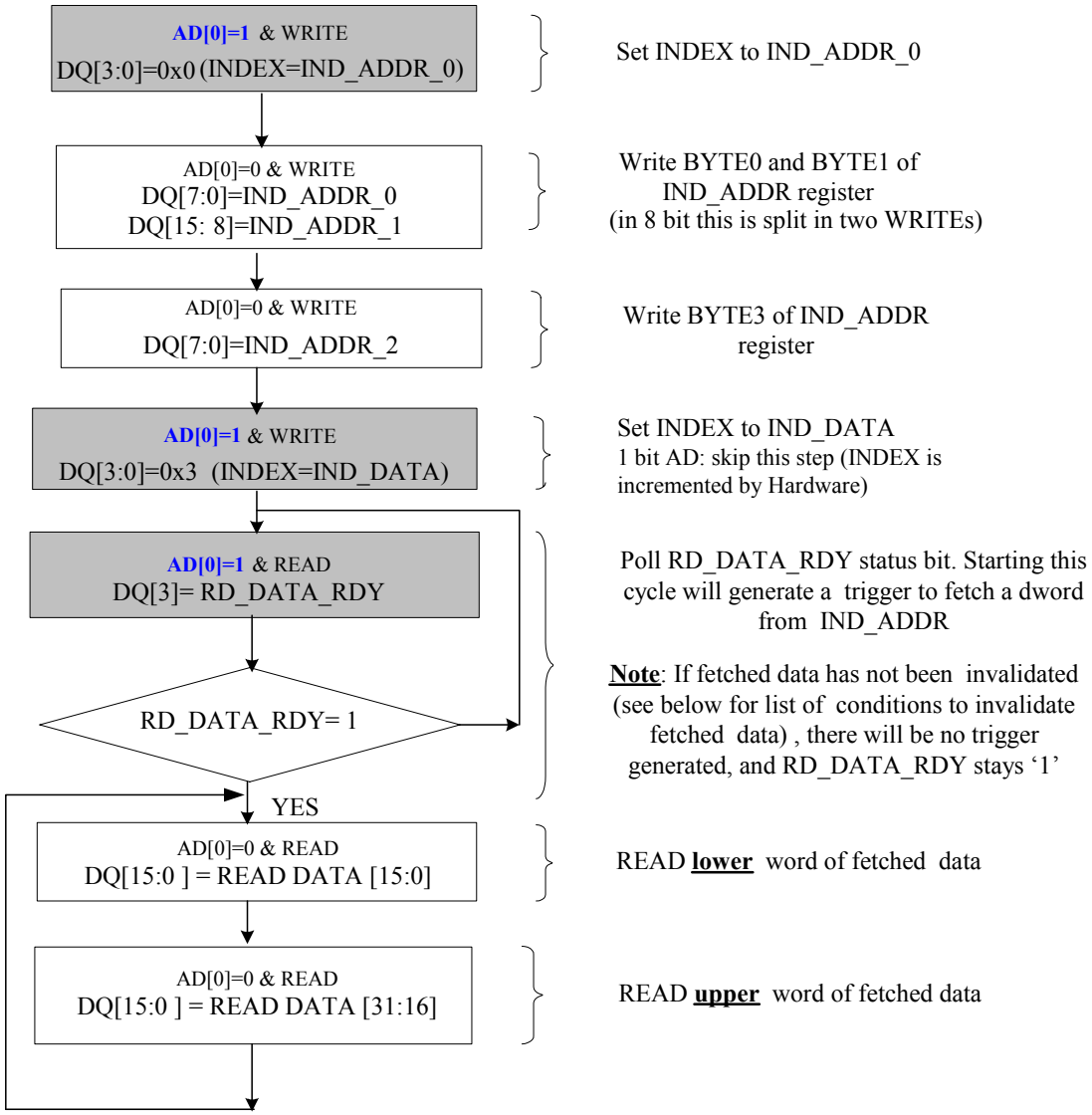

Figure 3-24 READ Protocol with Pre-fetch (ADDR=1 bit, DATA = 16 bit) - Indirect Mode

3.6.5 Read Waveforms - INDIRECT Mode

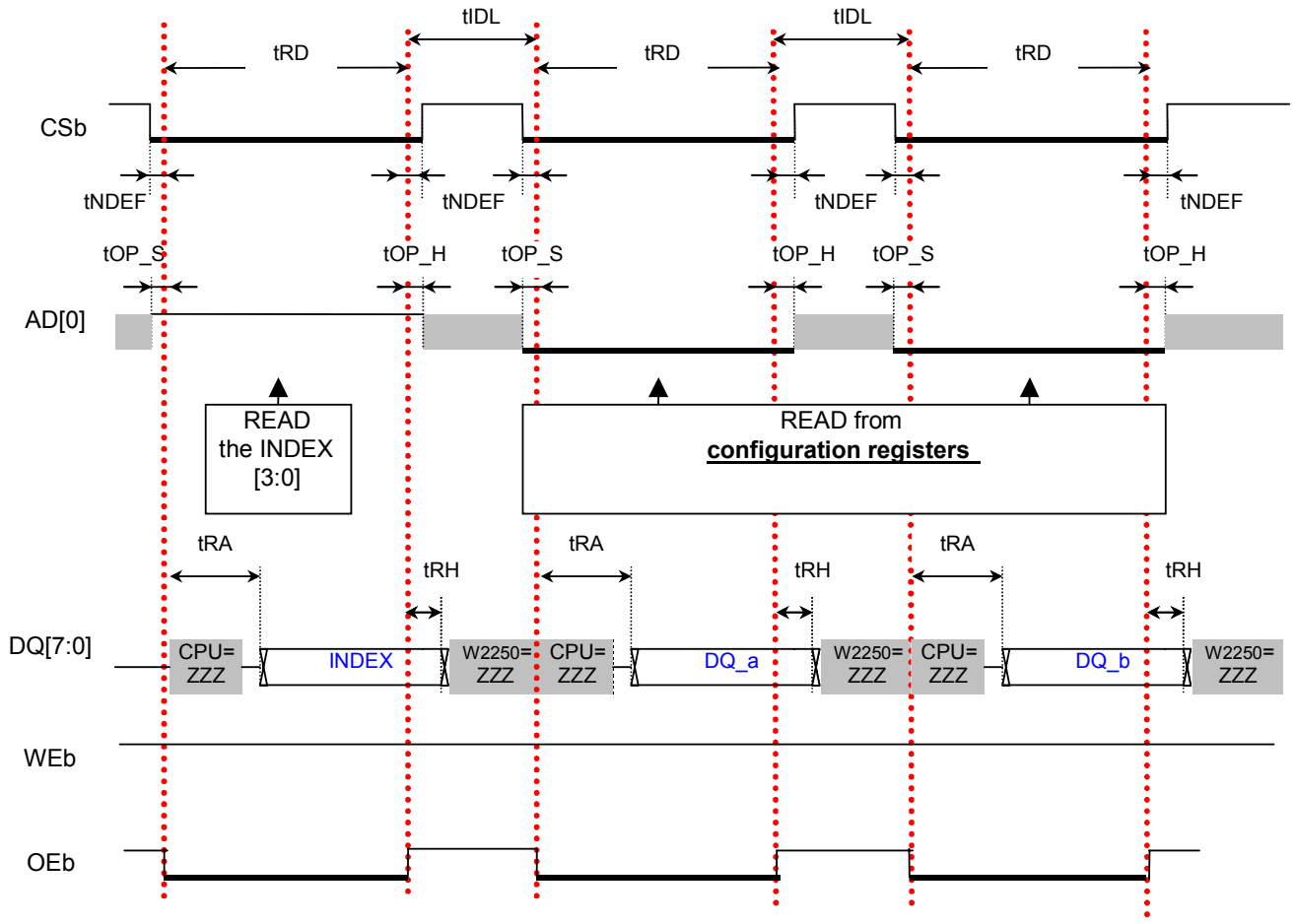

Figure 3-25 Read from Configuration Registers (ADDR=1 bit, DATA = 16 bit) - Indirect Mode

Note: Since configuration registers size is only one byte, they are only accessed by lower byte of DQ bus.

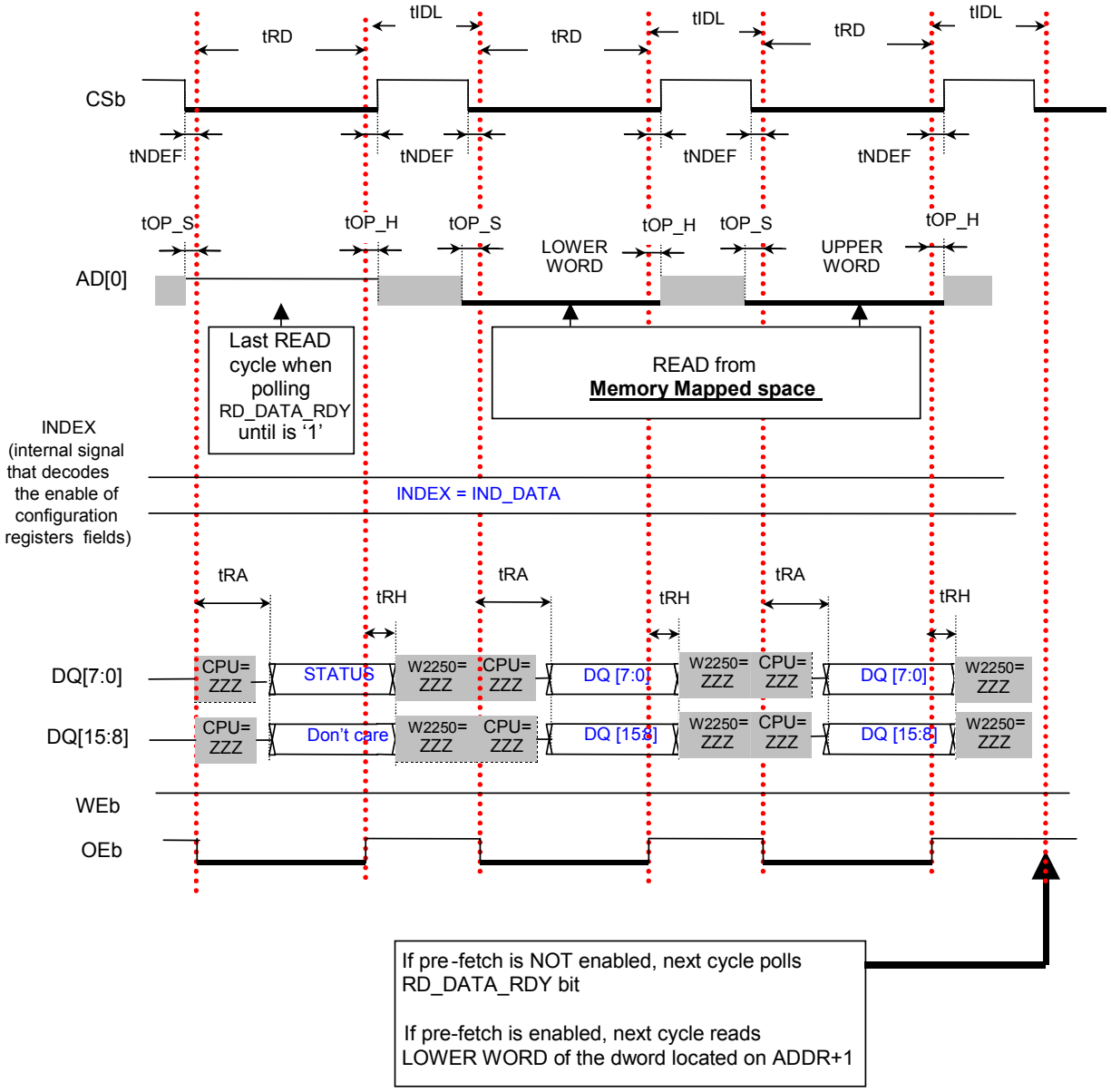

Figure 3-26 Read from Memory Mapped Space - Indirect Mode

3.6.6 Write Protocol - INDIRECT Mode

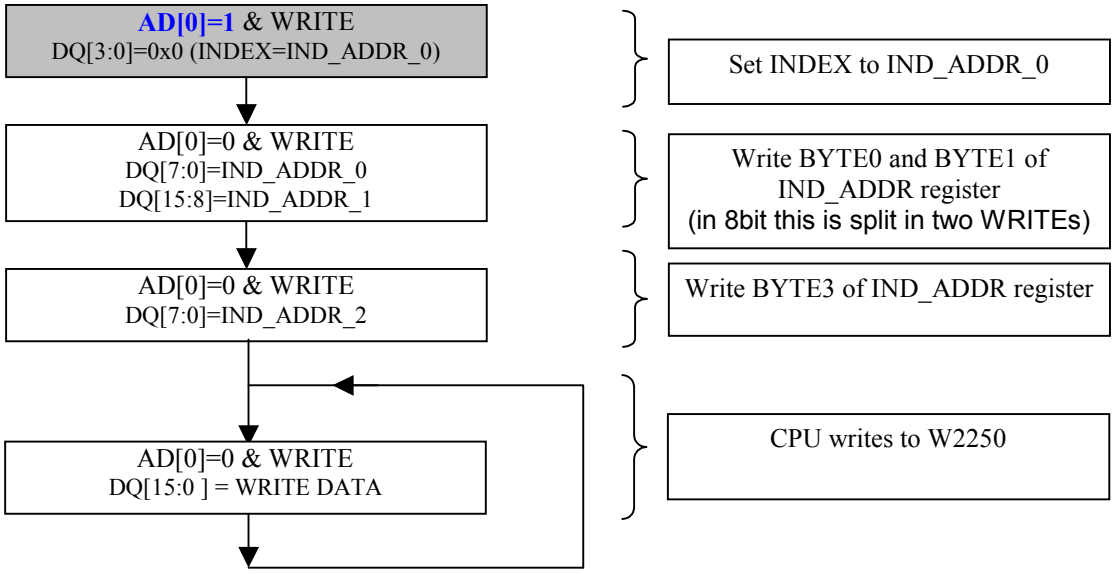

Figure 3-27 Write Protocol (ADDR = 1 bit, DATA = 8/16 bit) - Indirect Mode

3.6.7 Write Waveform - INDIRECT Mode

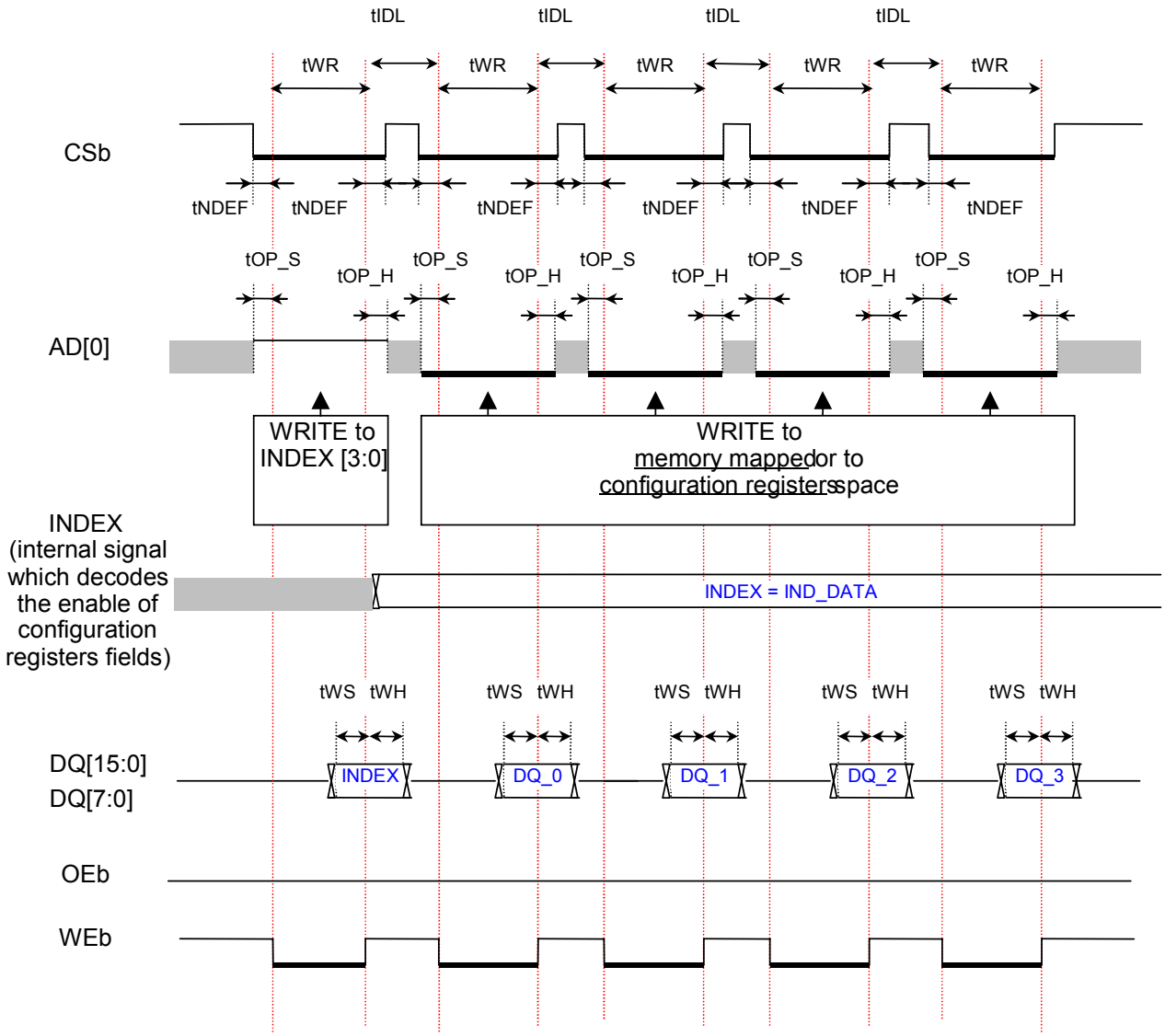

Figure 3-28 Write Protocol Waveform (ADDR = 1 bit, DATA = 8/16 bit) - Indirect Mode

Note: Interrupt is never asserted when writing to configuration space

3.6.8 Timings Parameters - INDIRECT Mode

Table 3-24 Timing Parameters - Indirect Mode

Timing symbol	Where it applies	Definition	Value [ns]				Comments
			min	typical		max	
				IDEN	PCS (DMAC)		
tIDL	WR/RD	IDLE period (CSb removed)	10	N/A	N/A		
tWR	WR	CSb active during a WRITE cycle	20	60	N/A		
tRD	RD	CSb active during a READ cycle	30	60	N/A		
tOP_S	WR/RD	setup timing for AD[0]	0	30	120		In respect to WEb/OEb falling edge.

Table 3-24 Timing Parameters - Indirect Mode (Continued)

Timing symbol	Where it applies	Definition	Value [ns]				Comments
			min	typical		max	
				IDEN	PCS (DMAC)		
tOP_H	WR/RD	hold timing for AD[0]	0	5	120		In respect to WEb/OEb rising edge.
tRA	RD	Read data access time from CSb falling edge		12	N/A	20	DQ pads are driven by W2250, as long as CSb is active and RWb is HIGH.
tRH	RD	Read data hold	0		N/A		In respect to OEb rising edge.
tWS	WR	Write data set-up	10	48	120		In respect to WEb rising edge.
tWH	WR	Write data hold	0	20	120		In respect to WEb rising edge.
tNDEF	RD/WR	NOT defined value					W2250 has no requirement for the skew between CS assertion/removal and OEb or WEb clock edges. tNDEF can have any positive value.

3.6.9 Rectangular Fill - INDIRECT Mode

Optionally, the HOST interface can increment the indirect address IND_ADDR_A, as described below, such as to generate a RECTANGLE shape, without using the drawing engine. This feature can be enabled either during WRITE or READ operations.

The advantage is that software can use a basic memfill or memcopy command that would execute continuous (burst) WRITE or READ to/from W2250 Frame Buffer, without being concerned about controlling the physical address, in order to obtain a RECTANGLE shape.

Figure 3-29 describes how PITCH[9:2] and LENGTH[8:1] values are applied when IND_ADDR_A is incremented during RECTANGLE mode.

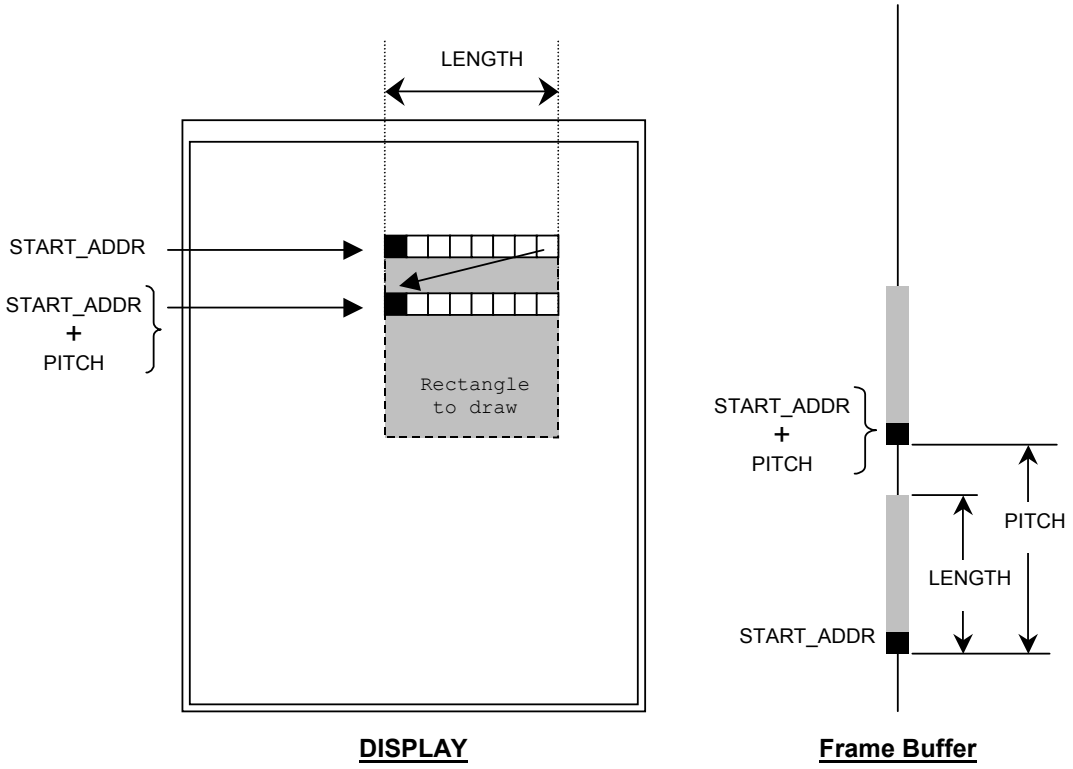

Figure 3-29 Rectangular Fill - Indirect Mode

Notes:

- PITCH is dword aligned and LENGTH is word aligned
- PITCH and LENGTH can be read back from Memory Mapped register CIF_PF_CNTL[15:0]
- Minimum supported LENGTH is one dword
- START_ADDR can be aligned either to BYTE_0 or to BYTE_2

3.6.9.1 Using RECTANGLE_ACCESS

In order to use RECTANGLE_ACCESS, the software needs to perform the following:

1. Set configuration register bit UPDATE_RECTANGLE_PARAM = 1.
2. Set INDEX to IND_DATA_A.
3. Write 16bit to IND_DATA_A, first/lower byte is assigned by hardware to PITCH value and the second/upper byte is assigned to the LENGTH value. Hardware will clear UPDATE_RECTANGLE_PARAM bit, when second byte is written.
4. Set RECTANGLE_ACCESS configuration register bit.
5. Set INDEX to IND_ADDR_A_0.
6. Update IND_ADDR_A, in order to specify START_ADDRESS of the rectangle.
7. Execute memfill or memcopy commands to/from IND_DATA_A, that consist in continuous data transfer, without any address control requirement.
8. Update IND_ADDR_A for the next operation. RECTANGLE_ACCESS bit is cleared by hardware during this step.

Notes:

- The RECTANGLE_ACCESS bit can be cleared by software or automatically by hardware. Hardware will clear this bit when the second update of IND_ADDR_A occurs (after RECTANGLE_ACCESS is set, step #5 performs the first update, and step #8 performs the second update).
- Optionally, the INTERRUPT will reset the RECTANGLE_ACCESS bit and UPDATE_RECTANGLE_PARAM bit if CIF_CNTL_DBG/EN_INT_TO_RST_PENDING_ACCESS = 1.
- EN_SUSTAIN_DWORD_ADDR_A has precedence over RECTANGLE_ACCESS.
- Rectangle parameters can be updated once, then many rectangles can be drawn using the same parameters.
- For RECTANGLE WRITE (but not available for READ), step #3 can be included in step #7.

3.6.9.2 RECTANGLE READ without Pre-fetch

- Poll on dword boundary of IND_ADDR_A.
- If NOT a multiple of dwords, incorporate a LENGTH counter in software, and poll after each LENGTH.

Example 1:

- PITCH=0x8, LENGTH = 0x5 (not a multiple of dwords)
- set address 0x100
- poll, and RD32 0x100, 0x102 (LENGTH counter = 0x2)
- poll, and RD32 0x104, 0x106 (LENGTH counter = 0x4)
- poll, and RD16 0x108 (LENGTH counter =0x5)
- poll, and RD32 0x120, 0x122 (LENGTH counter = 0x2)
- etc...

Example 2:

PITCH=0x8, LENGTH = 0x5 (not a multiple of dwords)

set address 0x102

poll, and RD16 0x102 (LENGHT counter = 0x1)

poll, and RD32 0x104, 0x106 (LENGTH counter = 0x3)

poll, and RD32 0x108, 0x10a(LENGTH counter =0x5)

poll, and RD16 0x122 (LENGHT counter = 0x1)

etc...

3.6.9.3 Rectangle READ with pre-fetch :

- Poll first dword only.
- Even if a multiple of dwords, incorporate a LENGTH counter in software, and always poll after each LENGTH.

Example :

- PITCH=0x8, LENGTH = 0x5 (not a multiple of dwords)
- Set address 0x100
- Poll, and RD32 0x100, 0x102 (LENGHT counter = 0x2)
- Without polling, RD32 0x104, 0x106 (LENGTH counter = 0x4)
- Without polling, RD16 0x108 (LENGTH counter =0x5)
- Poll, and RD32 0x120, 0x122 (LENGHT counter = 0x2)
- etc...

3.6.10 CPU Interface Read/Write - DIRECT Mode

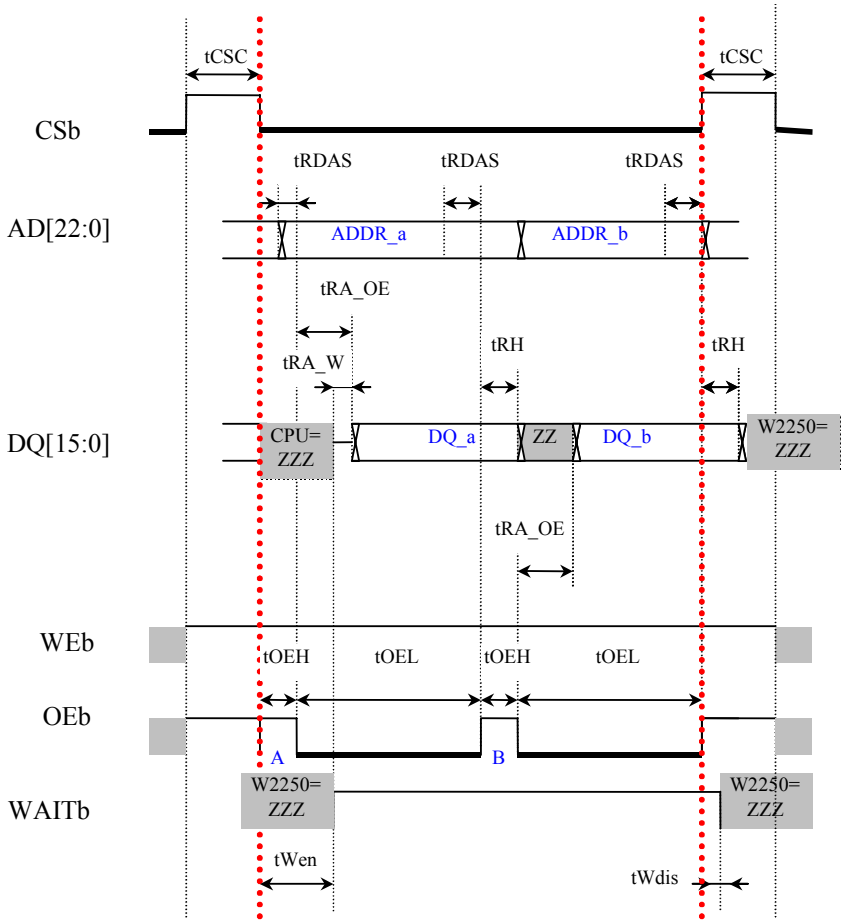

Figure 3-30 Read Timing - Direct Mode

Note 1: Optional CSb can be removed after each OE rising edge. When CSb is inactive, WAITb is tristated.

Note 2: If CSb is removed **before** OE rising edge, then tRDAS and tRH will apply to CSb rising edge instead.

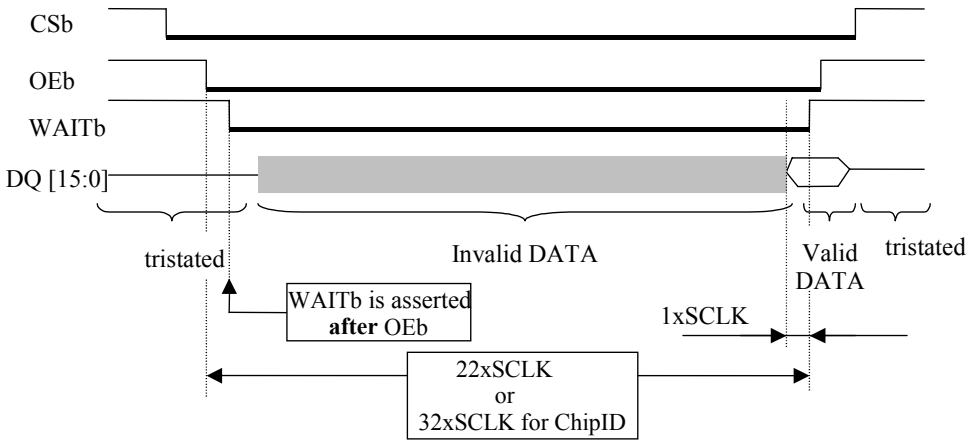

Figure 3-31 Read from Memory-Mapped Space with EARLY WAITb Disabled - Direct Mode

Note 1: GPIO13 = pull down; MR = pull down

Note 2: Behavior WAITb is asserted about 8ns after OEb.
Advantage: RWb signal is NOT needed.
Disadvantage: CPU could sample WAITn state before OEb is asserted, considering that READ data is valid before the fetching process is completed. Therefore, this could lead to READ errors.

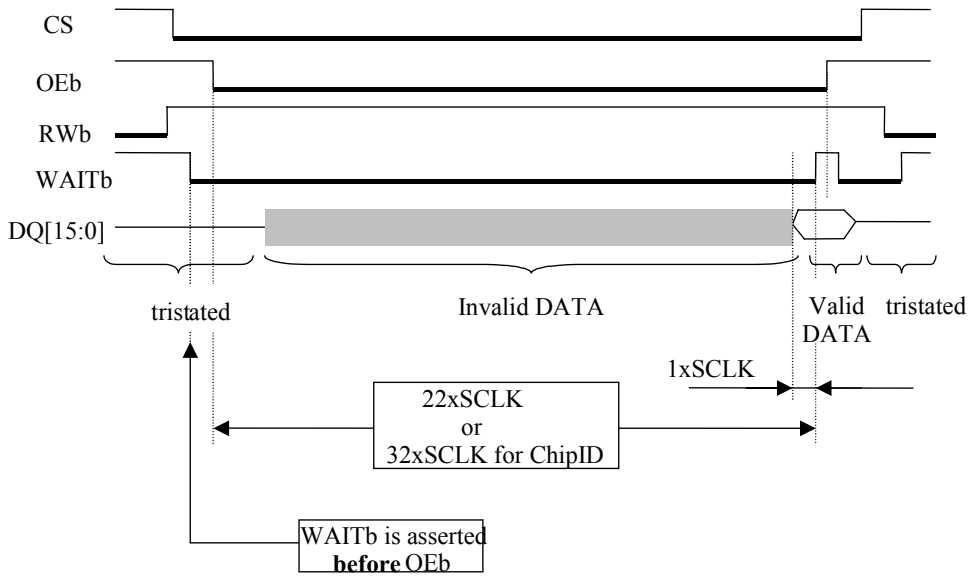

Figure 3-32 Read from Memory-Mapped Space with EARLY WAITb Enabled - Direct Mode

Note 1: GPIO13 = pull up; MR = driven by RWb signal from CPU

Note 2: Reading ChipID happens before ATI driver initializes the W2250 registers. Therefore, WAITb is asserted for 32xSCLK instead of 22xSCLK, because the W2250 registers are not optimized until after ChipID is read.

Note 3: Behavior: WAITb is asserted about 8ns after CSb.
Advantage: OEb and WEb speeds can be increased.
Disadvantage: RWb signal is required by the protocol.

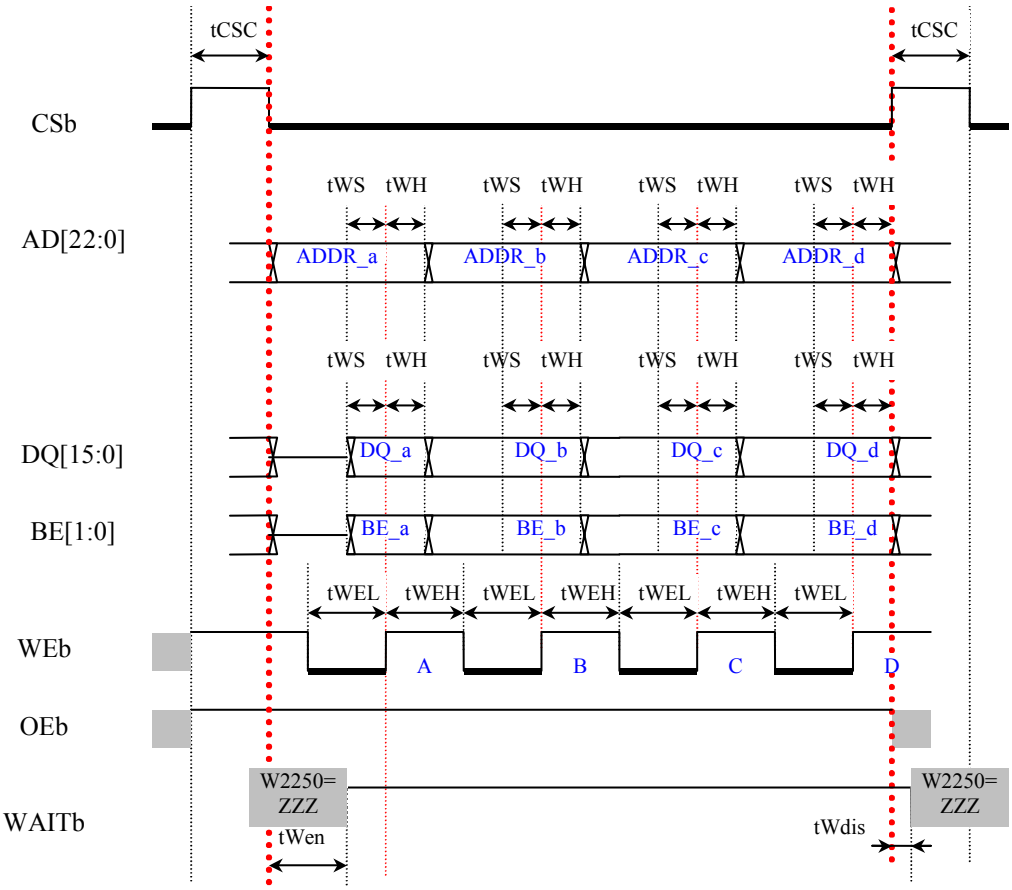

Figure 3-33 Write Timing - Direct Mode

Note: Optional CSb can be removed after each WE rising edge. When CSb is inactive, WAITb is tristated.

3.6.11 Timing Parameters - DIRECT Mode

Table 3-25 Timing Parameters - Direct Mode

Parameter	Application	Definition	Value [ns]		Comments
			min	max	
tCSC	RD & WR	CS inactive cycle	10		
tWen	RD & WR	WAIT output pad enable		17	In respect to CS falling edge
tWdis	RD & WR	WAIT output pad tristate	0	10	WAITb pad is tristated after CS is removed
tRA_W	RD	Read data access time from WAITb rising edge		5	Read data is driven on the bus **in the same time** WAITb is deasserted.
tRA_OE	RD	Read data access time from OEb falling edge		20	During this time W2250 tristates DQ bus
tRH	RD	Read data hold	0		
tOEH	RD	OE strobe high	8		
tOEL	RD	OE strobe low	30		During WAITb = HIGH
tRDAS	RD	Read address setup	20		
tRDAH	RD	Read address hold	0		In respect to OE rising edge
tWEH	WR	WE strobe high	10		
tWEL	WR	WE strobe low	10		During WAITb = HIGH
tWS	WR	Write data/address set-up	10		In respect to WEb rising edge
tWH	WR	Write data/address hold	0		In respect to WEb rising edge

3.7 SPI Interface

3.7.1 QSPI

QSPI (Queued Serial Port Interface) is a simple four-bit serial port. The QSPI protocol consists of four I/Os:

- SCK - the clock for the interface, provided by the CPU
- SSb - the active-low select signal for the QSPI interface, also driven by the CPU
- MOSI - an acronym for "master out, slave in", this is the connection for commands and data passing from the CPU to W2250
- MISO - an acronym for "master in, slave out", this is the connection for data passing from W2250 to the CPU

The QSPI block in W2250 supports two types of requests: Type 0 and Type 1.

3.7.1.1 Type 0 Requests

Type 0 requests are in a traditional SPI format, which is illustrated in *Figure 3-34 Format of Type 0 Request on page 3-50*. When the CPU transmits a header, the bits are expected to arrive from bit 31 through to bit 0 (in the case of MSB-first transmission) or from bit 0 through bit 31 (in the case of LSB-first transmission). In MSB-first mode, if the MSB of a command received by QSPI is a '0', this indicates that a Type 0 request is being transmitted. (In LSB-first mode, only Type 0 requests are supported so it is assumed all requests are Type 0.) The next two bits represent the op code, which determines whether the request is for a read or a write operation:

Table 3-26 Op Codes

Op Code	Definition
00	Read from address = byte address, number of bytes = length in bytes, including read latency
01	Write to address = byte address, number of bytes = length in bytes, including write latency
10	Write to Type 1 table (Type 1 requests only)
11	Reserved

Bits 28:24 contain the number of bytes to be read/written, including the read/write latency. There are only five bits for this field because a maximum of 32 bytes can be transmitted by a single request, including header and dummy bytes. The last 24 bits are the byte address, which is the starting address for the read/write operation. W2250 will auto-increment the address for each successive byte to be read/written (except when accessing SDIO, in which case the starting address is held constant). Type 0 requests are ideal for multi-dword data transfers, since each Type 0 command consumes at least 5 bytes of overhead (4 bytes of header plus one byte of latency).

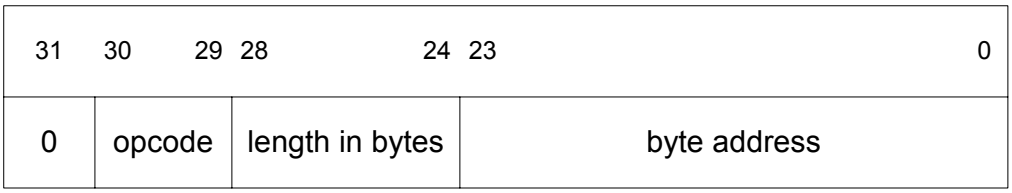

Figure 3-34 Format of Type 0 Request

While Type 0 request headers may be transmitted most or least significant bit first, the header is always treated as a complete entity and is not broken down into bytes. In other words, if transmitting MSB-first, the sequence of bits would be from 31 to 0, in that order. LSB-first would be 0 to 31, again in that order.

The data that follows the Type 0 request header is always sent least significant bit first, to correspond with the starting byte address given in the header. The MSB or LSB-first setting only determines the order of the bits within each data byte; the order of the bytes never changes.

3.7.1.2 Type 1 Requests

The format of a Type 1 request is shown in *Figure 3-35 Format of Type 1 Request on page 3-51*. Type 1 requests are shorter and, as such, are intended to reduce the overhead for shorter or common transfers. (Note: Type 1 requests also support special operations of up to 512 bytes.) Type 1 requests are identified by a '1' in the MSB. Note that Type 1 requests can only be used when in MSB-first mode. If QSPI is configured in LSB-first mode, then requests are restricted to Type 0.

The op codes are the same as the ones described for Type 0 requests, with the exception of "10", which is only supported for Type 1 requests (see *Table 3-26, "Op Codes," on page 50*). Instead of specifying a starting address and a transfer length, however, a client ID is used. The client ID is an index into a lookup table of commonly accessed data. Each item in the table consists of a starting address (Starting Address), a length in bytes (Transaction Length), and an auto-increment flag (AI), as shown in *Figure 3-36 Format of Table of Type 1 Requests on page 3-51*. The table of client IDs contains up to 8 entries and is fully programmable through QSPI Type 1 writes.

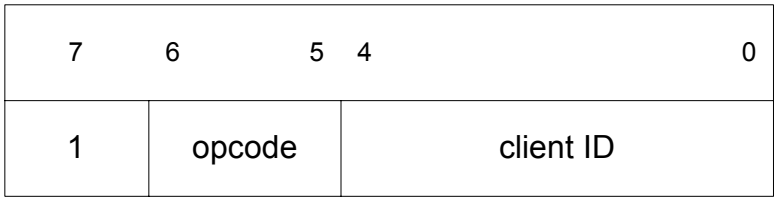

Figure 3-35 Format of Type 1 Request

Client ID	Starting Address[33:10]	Transaction Length[9:1]	AI
000			
001			
• • •	• • •	• • •	• • •
111			

Figure 3-36 Format of Table of Type 1 Requests

If the autoincrement flag is set, the SPI will automatically update the starting address for that client after each transaction requested for that client. For example, assume Client 0 is address xxxx within the framebuffer, and the associated length is 4. The first time Client 0 is accessed, the starting address within the table will be automatically updated to xxxx + 4 when the transaction is complete. The second time Client 0 is accessed, the starting address will be xxxx + 4, and it will be updated to xxxx + 8 upon completion of the transfer. In order to reset the starting address, the client ID must be reprogrammed into the table.

Op code "10" is reserved for programming the table of Type 1 client IDs. Each client ID can be programmed individually. When QSPI sees an op code of "10" in a Type 1 request, it will process the Client ID and treat the next 5 bytes as the data for the Type 1 lookup table. The entire transaction is illustrated in *Figure 3-37 Programming an entry in the Type 1 lookup table on page 3-52*. When programming the table, the bytes are sent least significant byte first. In other words, Byte 0 is bits [7:0] in the Type 1 lookup table. Byte 1 is bits [15:8], etc.

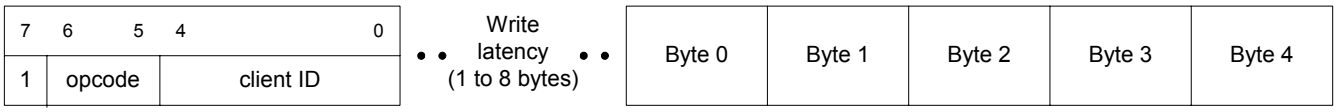

Figure 3-37 Programming an entry in the Type 1 lookup table

Note that the Type 1 lookup table is 34 bits wide (24 bits of address, 9 bits of length, and 1 bit for autoincrement), which does not fall on a byte boundary. Therefore, the 6 most significant bits of Byte 4 are discarded by the QSPI slave.

Direct reading of the table entries is not supported.

3.7.1.3 QSPI/CPU Handshaking

The read and write operation for a Type 0 request can be seen in *Figure 3-38 Data Transfer Scheme for Type 0 Requests on page 3-53*. The W2250 QSPI is a slave device; therefore, all operations are initiated by the CPU. The request begins with the CPU starting the clock (SCK) and asserting the QSPI select (SSb). On the next clock edge (either rising edge or falling edge - see QSPI control registers for details), the first bit of the request is transmitted from the CPU. Data can be transmitted either MSB first or LSB first, but Type 1 requests are only valid in the MSB-first configuration.

QSPI shifts in the request 8 bits at a time and interprets the command. The starting address and number of bytes are passed to the CIF as soon as they are decoded. If the operation is a write, a programmable number of dummy bytes (1-8 bytes) are sent by the CPU to give QSPI time to process the command and prepare to write. After the dummy bytes, the data is shifted in, packed into dwords, and passed to the CIF with byte enables. If the operation is a read, it is expected that it will take some time for W2250 to return the requested data. Therefore, a programmable delay (1-8 bytes) is inserted before the readback data arrives. Once the readback data is available, the CPU continues to transmit dummy information while QSPI shifts out the data. The CPU transmits as much dummy data as it is expecting real data, i.e. when all the real data has been sent, the dummy data will have ceased transmitting as well. At that time, the CPU de-asserts SSb for at least one cycle and stops the clock if no other operations are pending. If SSb is de-asserted for any reason in the midst of a transaction, QSPI will interpret that as an incomplete transaction and will automatically reset itself. The next time SSb is de-asserted, QSPI will interpret that as the beginning of a new command header.

Type 1 requests are similar, as illustrated in *Figure 3-39 Data Transfer Scheme for Type 1 Requests on page 3-54*. The CPU first starts the clock and asserts the select. On the next clock edge, the first bit of the request is transmitted from the CPU. Once again, note that Type 1 requests can only be used in MSB-first mode. The request is identified as Type 1 by the '1' most significant position. QSPI shifts in all 8 bits of the request and interprets the command. This includes indexing the client ID into a lookup table to determine the starting address and number of bytes to transfer. The rest of the transaction occurs in the same way that a Type 0 request would.

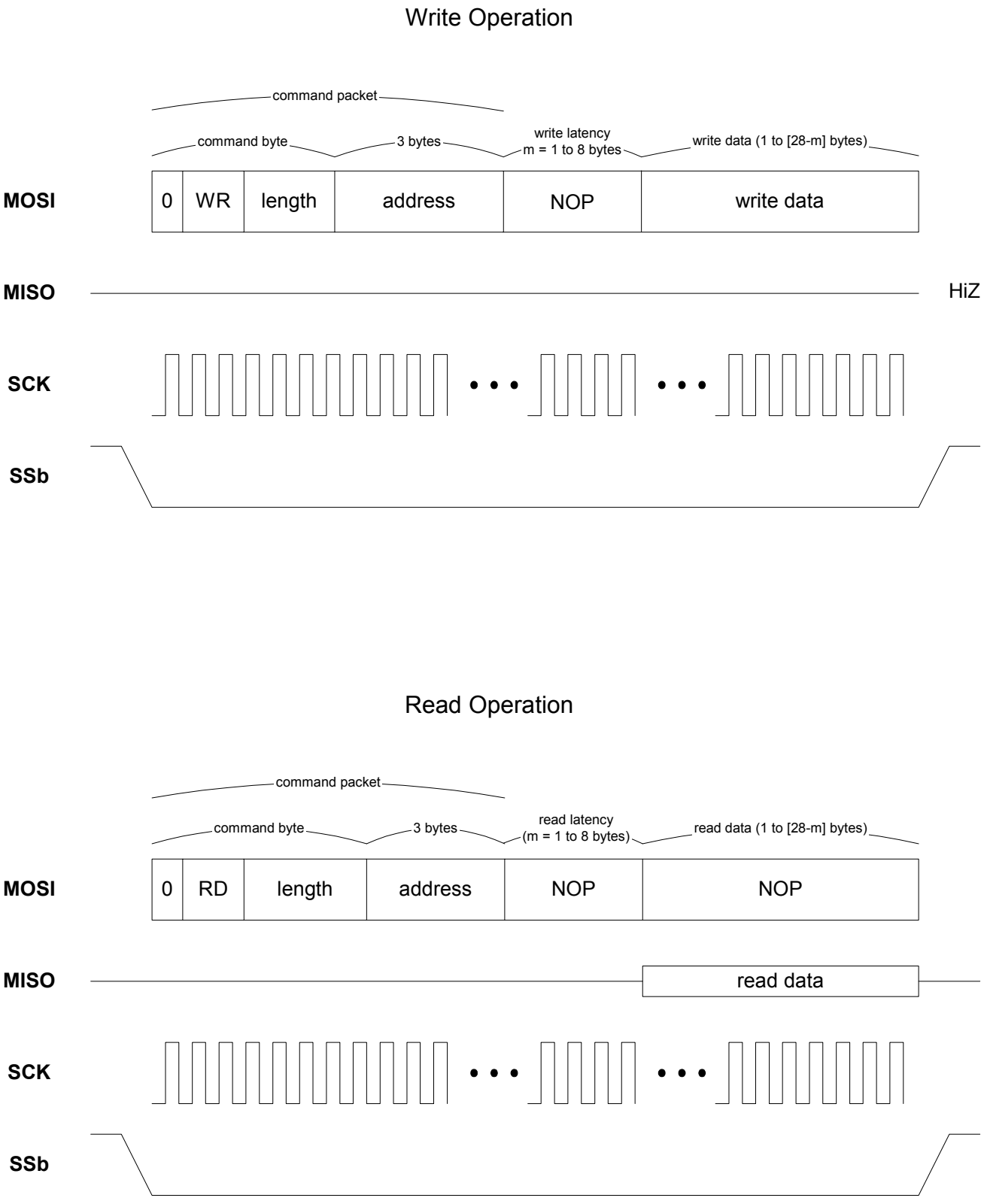

Figure 3-38 Data Transfer Scheme for Type 0 Requests

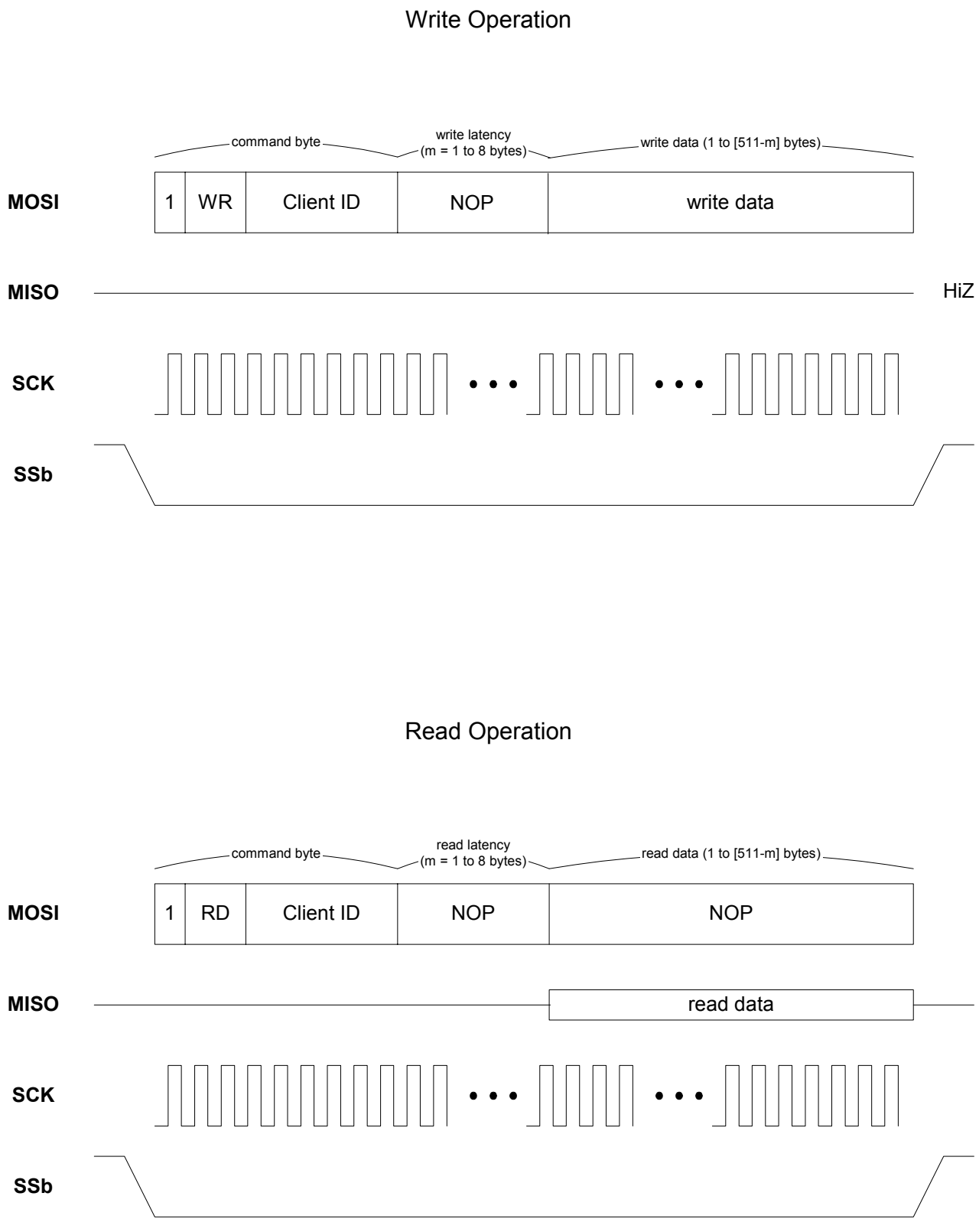

Figure 3-39 Data Transfer Scheme for Type 1 Requests

3.7.1.4 Error Handling

The QSPI logic is able to detect several error conditions and recover from them automatically. The four error conditions that can be detected are:

- format error
- first read failure
- write failure
- incorrect length

A **format error** is an error in the command header. QSPI will flag an error if a Type 0 request is made to write the Type 1 table (i.e. op code of 2'b10), if a Type 1 request attempts to access an undefined row in the table (i.e. client ID > 4'b0111), or if a request is made to perform a transaction with a length less than 1 byte. Format errors are handled by generating an asynchronous reset pulse on the next rising edge of iSSb (i.e. at the end of the transaction) so that QSPI is in the proper state to begin the next transaction. In addition, an interrupt is generated.

A **first read failure** occurs when the CIF does not return read data quickly enough for QSPI to serialize it and send it to the CPU. Note that if the second or subsequent read fails, the error is not flagged. Triggering this condition will also generate an interrupt.

A **write failure** is produced if the CIF does not respond to the write RTS before the RTS for the second dword is asserted. It is assumed that this condition will never arise because if it does, the chip will most likely be in an overall bad state. Although QSPI can flag this error, it cannot recover gracefully from it, so the interface will cease to function properly. This condition generates an interrupt.

Finally, if too many or too few clock transitions occur while iSSb is asserted, QSPI will detect the failure and reset itself on the next rising edge of iSSb. There is no guarantee that the mis-aligned transaction will be successful, but subsequent transactions (of the correct length) will be processed correctly. This condition does not generate an interrupt.

3.7.1.5 Clocks

The majority of the QSPI block will run on the clock provided by the SPI master, SCK. The target frequency is a maximum of 14 MHz. The remainder of QSPI (e.g. its interface with the CIF) will run on a branch of SCLK, QSPI_SCLK, in the range of 5.2 - 50 MHz. This SCLK branch is not dynamic, i.e. it will always be on, unless the chip is in suspend mode. There are two ways that QSPI can be used to bring W2250 out of suspend mode:

- by issuing dummy writes through the QSPI interface
- by issuing dummy writes or dummy reads (or reads from config space) through the QSPI interface

A register bit in the QSPI control registers is used to make this selection.

When waking up from suspend, the software must use QSPI to poll the STATUS register in config space. Once W2250 is no longer in suspend, normal operation may continue.

If QSPI transactions are to be run when SCLK is in slow mode (i.e. 5.2 MHz), iSCK should be reduced accordingly or QSPI should be limited to single dword requests. In addition, QSPI cannot tolerate the SCLK stoppages associated with switching from slow clock mode to normal/fast clock mode and vice versa. Therefore, software must ensure that mode switches of this type do not occur during a QSPI transaction.

3.7.2 Interface Timing

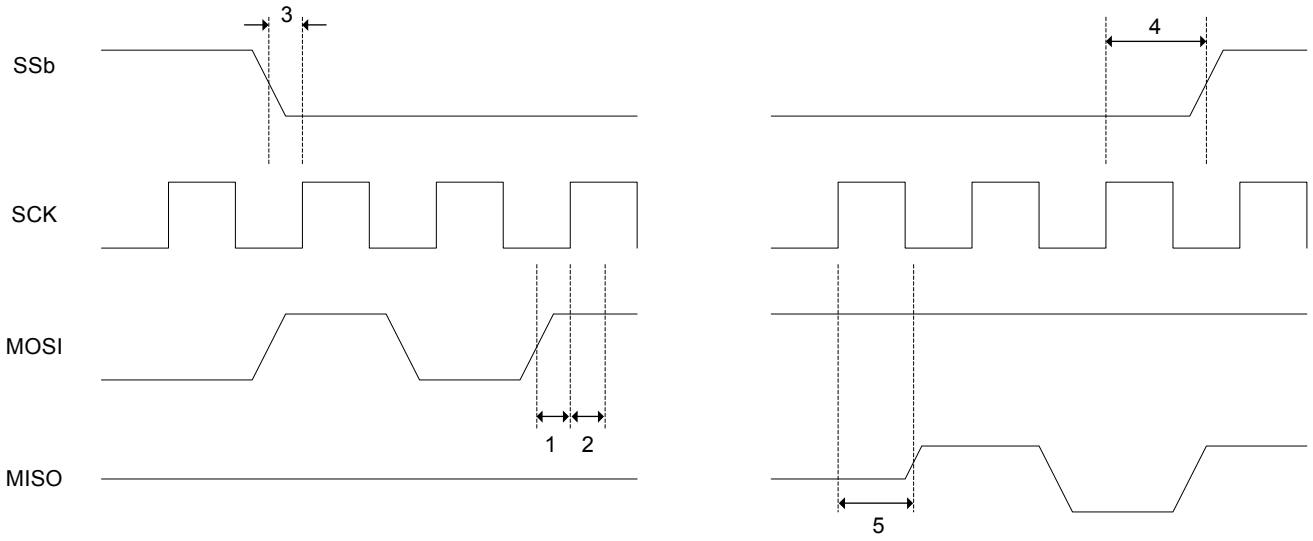

Figure 3-40 QSPI I/O AC Parameters

Table 3-27 QSPI I/O AC Parameters

Reference To *Figure 3-40*	Parameter	Minimum Value	Typical Value	Maximum Value	Units	Notes
1	Setup of MOSI vs. SCK	5			ns	
2	Hold of MOSI vs. SCK	8			ns	
3	Setup of SSb vs. SCK	6			ns	
4	Hold of SSb vs. SCK	13			ns	
5	Access Time			40	ns	

3.8 CLI Interface

W2250 supports a 5-wire serial interface. This interface only allows writes to 96x64 resolution CLI. For this interface, 4 pins are used to transfer data. 4 pins transfer CLI_CSB, CLI_CMD, CLI_CLK and CLI_DOUT. There is one more pin which sends the CLI_RESETB signal to the CLI.

There is an option to let CLI_CSB out from GPIO7. In this case, GPIO7 can be used as CLI_CSB. The default setting for cli chip select (CLI_CSB) is through the CLI_CSB pin. Switching to send out CSB either through CLI_CSB or GPIO7 can only be done after the transaction. Do NOT effect this switch during a transaction.

CLI_DOUT is sent most significant bit first, latched on the rising edge of CLI_CLK. CLI_DOUT comprises display data and display command. When display data is sent, CLI_CMD is high. CLI_CMD is low when command data is sent.

For proper operation, the SCLK:CLI_CLK ratio must be less than or equal to 8.

When the CLI_RESETB is asserted (active low), the display finishes the present byte transaction and then sends the reset to the CLI.

Display registers control the display command/data indication. The CPU writes the display command and data in local memory at a certain offset. The registers in the display block have the length of the display command/data. The offset register consists of the offset address of the display command/data stored by CPU. This register also consists of a trigger bit. When this bit is programmed, it triggers the display and starts making requests to the memory controller (*Figure 3-41 Functional Block Diagram of CLI Interface on page 3-58*). This trigger bit is post write trigger pulse, hence it is 1 clock wide.

After the trigger pulse, the total data and command length is counted. The display starts making request to the memory for data until it counts to command and data length. The command and data length are specified in bytes. If there are no commands, the command length register is programmed zero. If there is no data, then the data length register is programmed zero. The maximum command length is 255(bytes) and the maximum data length is also 255(bytes).

The CLI interface has the least priority over graphics and video.

When display starts sending data out from the 4-wire interface, the active signal goes high. During active high, do NOT trigger the CLI again. After the end of each transaction, the interrupt goes high indicating all the display data (command and data) programmed has been sent and the interface is ready for another transaction. Post write trigger pulse interrupt clears this interrupt. See *Figure 3-42 Functional Timing Diagram for CLI Interface on page 3-59* for more information.

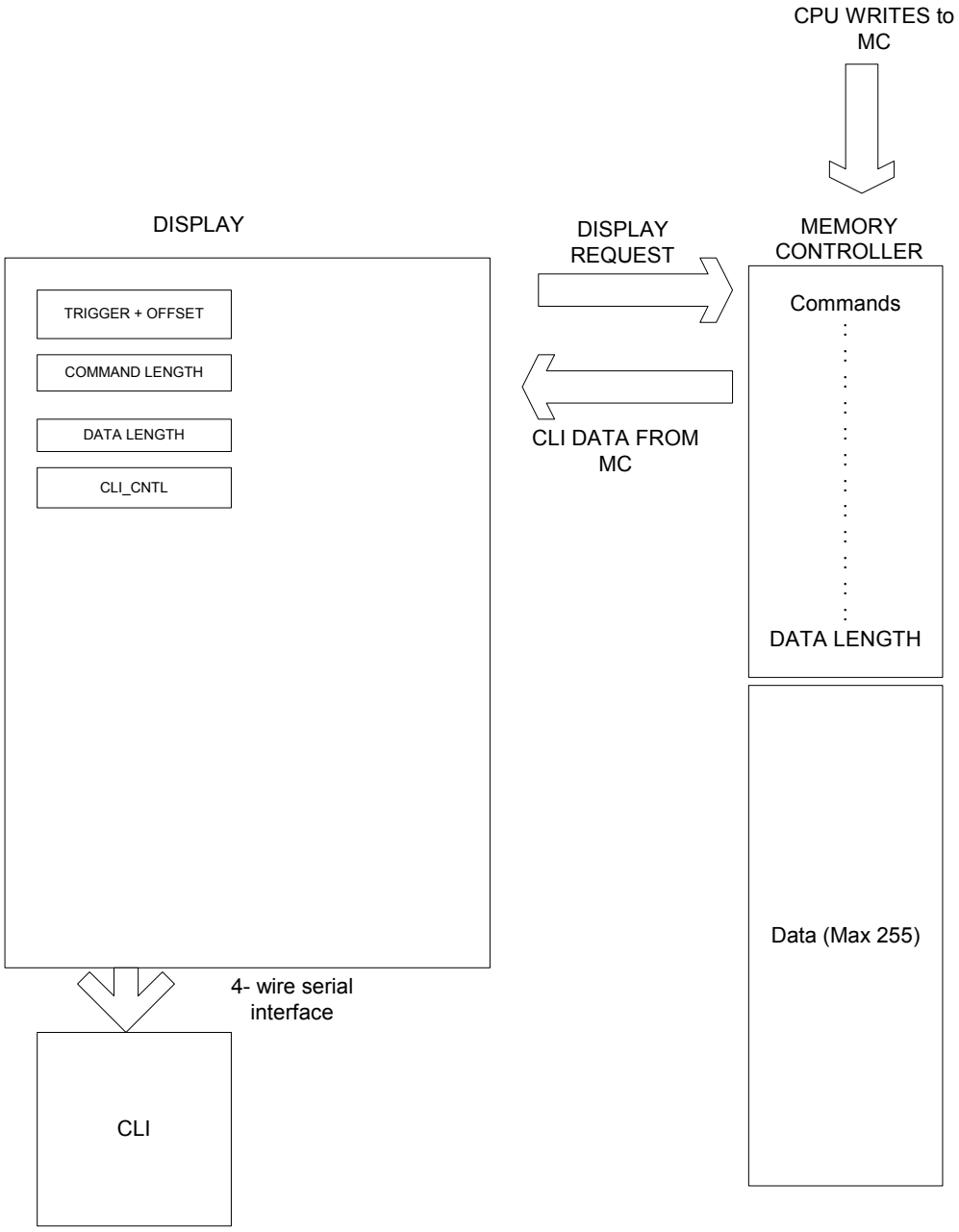

Figure 3-41 Functional Block Diagram of CLI Interface

The data coming in from memory is sent serially. This data is sent at the positive edge of every CLI_CLK to CLI. The CLI_CSB goes low when the data is serially sent every positive serial clock to the interface. The CLI_CSB goes high again when it finishes sending all the command data and the display data programmed. When the CLI_CSB goes high, it stops the transfer of data.

A register bit is used to start conversion of data from parallel to serial.

The data sent out to CLI respects the byte boundary. The command length and the data length programmed represents the number of bytes sent.

EXAMPLE 1:

Command length = 1

Data Length = 4

In this case, the total bits to be sent are {8-bits (command) + 4x8 = 32-bits (data)} = 40- bits (5 bytes)

EXAMPLE 2

Command Length = 2

Data Length = 5

In this case, the total bits to be sent are {(8x2) = 16 bits (command) + (8x5) = 40 -bits (data)} = 56 -bits (7 bytes)

3.8.1 Timing Diagram

The following is the functional timing diagram for the CLI interface. After all the display commands are sent out, the CLI_CMD signal becomes high indicating display data.

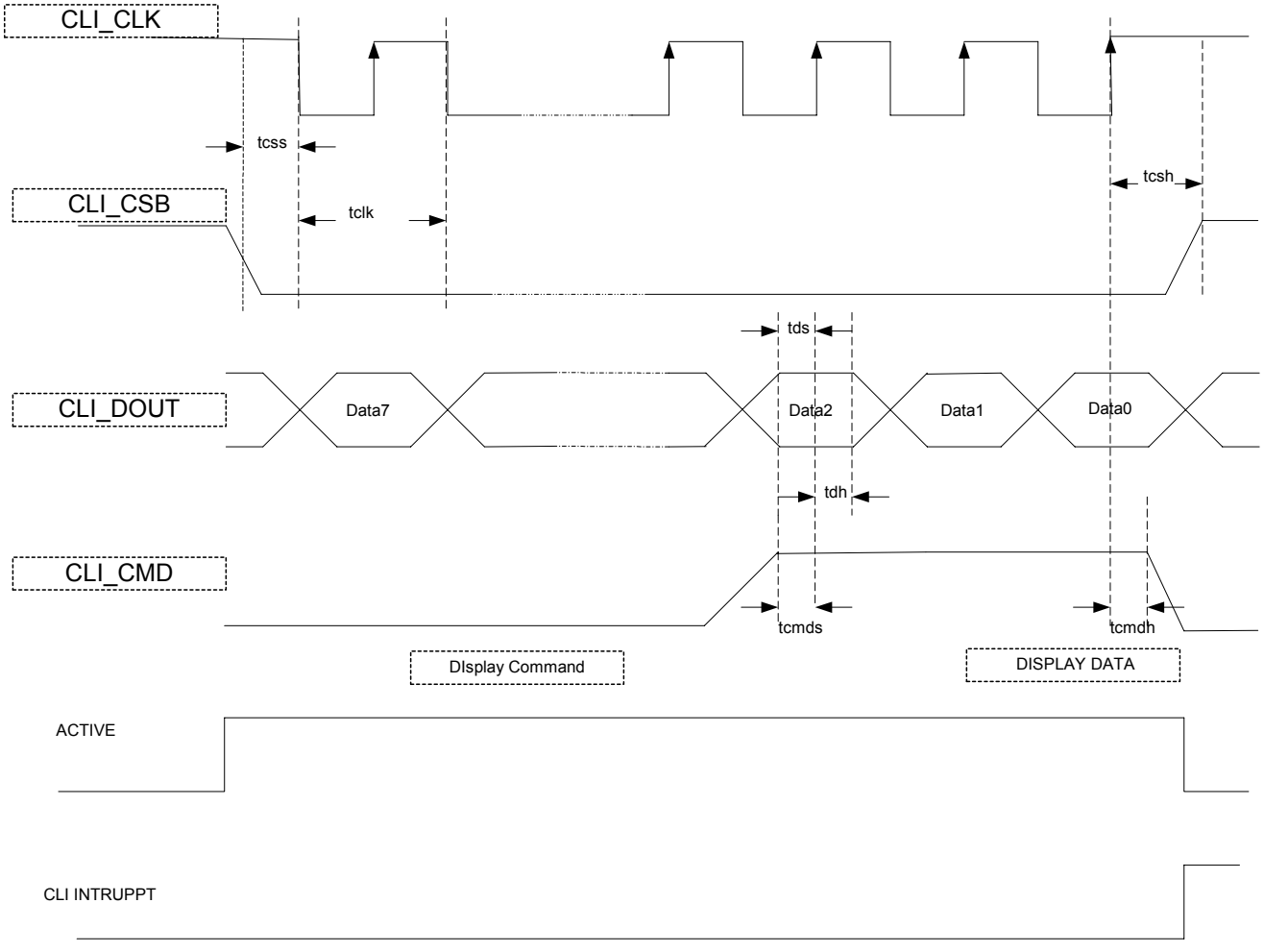

Figure 3-42 Functional Timing Diagram for CLI Interface

Table 3-28 Timing for CLI Interface

Parameter	Symbol	Min	Max	Units
CLI_DOUT	tOD	2	5	ns
CLI_CSB	tOD	2	8	ns

3.9 The Oscillator

3.9.1 Clock Distribution for W2250

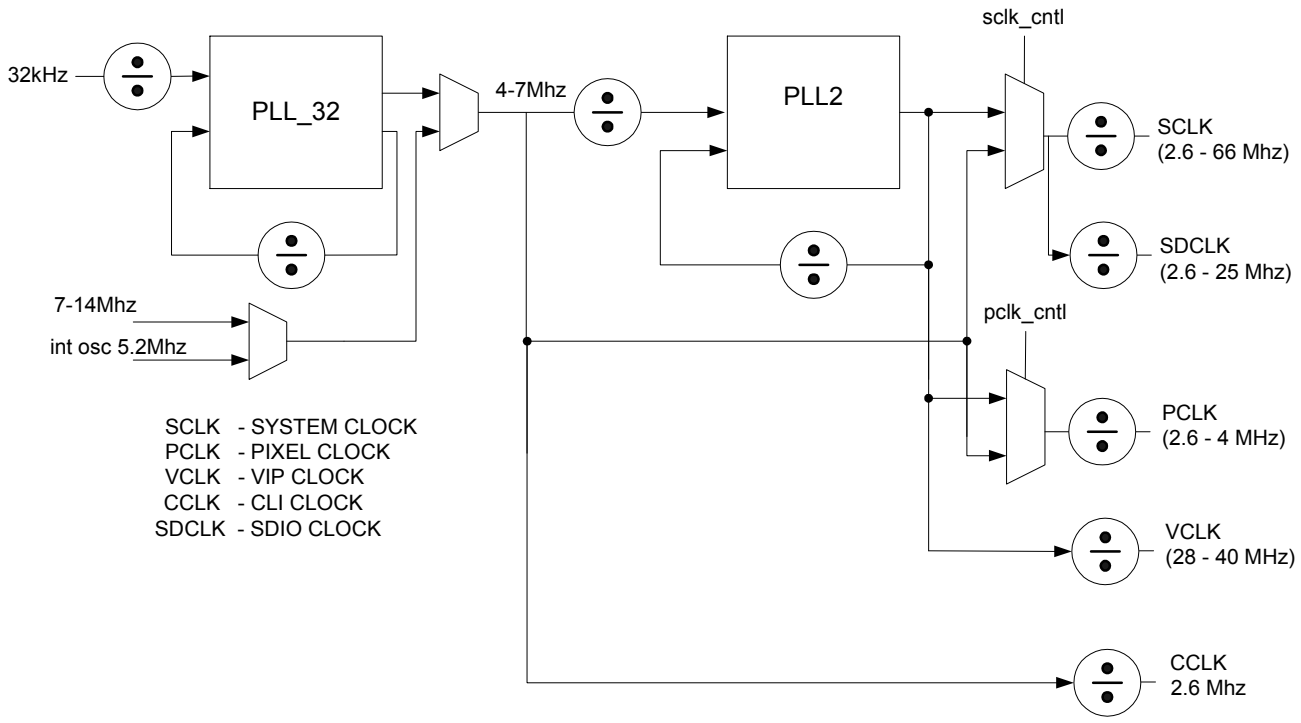

Figure 3-43 Clock Distribution for W2250

3.9.2 Oscillator Modes

The oscillator may have up to four different modes. Two modes allow the selection of an external oscillator source. Device configuration bits are used to select the oscillator mode.

The oscillator modes are:

- LPC Low Frequency (Power) Crystal- 14 Mhz
- EXTRC External Resistor/Capacitor
- Input clock from clock source 32kHz
- Input clock from clock source (7-14 MHz)

These oscillator modes allow a single device type the flexibility to fit applications with different oscillator requirements. The RC oscillator mode saves system cost while the LPC crystal mode saves power.

LPC Low Frequency (Power) Crystal- 14 Mhz Mode

The LPC mode uses a 14MHz crystal and a PLL for producing the 28..66MHz clock. The other PLL (PLL_32) is not active in this configuration. The PLL current is less than 0.5mA.

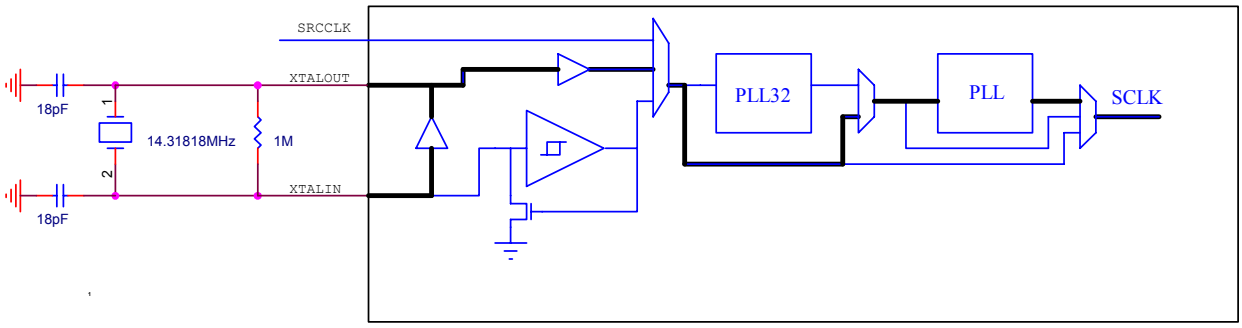

Figure 3-44 LPC Crystal Mode

EXTRC External Resistor/Capacitor Mode

The EXTRC is the least expensive solution for device oscillation (only an external resistor and capacitor are required). It also has the most variation in time-base.

The RC oscillator frequency is a function of: the supply voltage, the resistor (REXT) and capacitor (CEXT) values, and the operating temperature. In addition, the oscillator frequency will vary from unit to unit due to normal process parameter variation. Furthermore, the difference in lead frame capacitance between package types will also affect the oscillation frequency, especially for low CEXT values. The user also needs to take into account the variation due to the tolerance of the external REXT and CEXT components used. The resulting 5.2MHz follow the same path as the crystal oscillator signal and uses a second PLL to supply 20..60MHz for core. The PLL_32 is not active. For 5.2MHz output frequency at Vdd=1.2V, the suggested RC value for the oscillator is R=7.0K and C=10pF.

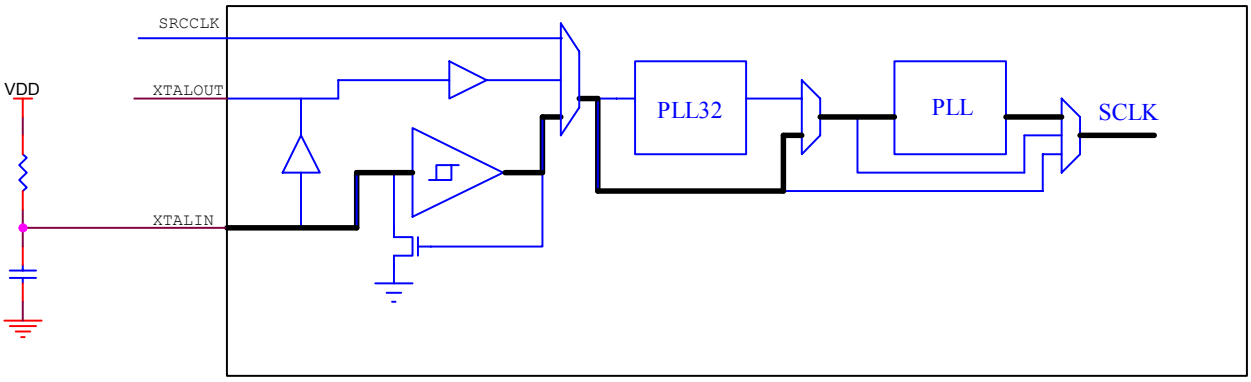

Figure 3-45 EXTRC External Resistor/Capacitor Mode

Input Clock from Clock Source 32kHz Mode

If the system already has a 32kHz source, then this mode is the best solution in terms of price/power. In this configuration, the 32kHz signal drives the first PLL_32 to reach the 4-7MHz reference for the second PLL. The PLL-32 current is in the range 0.5-1mA.

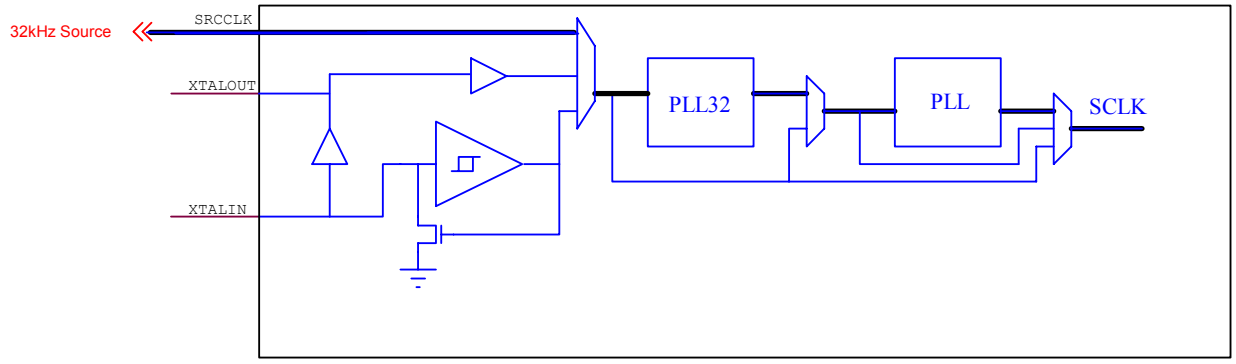

Figure 3-46 Input Clock from Clock Source 32kHz Mode

Input Clock from Clock Source (7-14 MHz) Mode

In this mode, the input clock 7-14Mhz bypasses the internal oscillator and PLL_32 and drives the second PLL directly.

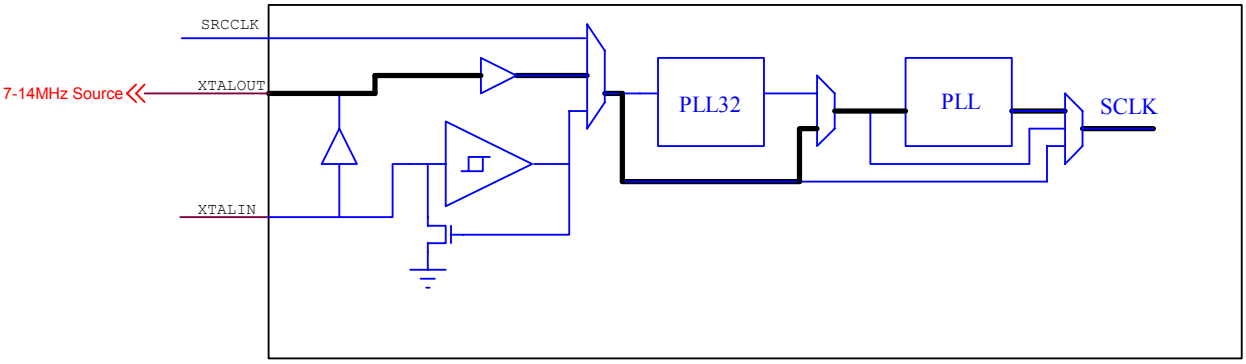

Figure 3-47 Input Clock from Clock Source (7-14 MHz) Mode

The following table lists the clock modes and associated input pins and power rails:

Table 3-29 Clock Pin/Power Specification

Clock Mode	Input Pin	Powered On
32kHz	SRCCLK	VDDR1
7-14MHz	XTALIN	PVDD
	XTALOUT	PVDD

3.10 Drawing Modes in Acceleration-Operation Mode

3.10.1 Programmable I/O (PIO) Mode

In this mode, the host application programs the GUI engine by writing directly to the W2250's memory-mapped registers. The registers are written through one of the W2250's two register apertures over the bus interface. The register writes are queued in the W2250's internal Command FIFO buffer as register-datum pairs. These Command FIFO buffer entries are processed by the GUI engine to draw into the frame buffer.

3.10.2 Concurrent Command Execution (CCE) Programming Mode

In this mode, the host sends commands to the W2250 in the form of command packets. A command packet is a data block that consists of a header followed by a variable size data body. Within the W2250, the packets are queued in the CCE FIFO buffer. A micro controller processes the packets, produces the conventional register data, and feeds this data to the Command FIFO buffer. The Command FIFO buffer data is processed (as it is in PIO mode) to render into the frame buffer.

The host application transfers packets to the CCE FIFO buffer by queuing them in frame buffers. Streaming packets in this manner enables significant concurrency between the host and the W2250. In addition, there are several predefined single-purpose packets that greatly simplify the programming of common drawing operations.

The W2250 uses the ring buffer mechanisms for bus-mastering packets to the CCE FIFO buffer.

3.10.3 Ring Buffer

The ring buffer is a continuous block of memory allocated by the host frame buffer. The host and the W2250 treat this buffer as a circular buffer by wrapping back to the starting address when they reach the end. The starting address and the size of the buffer are passed to the W2250 when initializing the CCE bus-mastering mode.

The application copies packets into the ring buffer in consecutive order starting at the top. It instructs the W2250 where to read the next packet by writing to a CCE write-pointer register. The W2250 triggers bus-mastering operations to transfer packets from the ring buffer to its CCE FIFO buffer according to watermarks set during CCE initialization. After completing the transfer, the W2250 uses bus-mastering to update a host application read-pointer to indicate where it has read to in the ring buffer. The physical address of this pointer is passed to the W2250 during CCE initialization (refer to *Figure 3-48 Ring Buffer on page 3-64*).

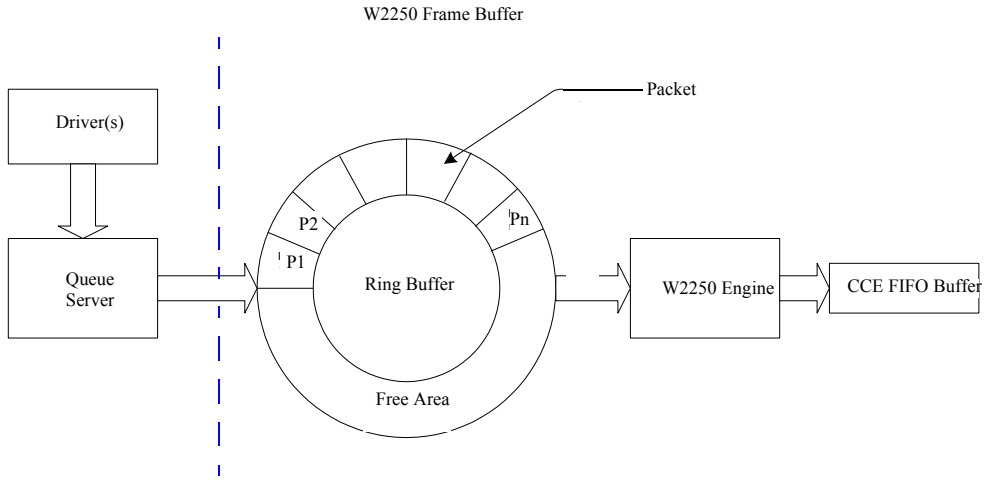

Figure 3-48 Ring Buffer

3.10.4 Ring Buffer Queue Server

For multitasking operating systems where multiple clients may require synchronized access to the graphics resources, it may be beneficial to employ a queue server mechanism to arbitrate and control access to the ring buffer. This mechanism could enumerate clients and use semaphores to synchronize and protect access.

3.11 Pulse Width Modulation Output

The W2250 provides a programmable pulse width modulated output through GPIO11. A 16-bit clock divider and a 16-step counter are used to achieve the desired frequency and duty cycle. The input clock frequency range is 2.6 - 9 MHz and the clock source is the CLI clock, which must be enabled by setting register CLI_CNTL bit [0] “EN_CLI” to "1".

The pulse width modulation output frequency is determined as follows: CLI clock frequency / (16 * PWM_REF_DIV)

The duty cycle can be changed by changing the PWM_DUTY_CNTL register:

- When PWM_DUTY_CNTL [20:16] is programmed to “0", then the pulse going out is always low.
- When PWM_DUTY_CNTL [20] is programmed to “1”, then the pulse going out is always high.
- When PWM_DUTY_CNTL [20:16] is programmed to “4", then the pulse going out is high for 4 clocks and then stays low for next 12 clocks.
- When PWM_DUTY_CNTL [20:16] is programmed to “8", then the pulse going out is high for 8 clocks and then stays low for the next 8 clocks.

NOTE: Bits [20:16] can be any value other than 00000 and 11111 to be valid.

If the input clock frequency changes, then the clock divider has to be programmed again. Care should be taken to ensure that the input clock does not vary dynamically. It is advised that you program the PWM_DUTY_CNTL to zero before changing the input clock frequency.

The pulse width modulation goes out through GPIO11. To enable GPIO11 output, set GPIO_DATA bit 27 to 1.

3.12 GPIO

The GPIOs in the W2250 are the general purpose I/Os. There are 24 GPIOs in the W2250. These can be used for various applications, either as input or as output pins. These are in tri-state initially at boot-up. By programming the output-enable and Schmitt Trigger enable for these I/Os, the GPIOs can be used as input or output pins.

The registers used to control GPIOs are as follows. Please refer to *Appendix C* for recommended GPIO assignments.

GPIO_DATA - RW - 32 bits - MMReg:0x4B8			
Field Name	**Bits**	**Default**	**Description**
GIO_OUT	15:0	0x0	16 bits of data to be output
GIO_OE	31:16	0x0	
This register defines the in/out data for GPIO bits 0 to 15.			

When GIO_OE is programmed to 0x0 then GPIO [15:0] are used as input.

GPIO_CNTL1 - RW - 32 bits - MMReg:0x4BC			
Field Name	**Bits**	**Default**	**Description**
GIO_PD	15:0	0xff00	Define 16 pull down enable for GPIO pin, default on
GIO_SCHMEN	31:16	0xff00	Enable the Schmitt trigger in GPIO pins
This register defines the control for GPIO bits 0 to 15.			

GIO_SCHMEN is enabled to read the contents of GPIO out through GIO_IN in GPIO_CNTL2 register.

GPIO_CNTL2 - RW - 32 bits - MMReg:0x4C0			
Field Name	**Bits**	**Default**	**Description**
GIO_IN (R)	15:0	0x0	
GIO_SRP	16	0x0	Define the SRP for 16 GPIO
GIO_SRN	17	0x0	Define the SRN for 16 GPIO
GIO_SP	21:18	0x0	Define the strength control SP for 16 GPIO
GIO_SN	25:22	0x0	Define the strength control SN for 16 GPIO
This register defines the control for GPIO bits 0 to 15.			

GPIO_DATA2 register is used to control GPIO[23:16].

GPIO_DATA2 - RW - 32 bits - MMReg:0x528			
Field Name	**Bits**	**Default**	**Description**
GIO2_OUT	15:0	0x0	Output value for GPIO 16 to 23
GIO2_OE	31:16	0x0	
This register defines data for GPIO bits 16 to 23.			

GPIO_CNTL3 - RW - 32 bits - MMReg:0x52C			
Field Name	**Bits**	**Default**	**Description**
GIO2_PD	15:0	0xff	Pull down enable for GPIO 16 to 23
GIO2_SCHMEN	31:16	0xff	Schmitt enable for GPIO 16 to 23
This register defines the control for GPIO bits 16 to 23.			

GPIO_CNTL4 - RW - 32 bits - MMReg:0x530			
Field Name	**Bits**	**Default**	**Description**
GIO2_IN (R)	15:0	0x0	
GIO2_SRP	16	0x0	
GIO2_SRN	17	0x0	
GIO2_SP	21:18	0x0	
GIO2_SN	25:22	0x0	
This register defines the output enable for GPIO bits 16 to 23.			

Chapter 4
Power Management

4.1 Power Modes

The W2250 chip has four main power modes:

- Fast
- Normal
- Slow
- Suspend

Fast Mode: The W2250 operates off of the PLLs in the chip with the sclk_fast_divider. The clocks to various blocks within the W2250 are enabled and disabled depending upon the operations that the chip is asked to perform. At that time PLL is programmed to 56Mhz and system clk is running at 56Mhz, CPU is running at maximum read/write access with write cycle = 40ns.

Normal Mode: The W2250 operates off of the PLLs in the chip with the sclk_slow_divider. The clocks to various blocks within the W2250 are enabled and disabled depending upon the operations that the chip is asked to perform. At that time PLL is programmed to 56Mhz and system clk is running at 14 Mhz, and the write cycle for the CPU is around 72ns.

Slow Mode: There are three configurations for this mode:

Configuration 1 (Crystal 14MHz): The W2250 PLL is powered down and the chip operates off the crystal oscillator.

Configuration 2 (RC - 5.2MHz): The W2250 PLL_32 is powered down and the chip operates off the RC oscillator to get 5.2MHz.

Configuration 3 (external 32kHz input or RC-32Khz): The W2250 PLL is powered down and the chip operates off the PLL_32 to get 5.2MHz.

At that time system clk is running at 5.2Mhz and CPU is running at slow mode (CLI update).

Suspend Mode: The W2250's PLLs are powered down and the crystal oscillator is disabled. In this mode, all clocks within the W2250 chip are disabled. Since power to the W2250 will not be disabled, all register contents and frame buffer contents are preserved. Switching to this mode can only be made through software control. The software must guarantee that the W2250 is idle, including the disabling of display refresh, before switching to this mode. This mode may be entered from either normal mode or slow mode.

4.2 Mode Transitions

4.2.1 Slow Mode to Normal Mode

The transition from slow mode to normal mode is completed by the W2250 in less than 1 ms. During this transition, the W2250 is able to accept bus cycles, i.e., access into the W2250. These bus cycles may be extended using the WAITb signal. Typically, the only access to configuration registers that may require the assertion of the WAITb signal when the chip is in slow mode are those to the special configuration registers: IND_DATA_A, IND_DATA_B, PM4_WRPTR, and SCRATCH. Concerning the access to other configuration registers, it does not matter whether the chip is in slow mode or normal mode. Also, slow mode is the mode that the chip will be in after reset.

4.2.2 Normal Mode to Fast Mode

Transition from fast mode to normal mode, and vice versa, can be performed using either hardware or software control. By default, any hardware control of power mode switching is disabled. Instead, there are independent register bits for enabling the features that allow hardware to control the transition from fast mode to normal mode, and vice versa.

4.2.3 Suspend Mode to Slow Mode

The W2250 wakes up from suspend mode either from seeing the transition of CSb to a low state, or from seeing the transition of both CSb and WEb to a low state, the transition of SSb or CSb to a low state, or the transition of SSb to a low state or both CSb and WEb to a low state. Also, the SD block can generate a wakeup signal to wakeup the system when

there is a SD card insertion or removal during system suspend. The condition the W2250 uses for waking up is determined by memory mapped register control programmed before putting the chip into suspend mode. The default state of this register is to wake up the W2250 when both CSb and WEb are low. Upon waking up, the W2250 will enter slow mode and then normal mode, depending upon memory mapped register settings programmed before putting the chip into suspend mode.

The transition from suspend mode to slow mode is completed in less than 2 ms. While the chip is in suspend mode, configuration register reads may be executed normally and will not wake up the chip if the W2250 registers are programmed accordingly. The only exceptions are register reads to the special configuration registers: IND_DATA_A, IND_DATA_B, PM4_WRPTR, and SCRATCH, where garbage data will be returned for the read data.

Note: While in suspend mode, any configuration register write other than the special configuration registers will still execute properly because these register writes do not require a clock. Writing to any of the special configuration registers while in suspend mode may result in the W2250 asserting WAITb for an unacceptably long time and may cause the W2250 to hang the bus interface because these special configuration registers require a clock in the W2250 to be present. Thus, special care should be taken to avoid writing to these registers while the chip is in suspend mode.

Through memory mapped register writes, software tells the W2250 which power mode it wants the W2250 to transition to. This register write can also be performed through the ring buffer. In this case, the rules for when this register write will be allowed to take place are either anytime, if the target power mode is normal mode; or, when the drawing engine becomes idle, if the target mode is slow mode. It is strongly recommended that software do **not** attempt to put the W2250 into suspend mode by a register write through the ring buffer for the following reason: Once the chip is put into suspend mode, all of the clocks in the W2250 will be disabled. Therefore, if the W2250 is put into suspend mode while some parts of the chip are still busy, then this may cause it to hang, and in the most severe case, the chip may hang the bus interface. **Software should only put the chip into either normal mode or slow mode through the ring buffer.**

The power management state transition diagrams for hardware and software operation are as follows:

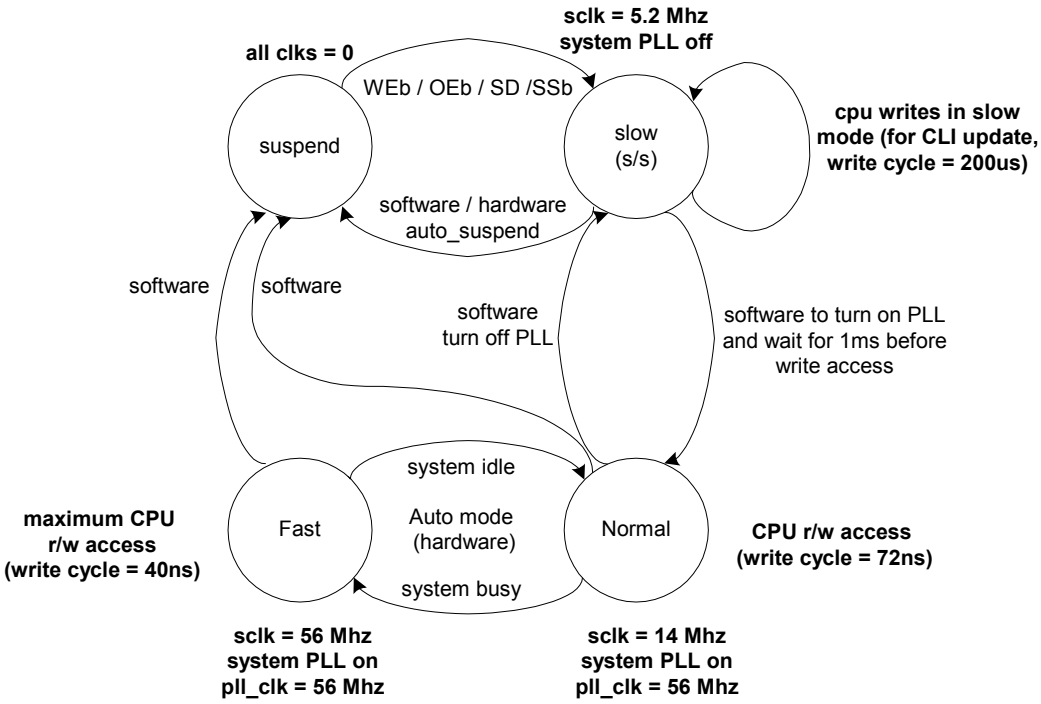

Figure 4-1 Power Management: State Diagram for Hardware Operation

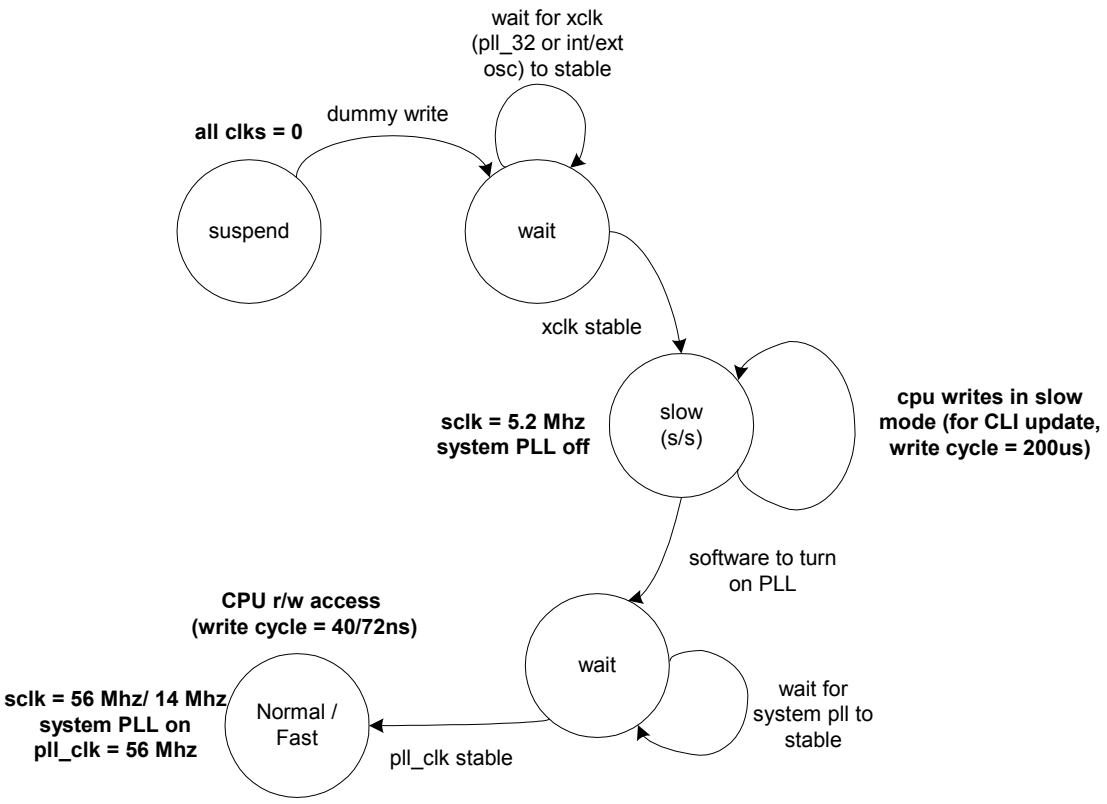

Figure 4-2 Power Management: State Diagram for Software Operation

4.3 Power Management - POWERPLAY

The ATI power management feature POWERPLAY™ provides a combination of software and hardware capabilities that optimize power consumption and performance characteristics on all PDA type graphics products, including the W2250 family. POWERPLAY includes the following features:

- On-chip power management logic to switch the chip between different power states.
- Dynamic clock switching. This technique switches off clocks to idle hardware blocks during normal operation, thereby reducing power consumption.
- Clock gating on all registers.
- New low power internal clock design.

Table 4-1 shows a breakdown of W2250 main functions and associated power. All main blocks will be shut off automatically if they are idle, or can be shut off through software control.

Table 4-1 W2250 Main Functions and Associated Power

Core power while running the following operation:	2.6MHz SCLK				5.2MHz SCLK				20MHz SCLK				30MHz SCLK			
	1.3V		1.5V		1.3V		1.5V		1.3V		1.5V		1.3V		1.5V	
	Peak (mW)	Typical (mW)	Peak (mW)	Typical (mW)	Peak (mW)	Typical (mW)	Peak (mW)	Typical (mW)	Peak (mW)	Typical (mW)	Peak (mW)	Typical (mW)	Peak (mW)	Typical (mW)	Peak (mW)	Typical (mW)
2D Moving Blit		2.39		3.27		4.60		6.15		16.62		22.19		25.36		38.88
MPEG		2.59		3.90		4.94		7.10		15.83		21.15		17.61		24.61
JPEG Encode		n/a		n/a		n/a		n/a		9.14		12.35		13.17		19.67

Table 4-1 W2250 Main Functions and Associated Power (Continued)

Core power while running the following operation:	2.6MHz SCLK				5.2MHz SCLK				20MHz SCLK				30MHz SCLK			
	1.3V		1.5V		1.3V		1.5V		1.3V		1.5V		1.3V		1.5V	
	Peak (mW)	Typical (mW)	Peak (mW)	Typical (mW)	Peak (mW)	Typical (mW)	Peak (mW)	Typical (mW)	Peak (mW)	Typical (mW)	Peak (mW)	Typical (mW)	Peak (mW)	Typical (mW)	Peak (mW)	Typical (mW)
VIP Preview with scale		1.80		2.49		3.60		4.85		9.20		12.46		13.21		19.79
SD		0.62		0.85		1.19		1.63		4.21		5.59		6.39		9.82
Display refresh		0.87		1.20		1.37		1.85		3.88		5.18		5.74		8.70
PLL (both)		0.07		0.10		0.11		0.14		0.18		0.23		0.19		0.25
Clock Block Power		0.45		0.63		0.89		1.22		3.07		4.12		4.71		7.19
Host Memory Access (CIF+MC only)		1.09		1.49		1.58		2.17		4.18		5.62		6.05		9.38

Table 4-2 Power Consumption for Screen Refresh - Refresh Only Mode 176x220 by 16bpp

PCLK Frequency (divider setting)	Core Power (mW)
650 kHz (1/4)	0.6
867 kHz (1/3)	0.7
1.3 MHz (1/2)	0.8
2.6 MHz(1/1)	0.9

Table 4-3 Video Capture Power (mW)

Core Voltage (V)	QCIF		CIF		QVGA		VGA	
	Preview only	Capture 30f/s	Preview only	Capture 30f/s	Preview only	Capture 30f/s	Preview only	Capture 15f/s
1.2	2.43	3.21	2.76	7.92	2.48	3.25	5.23	13.48
1.50	3.79	5.01	4.31	12.37	3.88	5.07	8.18	21.07
1.80	5.46	7.22	6.21	17.81	5.59	7.30	11.77	30.34

Table 4-4 Power Saving States and Estimated Power Consumption

Power Mode	Core Frequency/Clock Source	Estimated Average Power Consumption
Low Speed	7 MHz (approx.) / Internal PLL	6mW
Display Refresh Only	Minimum clock frequency needed for the display refresh (1 to 3MHz)	< 2.0mW
Suspend	0 MHz	40 μW

Software Implementation

The software driver sets the W2250 to a specified clock and power configuration during initialization. The configuration is tailored to specific OEM requirement using a text hardware information table. The driver will automatically detect what the board configuration (1, 2, or 3) is, based on the strap setting. When needed, the software will also automatically calibrate the PLL lock to the required frequency or to the closest frequency.

The following items can be specified by the OEMs based on their platform requirement:

- Fast system clock frequency (MHz)
- External oscillator frequency (MHz)
- Normal system clock frequency divider

- Normal system clock source from pll or the external oscillator
- Pixel clock frequency divider
- Pixel clock source from pll or the external crystal
- VIP clock frequency divider
- VIP clock source from pll or the external crystal
- CLI clock frequency divider
- CLI clock source from pll or the external crystal
- Enable or disable software power management
- Enable or disable hardware automatic fast-slow-fast mode switching

To enhance the POWERPLAY feature and save power, the driver will be based on the target operating frequency specified to turn on and off any PLLs that are not necessary.

Upon any power mode changes - for example from fully-on to suspend - the operating system will inform the driver via a registered API and the driver will set the hardware to the requested mode.

This page intentionally left blank.

Chapter 5
Electrical and Timing Specification

5.1 CPU Interface

5.1.1 Parallel CPU Interface (powered on VDDR1)

Table 5-1 Parallel CPU Interface (powered on VDDR1)

Pin Name	Type	Description
DQ[15:0]	I/O	Data bus
AD[22:0]	I	Address bus
CSb	I	Chip Select active LOW
OEb	I	Output Enable active LOW
WEb	I	Write Enable active LOW (optional active HIGH by STRAP_INV_DMAC_CLK)
BE[1:0]	I	Byte Enable during Write Operation
INTb	O	INTERRUPT active LOW (optional active HIGH by programming)
RESETb	I	System RESET active LOW
MR	I	Chip Select active HIGH (indirect mode) or R/Wb for early wait (direct mode)
WAITb	I	Used in Direct mode only. Wait for Data ready (read cycle) or accepted (write cycle). Selected by strap to be active LOW (default) or HIGH.
SRCCLK	I	32kHz clock input

Notes:

- The data bus is available in either 16-bit or 8-bit configuration, determined by packaging.
- INDIRECT mode requires only AD[0] to be driven by CPU (i.e. by CPU address bit 23). In this case, all the rest of AD pins are internally pulled down.

5.1.2 Serial CPU (QSPI) Interface (powered on VDDR1)

Table 5-2 Serial CPU (QSPI) Interface (powered on VDDR1)

Pin Name	Type	Description
SSb	I	QSPI Select active LOW (muxed with AD[22])
SCK	I	QSPI clock (muxed with AD[21])
MISO	O	Data from W2250 to CPU (muxed with AD[20])
MOSI	I	Data from CPU to W2250 (muxed with AD[19])

5.1.3 Mapping the CPU interfaces to W2250 BUS interface pins

Table 5-3 Mapping the CPU Interfaces to W2250 BUS Interface Pins in INDIRECT Mode

W2250	PIN to W2250	CPU EIM intf	CPU DMAC intf
DQ [15:0]	I/O	D[15:0]< see note 2 >	LCD_DATA[7:0]
MISO(AD[20])	O	N/C< see note 1 >	MISO
MOSI(AD[19])	I	N/C< see note 1 >	MOSI
AD [3:0]	I	A[0] < see note 3 >	LCD_RS
CSb	I	CS_n or TIEHI (inactive)	CS_n or TIEHI (inactive)
SSb	I	TIEHI to disable<See note 1>	SSb
OEb	I	OEb	TIEHI(inactive)
WEb	I	EB01_b	LCD_CS
SCK	I	TIELO to disable<See note 1>	SCK
BE [1:0](optional)	I	TIELO(active)< see note 4 >	TIELO(Don't care)
MR(optional)	I	TIELO(inactive)	TIELO(inactive)
WAITb	O	N/C	N/C
INTb	O	INT	

Table 5-3 Mapping the CPU Interfaces to W2250 BUS Interface Pins in INDIRECT Mode (Continued)

W2250	PIN to W2250	CPU EIM intf	CPU DMAC intf
RESETb	I	System Reset	

Note:

1. After system reset, QSPI is, by default, enabled for PCS and disabled for IDEN. However, QSPI can also be enabled for the IDEN configuration. In this case, the CPU pins need to be mapped according to the W2250 pins dedicated to the QSPI, the same as is specified for the PCS configuration. If QSPI is enabled in the IDEN configuration, a pull down is required on SSb (10K is suggested).
2. The data bus can be available in either 16bit or 8bit configuration, determined by packaging.
3. Configuration registers can be accessed by using either of the following methods:
   - Physically connect all AD[3:0] bits, and use them to select configuration registers.
   - Physically connect only the AD[0] bit, and use it to enable DQ[3:0] to select configuration registers, as described in the attached BUS protocol diagrams for 1bit addressing.
4. BYTE WRITEs can be controlled by configuration registers
5. WAITb is driven when CSb is active and when WAIT protocol is enabled (by strap or register settings); otherwise, it is tri-stated.
6. INTb is driven when an interrupt is being asserted; otherwise, it is tri-stated.

5.2 LCD Interface (powered by VDDR2)

Table 5-4 LCD Interface (powered on VDDR2)

Pin Name	Type	Description
LCDD0	O	TFT-Bit 0, red STN-D0
LCDD1	O	TFT-Bit 1, red STN-D1
LCDD2	O	TFT-Bit 2, red STN-D2
LCDD3	O	TFT-Bit 3, red STN-D3
LCDD4	O	TFT-Bit 4, red STN-D4
LCDD5	O	TFT-Bit 5, red STN-D5
LCDD6	O	TFT-Bit 0, green STN-D6
LCDD7	O	TFT-Bit 1, green STN-D7
LCDD8	O	TFT-Bit 2, green STN-D8
LCDD9	O	TFT-Bit 3, green STN-D9
LCDD10	O	TFT-Bit 4, green STN-D10
LCDD11	O	TFT-Bit 5, green STN-D11
LCDD12	O	TFT-Bit 0, blue STN-D12
LCDD13	O	TFT-Bit 1, blue STN-D13
LCDD14	O	TFT-Bit 2, blue STN-D14
LCDD15	O	TFT-Bit 3, blue STN-D15
LCDD16	O	TFT-Bit 4, blue
LCDD17	O	TFT-Bit 5, blue
DCLK	O	Data Clock
SS	O	Source Start
LS	O	Latch Start
GS	O	Gate Start
GCLK	O	Gate Clock
GOE	O	Signal for refresh control (GMODE)
DINV	O	Data Inversion
FRAME	O	AWL
CLI_CLK	O	CLI Clock
CLI_CMD	O	CLI Command
CLI_DOUT	O	CLI Data out

Table 5-4 LCD Interface (powered on VDDR2)

Pin Name	Type	Description
CLI_CSb	O	CLI Chip Select active LOW
CLI_RESETb	O	CLI Reset active LOW

5.3 General Purpose IO Pins

Table 5-5 GPIO PIns (powered on VDDR4)

Pin Name	Type	Description
GPIO[23:20]	I/O	General purpose I/O pins (powered on VDDR4)

Table 5-6 GPIO Pins (powered on VDDR2)

Pin Name	Type	Description
GPIO[19:0]	I/O	General purpose I/O pins (powered on VDDR2)

5.4 Video In Interface (powered on VDDR4)

Table 5-7 Video In Interface (powered on VDDR4)

Pin Name	Type	Description
VIPCLK	I	Video In Port Clock
VIPCLKO	O	Camera Clock
VIPDATA0	I	Video In Port Data
VIPDATA1	I	Video In Port Data
VIPDATA2	I	Video In Port Data
VIPDATA3	I	Video In Port Data
VIPDATA4	I	Video In Port Data
VIPDATA5	I	Video In Port Data
VIPDATA6	I	Video In Port Data
VIPDATA7	I	Video In Port Data
VIPVSYNC	I	Video In Port VSYNC
VIPHREF	I	Video In Port HREF
SCL	I/O	Industry-standard, 2-wire interface clock
SDA	I/O	Industry-standard, 2-wire interface address and data

5.5 SDIO Interface (powered on VDDR5)

Table 5-8 SDIO Interface (powered on VDDR5)

Pin Name	Type	Description
SDC0CLK	O	SD Card Clock
SDC0CMD	I/O	SD Card Command
SDC0DQ0	I/O	SD Card Data
SDC0DQ1	I/O	SD Card Data
SDC0DQ2	I/O	SD Card Data
SDC0DQ3	I/O	SD Card Data
SDC0CD	I	Card detection
SDC0WP	I	Position of Write Protect Mechanical Switch

5.6 JTAG Pins (powered on VDDR4)

Table 5-9 JTAG Pins (powered on VDDR4)

Pin Name	Type	Description
TESTEN (TRST)	I	JTAG asynchronous reset (must be tied LOW during normal operation)
TCLK	I	JTAG clock (muxed with GPIO[20])
TMS	I	JTAG Test Mode Select (muxed with GPIO[21])
TDI	I	JTAG Data In (muxed with GPIO[22])
TDO	O	JTAG Data Out (muxed with GPIO[23])

5.7 Crystal Interface

Table 5-10 Crystal Interface

Pin Name	Type	Description
XTALIN*	I	Crystal oscillator
XTALOUT*	I/O	Crystal oscillator

* When only the 32KHz clock mode is used, these pins do not need to be connected.

5.8 Specialty Power and Ground Pins

Table 5-11 Specialty Power and Ground Pins

Pin Name	Type	Description
PVDD	I	PLL/Oscillator power pad (connected to core power)
PVSS	I	PLL/Oscillator ground pad (connected to core ground)

5.9 Power and Ground Pins

Table 5-12 Power and Ground Pins

Pin Name	Type	Description
VDDC	I	Core power
VDDR1	I	CPU Interface power
VDDR2	I	LCD I/O power (includes GPIO[19:0], SPKG and SCVOLT[1:0])
VDDR4	I	JTAG, Video In power (includes GPIO[23:20])
VDDR5	I	SDIO power
VSS	I	Common ground

5.10 W2250 Straps

Table 5-13 W2250 Straps

Strap	Pin	Default	Description
STRAP_CORE_VOLTAGE	SCVOLT1SCVOLT0	00	'00' : 1.2V '01' : 1.3V '10' : 1.5V '11' : 1.8V Note: This strap is only read in during a hard reset using the RESETb pin asserted. If the SOFT_RESET configuration register bit is set, this strap register will not be reset.
STRAP_WAIT_ACTIVE_HIGH	GPIO15	TIELO	Configures WAITb to be active LOW
STRAP_BUS_SEL	GPIO16	TIELO	'0' : 16-bit data bus (QSPI defaults disabled) '1' : 8-bit data bus

Table 5-13 W2250 Straps

Strap	Pin	Default	Description
STRAP_EARLY_WAIT_RDWR_CNTL	GPIO13	TIELO	'0' : MR pin TIELO '1' : MR pin function is RD/WRb bus signal. Used for early indication of read or write from CPU.
STRAP_EARLY_WAIT_BKU	GPIO14	TIELO	Generate early WAITb, without using RD/WRb signal
STRAP_INV_DMAC_CLK	GPIO18	TIELO	Use inverted clock for DMAC interface
STRAP_XCLK_SRC_SEL	GPIO23GPIO22	'00	"00' : 32kHz clock + PLL '01' : Internal RC oscillator '10' : Crystal Oscillator '11' : Direct input
SPKG	SPKG	0	Strap for defining Direct or Indirect interface; Low for indirect.

5.11 Maximum Rating Conditions

Table 5-14 Maximum Rating Conditions

Parameter	Minimum	Maximum	Units
VDDC	0	1.98	V
VDDR	0	3.6	V

5.12 Operating Conditions

Table 5-15 Operating Conditions

Parameter	Minimum	Typical	Maximum	Units
Temperature	-20	25	85	°C
Humidity (non-condensing)	5		90	%

5.13 Storage Conditions

Table 5-16 Storage Conditions

Parameter	Minimum	Maximum	Units
Temperature	-40	150	°C
Relative Humidity	0	95	%

5.14 Power Requirements

The following tables specify the power supply voltages and current consumption for the W2250.

Table 5-17 Power Supply Voltages

Parameter	Minimum (V)	Recommended Value (V)	Maximum (V)
Core (VDDC)	1.08	1.2 to 1.8 ±10%	1.98
CPU interface (VDDR1)	1.35	1.5 to 3.3 ±10%	3.6
LCD Interface (VDDR2)	1.35	1.5 to 3.3 ±10%	3.6
Camera interface (Video in) (VDDR4)	1.35	1.5 to 3.3 ±10%	3.6
SD (VDDR5)	2	3.3	3.6

The highest voltage rail, VDDR4, must be powered up first - (See section 5.15 "ESD Protection (VDDR4)"). The remaining rails should be powered up in descending order of voltage. When powering down the chip, VDDR4 should be powered down last or at the same time as the other power rails.

Table 5-18 Maximum Current Consumption

Parameter	Maximum (mA)
Current consumption	250

5.15 ESD Protection (VDDR4)

VDDR4 provides ESD protection when the following conditions are met:

- VDDR4 is the highest voltage of all the power rails
- VDDR4 is powered up first and powered down last

5.16 CPU Interface Electrical Characteristics

- Typical loading for this pad is 35 pF.
- Output driver should have a minimum slew rate of roughly 5ns.

Table 5-19 CPU Interface Electrical Characteristics

Symbol	Parameter	Minimum	Typical	Maximum
VDDC	Core voltage	1.08V	1.2V 1.5V 1.8V	1.98V
VDDR	I/O Voltage	1.35V	1.5V 1.8V 2.5V 3.0V 3.3V	3.6V
VIN	Input voltage	VSS - 0.5V		VDDR + 0.5V
VOUT	Output voltage	VSS - 0.2V		VDDR + 0.2V
VIH	Input high level voltage: VDDR = 2.5/3.0/3.3V	VDDR - 0.8V		
	Input high level voltage: VDDR = 1.5/1.8V	VDDR - 0.4V		
VIL	Input low level voltage: VDDR = 2.5/3.0/3.3V			VSS + 0.8V
	Input low level voltage: VDDR = 1.5/1.8V			VSS + 0.4V
VOH	Output high level voltage: VDDR = 2.5/3.0/3.3V	VDDR - 0.3V		
	Output high level voltage: VDDR = 1.5/1.8V	VDDR - 0.15V		
VOL	Output low level voltage: VDDR = 2.5/3.0/3.3V			VSS + 0.3V
	Output low level voltage: VDDR = 1.5/1.8V			VSS + 0.15V
CIO	Pin Capacitance			10 pF
ISLEW	Output slew rate	3 ns	5 ns	
TCYC	Toggling period	20 ns		

Notes:

- Core voltage must be lower than I/O voltage.
- VDDR4 must be the highest voltage rail. (See section 5.15 "ESD Protection (VDDR4)".)

5.17 The LCD Interface Electrical Characteristics

- Maximum loading for this pad is 20 ~ 25 pF
- Output driver have a minimum slew rate of roughly 5ns.

Table 5-20 LCD Interface Electrical Characteristics

Symbol	Parameter	Minimum	Typical	Maximum
VDDC	Core voltage	1.08V	1.2V 1.5V 1.8V	1.98V
VDDR	I/O voltage	1.35V	1.5V 1.8V 2.5V 3.0V 3.3V	3.6V
VIN	Input voltage	VSS - 0.5V		VDDR + 0.5V
VOUT	Output voltage	VSS - 0.2V		VDDR + 0.2V
VIH	Input high level voltage: VDDR = 2.5/3.0/3.3V	VDDR - 0.8V		
	Input high level voltage: VDDR = 1.5/1.8V	VDDR - 0.4V		
VIL	Input low level voltage: VDDR = 2.5/3.0/3.3V			VSS + 0.8V
	Input low level voltage: VDDR = 1.5/1.8V			VSS + 0.4V

Table 5-20 LCD Interface Electrical Characteristics

Symbol	Parameter	Minimum	Typical	Maximum
VOH	Output high level voltage: VDDR = 2.5/3.0/3.3V	VDDR - 0.3V		
	Output high level voltage: VDDR = 1.5/1.8V	VDDR - 0.15V		
VOL	Output low level voltage: VDDR = 2.5/3.0/3.3V			VSS + 0.3V
	Output low level voltage: VDDR = 1.5/1.8V			VSS + 0.15V
CIO	Pin Capacitance			10 pF
ISLEW	Output slew rate	3 ns	5 ns	
TCYC	Toggling period	20 ns		

5.18 Reset State

The W2250 can be reset by asserting the RESETb pin (active low) during power-up. Please note that a stable clock input is required for hardware reset.

After reset, the W2250 chip runs in "NORMAL MODE". In NORMAL MODE, the W2250's PLL is powered down and the W2250 chip runs off the crystal clock or PLL_32. All of the W2250 registers are in their default states and all contents of the frame buffer must be considered invalid.

Table 5-21 Parameters of the W2250 Chip Related Reset

Symbol	Parameter	Minimum	Maximum
tRST	Reset active time with stable power	10 ms	-
tRSTOFF	Time from reset deassertion to first W2250 bus cycle	10 ms	-

The following is a diagram of signals after stable power is applied to the W2250 chip.

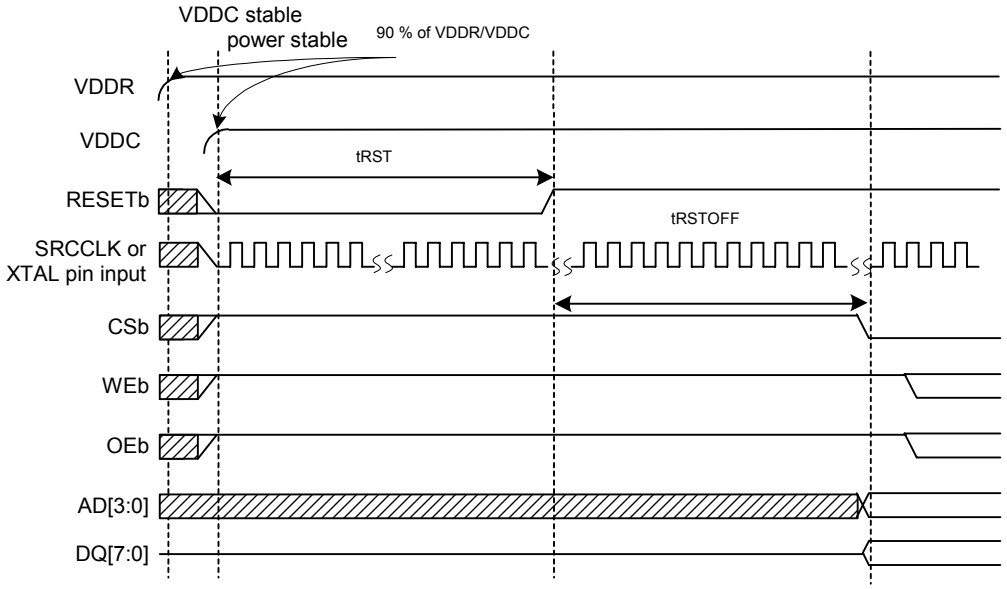

Figure 5-1 Waveform of Reset during Power-up

After stable power is applied to the W2250 chip, RESETb must be held low for tRST. During this time, CSb and OEb must be held high in order to guarantee that the W2250 is not driving the DQ lines. After RESETb is driven high, the first bus cycle to the W2250 chip can occur tRSTOFF-time later.

If an attempt to read any W2250 register occurs during either of the conditions listed below, the data returned will not be valid.

- RESETb pin is asserted.

  or

- the read is attempted before tRSTOFF ends following the deassertion of reset.

Note that if a bus cycle to read any of the W2250 registers (through the QSPI interface or the CPU interface) occurs while the RESETb pin is asserted or before tRSTOFF expires, then the data returned will not be valid.

This page intentionally left blank.

Chapter 6
Programming Interface

6.1 Software Flow

6.1.1 Graphic/Display Subsystem Flow

The graphics subsystem provides accelerated drawing functions, while the display subsystem provides a means to output regions of video memory onto various types of display panels.

Figure 6-1 Typical Flow of Imageon-Powered Graphics Subsystem on page 6-1 illustrates the typical flow of an Imageon-powered graphics subsystem.

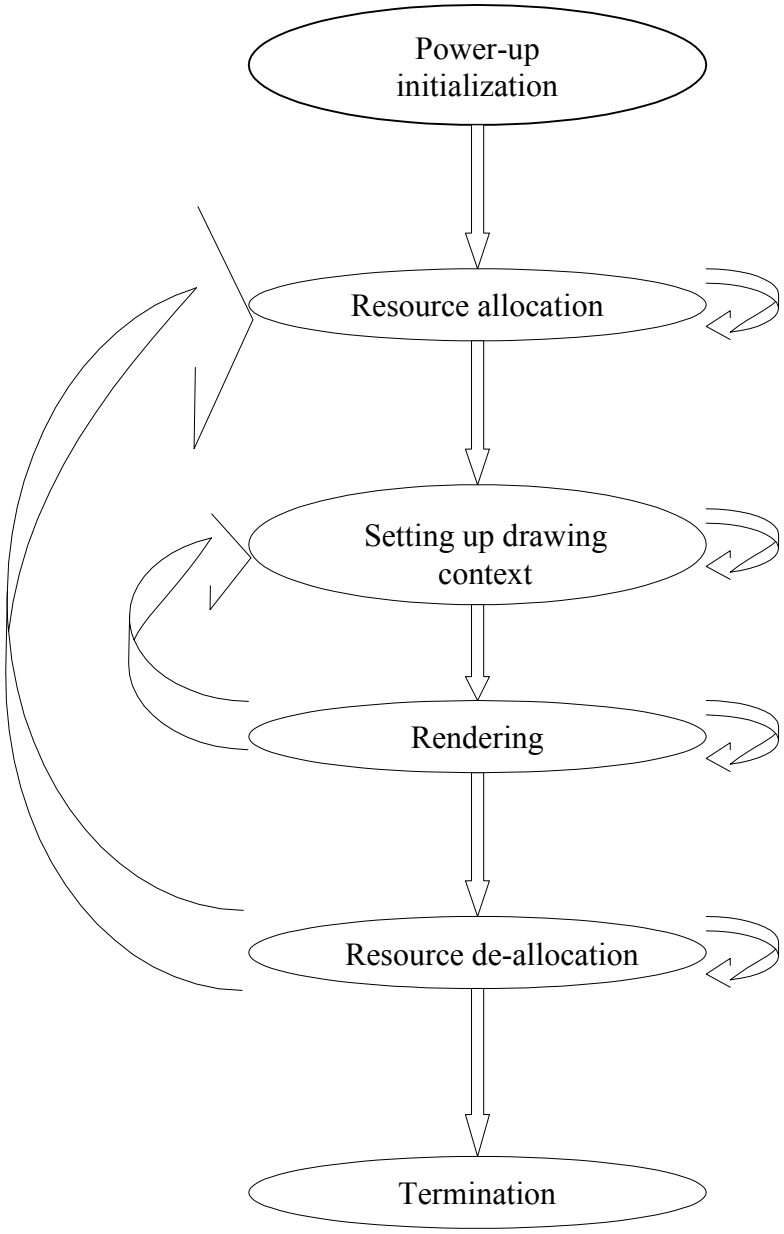

Figure 6-1 Typical Flow of Imageon-Powered Graphics Subsystem

Power-up initialization
This includes calls to bring up Imageon, as well as to initialize power management settings, clock settings, as well as display panel timing setup.

Resource allocation
Before any operation can take place, the necessary resources must be acquired. Resources important to graphics include:

- Surfaces
  These are blocks of memory to be used as source or destination in rendering operations.
- Graphic window
  This resource displays a view of the video memory. For example, in order to display a piece of surface, the graphic window has to be setup to point to that surface.
- Overlay window
  Similar to the graphic window, but uses YUV surfaces and Gfx when keying is required.
- Cursors
  Hardware cursor allows visual tracking of the pointer in a graphical user interface.

Setting up drawing context
Drawing context refers to one or more states that affect rendering operations. For example, in order to fill a rectangle with solid colour, the states that need to be setup are the destination surface, the raster operation (ROP), and the colour. Drawing context is included, but not limited to:

- Foreground colour
  Default colour for colour fills.
- Background colour
  Default colour for clearing the background.
- Brush type
  Specifies the pattern to be realized for drawing operations involving brush.
- Source surface
  Attributes of the source, such as its location, dimension, and type.
- Destination surface
  Attributes of the destination, such as its location, dimension, and type.
- Raster operation
  Specifies the relationship between source, destination, pattern, and optionally mask.
- Byte pixel order
  Specifies whether a 32-bit value should be interpreted LSB first or MSB first.
- Clipping regions
  Specifies separately on the source and destination surfaces that are to be ignored. For example, anything outside of the clipping region on the source surface will not get copied, whereas anything outside of the clipping region will not be written.

Rendering
These operations will draw to surfaces through the GUI engine, such as:

- Rectangles
  Filling or copying rectangular regions.
- Lines
  Filling paths specified by a series of endpoints.
- Host
  Source data comes in from outside of the graphics subsystem, to fill a 2-dimensional area.

Resource de-allocation
This stage allows for the de-allocation of resources, which may be needed before new resources are allocated. For example, in order to create a bigger size surface, you may need to first free up any surface that is not in use.

Termination
This involves all the clean up activities before power down.

6.1.2 IDCT/MCOMP Subsystem Flow

IDCT/Mcomp (Inverse Discrete Cosine Transform/Motion Compensation) subsystem provides entry points for IDCT/Mcomp decoding, as well as post-processing of the decoded video.

Figure 6-2 Typical Flow of Imageon-powered IDCT/MCOMP Subsystem on page 6-3 illustrates the typical flow of an Imageon-powered IDCT/MCOMP subsystem.

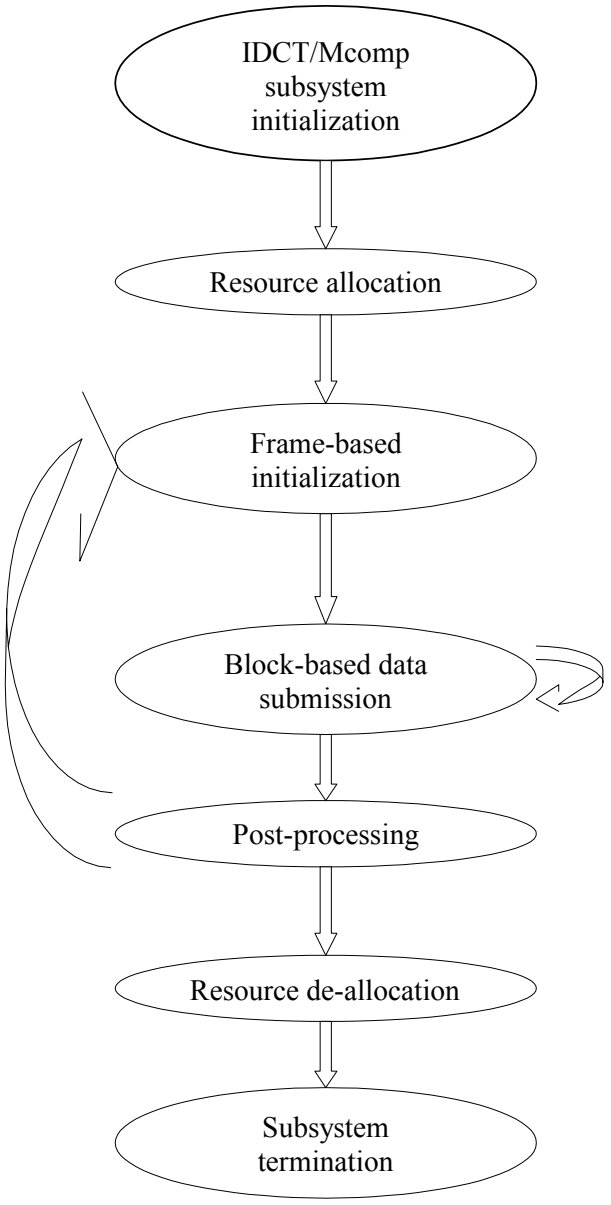

Figure 6-2 Typical Flow of Imageon-powered IDCT/MCOMP Subsystem

IDCT/Mcomp subsystem initialization
This signifies that the IDCT/Mcomp subsystem is to be used. This is also where some states are set which will affect how the incoming data will be treated.

- Zigzag order
  This specifies the order of the IDCT coefficients inside each block.
- Rounding
  Control rounding type for Mcomp.
- Bias
  Specifies whether to add 128 to intra block values. True for MPEG1/2, false for MPEG4.

Resource allocation
Before any operation can take place, the necessary resources must be acquired. The following resources are relevant to this subsystem:

- Surfaces
  These are blocks of memory to be used as decoding buffers or display buffers.
- Overlay window
  This resource displays a view of the video memory, and is able to translate YUV to RGB.

Frame-based initialization
For each frame, one needs to know the decoding and reference surfaces.

Block-based data submission
IDCT coefficients and motion vectors are submitted on a block basis. A block can be of type Y0, Y1, Y2, Y3, U, or V. Six blocks make up a macro-block. For each block, the associated macro-block number has to be specified, this combined with the block type determines where in a frame this block belongs.

Post-processing
After each frame has been decoded, you can specify post-processing options before the frame is displayed. Currently, the options to rotate and scale the frame are supported.

Resource de-allocation
This stage allows for the de-allocation of resources, which may be necessary before new resources are allocated.

Subsystem termination
This is the counterpart to the IDCT/Mcomp subsystem initialization.

6.1.3 JPEG Subsystem Flow

The JPEG subsystem provides entry points for JPEG Preview and JPEG Encoding. The flow represents how software can set the subsystem into its various mode, control data flow and enable features.

Figure 6-3 Typical Flow of Imageon-powered JPEG Subsystem on page 6-5 illustrates the typical flow of an Imageon-powered JPEG subsystem.

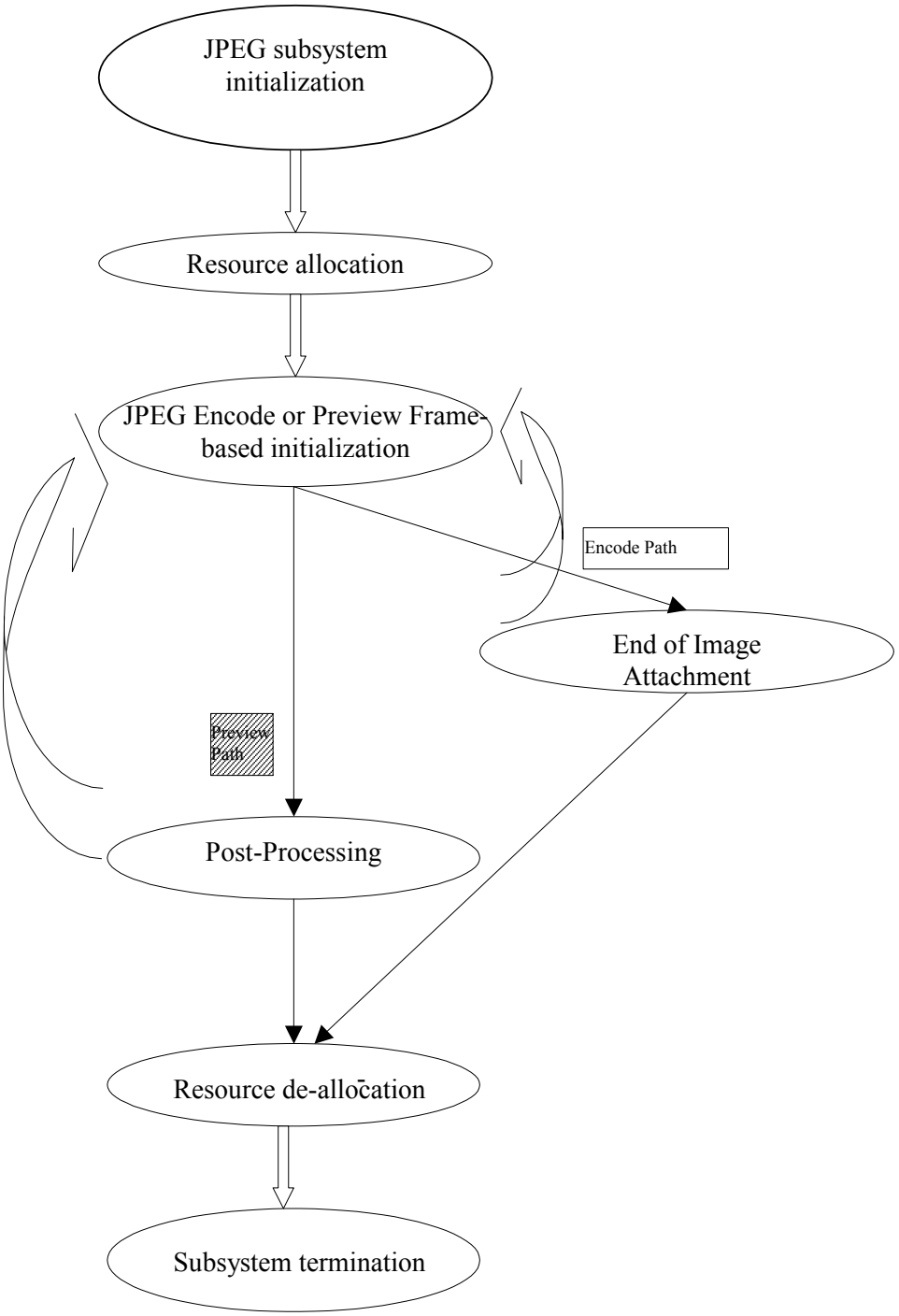

Figure 6-3 Typical Flow of Imageon-powered JPEG Subsystem

JPEG subsystem initialization
This signifies that the JPEG subsystem is to be used. This initialization includes setting up quantization tables and setting the JPEG block's mode into either Preview or Encoding.

- JPEG Preview Mode
  VIP Video Source data is converted and passed through to display in this mode.
- JPEG Encode Mode
  VIP Video Source data is encoded and a JPEG scan is written back to RAM.

Resource allocation
Before any operations can take place, the necessary resources must be acquired. The following resources are relevant to this subsystem:

- Surfaces
  These are blocks of memory to be used as encoding buffers or display buffers.
- Overlay window
  This resource displays a view of the video memory, and is able to translate YUV to RGB.

Frame-based initialization
For each frame, we need to know the input format, preview scale value and encoding surfaces.

Post-processing
Post-processing options can be specified before the frame is displayed in preview mode.

Resource de-allocation
This stage allows for the de-allocation of resources, which may be necessary before new resources are allocated.

Subsystem termination
This is the counterpart to JPEG subsystem initialization.

6.1.4 QSPI Subsystem Flow

The QSPI (The Serial Port Interface) is designed to interface with a SPI master and works in conjunction with the W2250 serial and parallel CPU interfaces. The software QSPI subsystem provides a mechanism to program the hardware QSPI into its various mode and behavior.

Figure 6-4 Typical Flow of QSPI Configuration Sequence on page 6-6 illustrates the typical flow of a QSPI Configuration Sequence.

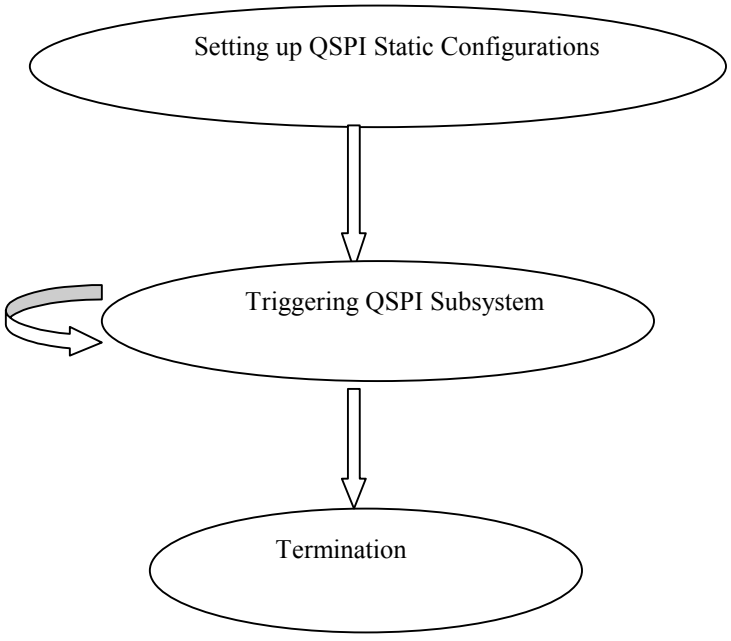

Figure 6-4 Typical Flow of QSPI Configuration Sequence

Setting Up QSPI Static Configurations
The QSPI static configurations need to be programmed at least once on power-up. This includes the setting of clock polarity, slave select polarity, half clock adjustment, transmitting/receiving options with MSB or LSB and read/write latencies.

Triggering QSPI Subsystem
QSPI Subsystem can be triggered multiple times.

Subsystem Termination
This is the counterpart to QSPI subsystem initialization.

6.2 AHI Driver Library Application Programming Interface

AHI (ATI Handheld Interface) is a platform-independent programming interface that provides access to the hardware features of various products ATI produces for handheld devices.

AHI is platform-independent because it is designed to work with various processors and operating systems on different hardware configurations. Providing access to hardware features through the abstraction of an interface allows developers to leverage hardware operations across multiple platforms. Furthermore, developers need not understand all the subtle nuances required to program each ATI-enabled, handheld device.

Developer's Guides are available for each of the various W2250 subsystems. Contact ATI Handheld Application Engineering for more details.

6.3 Industry-Standard 2-Wire Interface

This industry-standard, 2-wire interface is a 2-bit serial bus that allows the programming of peripherals such as video decoders, TV tuners, teletext, and volume control.

Two dedicated signals, I2CCLK and I2CDATA, are used to drive the clock and data interface pins. The data transfer can be done in two modes: software mode or hardware-assisted mode. In general, the software data transfer has greater flexibility, but yields a lower throughput because of the time involved in programming the GPIOs.

Since the clock and data signals are open-collector and rely on external pull-up resistors, noise can be a factor during transition from low to high. In the W2250, the clock and data signals can either be driven high directly, or be pulled up by external resistors. Note that these two signals are controlled separately, thereby allowing only one signal to be driven high without creating problems.

This page intentionally left blank.

Chapter 7
Mechanical Characteristics

7.1 W2250 BGA Package

Package outline: BGA 10x10mm - 181 pins.

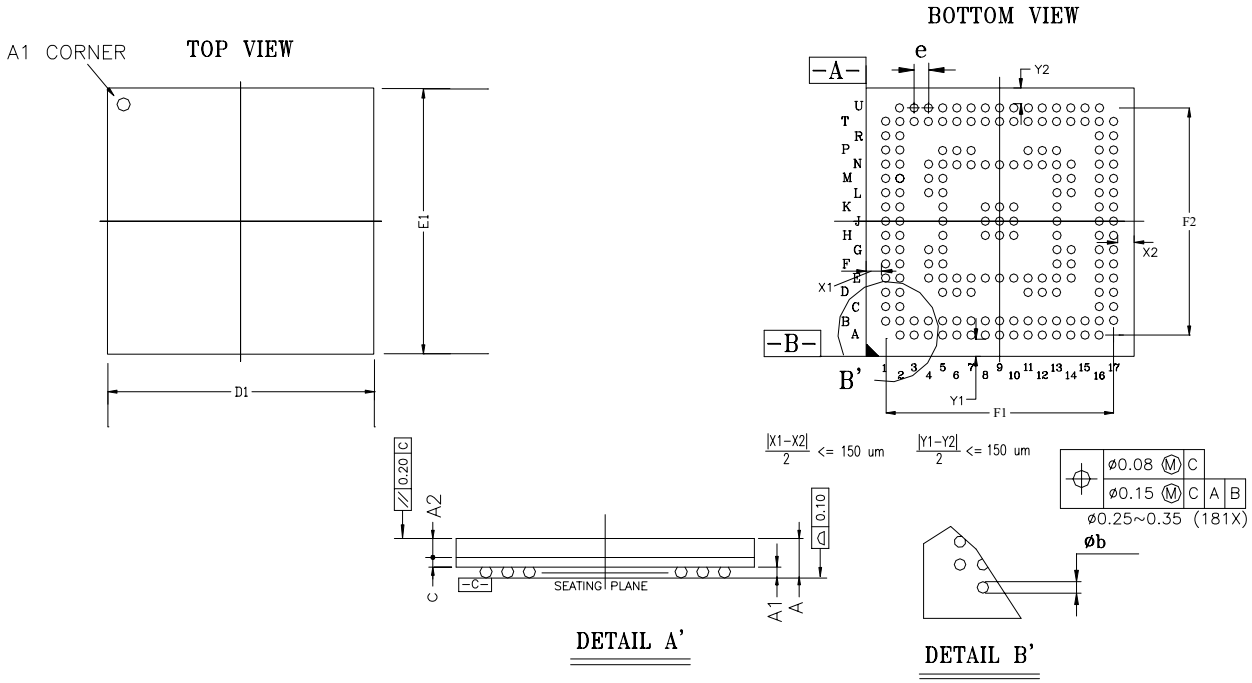

Figure 7-1 W2250 Package Outline

7.2 W2250 Flip Chip Package

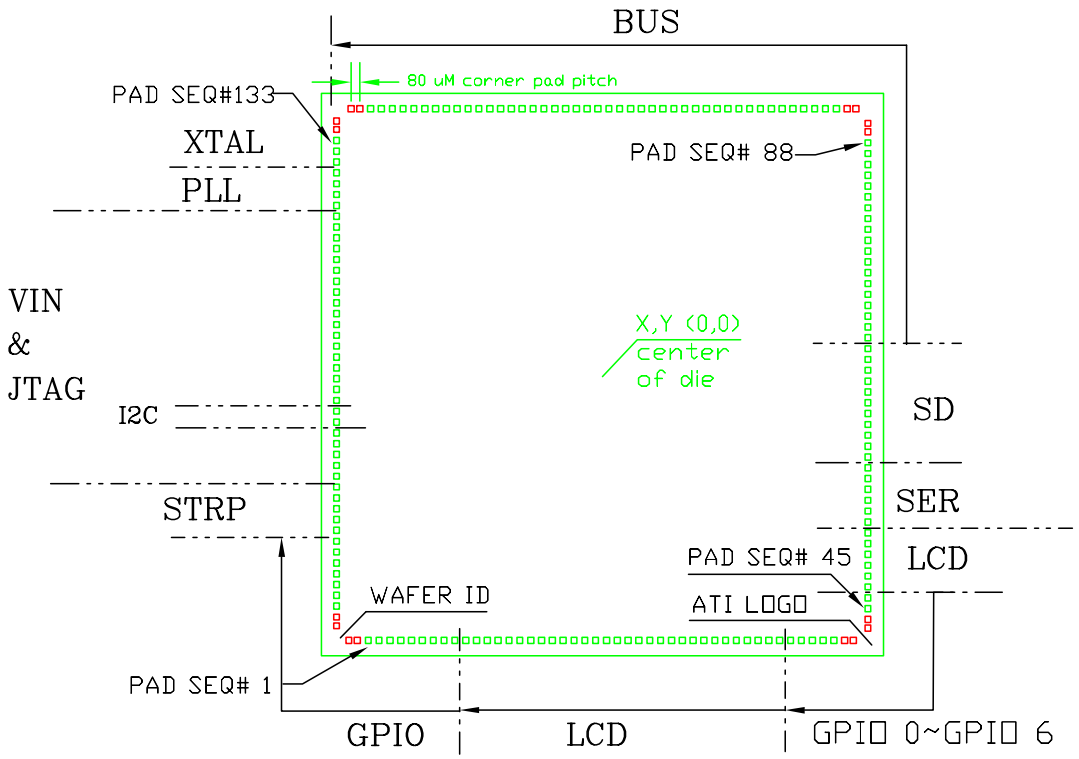

Figure 7-2 Bump Map Diagram for Flip Chip Package

Contact ATI Handheld Application Engineering for implementation details.

Appendix A
Pin Listings

A.1 W2250 Ball-out Assignment Top View

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
A		AD21	AD20	OEB	CSB	DQ15	DQ14	DQ12	DQ10	DQ8	DQ6	DQ2	INTB	BE0	AD17	AD16		A
B	PVSS	PVDD	AD19	MR	WEB	SRCCLK	DQ13	DQ9	DQ7	DQ4	DQ3	RESETB	DQ0	BE1	AD18	AD15	AD14	B
C	XTALOUT	XTALIN														AD10	AD12	C
D	VIPDATA6	VIPDATA7			AD22	AD0	DQ11				DQ5	DQ1	WAITB			AD13	AD9	D
E	VIPDATA5	VIPDATA4		VSS	VSS	VSS	VDDR1	VDDR1	VDDR1	VDDR1	VDDR1	VSS	VSS	VSS		AD11	AD7	E
F	VIPDATA3	VIPDATA2		VSS	VSS								VSS	AD6		AD8	AD5	F
G	VIPDATA0	VIPDATA1		TESTEN	VDDR4								VSS	AD1		AD3	AD4	G
H	TDO	TDI			VDDR4			VSS	VDDC	VSS			VDDR5			AD2	SDC0CD	H
J	TCLK	TMS			VDDR4			VDDC	VDDC	VDDC			VDDR5			SDC0WP	SDC0DQ1	J
K	SDA	SCL			VDDR4			VSS	VDDC	VSS			VDDR5			SDC0DQ0	SDC0CLK	K
L	VIPVSYNC	VIPCLKO		SPKG	VSS								VDDR5	SDC0DQ3		SDC0DQ2	SDC0CMD	L
M	VIPHREF	VIPCLK		SCVOLT0	VSS								VSS	VSS		CLI_CMD	CLI_CSB	M
N	GPIO19	GPIO18		SCVOLT1	VSS	VDDR2	VDDR2	VDDR2	VDDR2	VDDR2	VDDR2	VDDR2	VSS	CLI_CLK		CLI_DOUT	CLI_RESETB	N
P	GPIO17	GPIO16			VSS	LCDD12	LCDD6				LCDD0	GOE	VSS			GCLK	FRAME	P
R	GPIO15	GPIO12														DINV	SS	R
T	GPIO14	GPIO13	GPIO10	GPIO8	LCDD13	LCDD15	LCDD17	LCDD8	LCDD10	LCDD1	LCDD4	DCLK	GS	GPIO0	GPIO4	GPIO3	GPIO2	T
U		GPIO11	GPIO9	GPIO7	LCDD14	LCDD16	LCDD7	LCDD9	LCDD11	LCDD2	LCDD3	LCDD5	LS	GPIO5	GPIO1	GPIO6		U
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	

Figure A-1. W2250 Ball-out Assignment Top View

A.2 W2250 Pin Listing Sorted by Ball Reference

Table A-1 W2250 Pin Listing Sorted by Ball Reference

Ball Ref	Signal Name
A10	DQ8
A11	DQ6
A12	DQ2
A13	INTB
A14	BE0
A15	AD17
A16	AD16
A2	AD21
A3	AD20
A4	OEB
A5	CSB
A6	DQ15
A7	DQ14
A8	DQ12
A9	DQ10
B1	PVSS
B10	DQ4
B11	DQ3
B12	RESETB
B13	DQ0
B14	BE1
B15	AD18
B16	AD15
B17	AD14
B2	PVDD
B3	AD19
B4	MR
B5	WEB
B6	SRCCLK
B7	DQ13
B8	DQ9
B9	DQ7
C1	XTALOUT
C16	AD10
C17	AD12
C2	XTALIN
D1	VIPDATA6
D11	DQ5
D12	DQ1
D13	WAITB
D16	AD13
D17	AD9
D2	VIPDATA7

Table A-1 W2250 Pin Listing Sorted by Ball Reference (Continued)

Ball Ref	Signal Name
D5	AD22
D6	AD0
D7	DQ11
E1	VIPDATA5
E10	VDDR1
E11	VDDR1
E12	VSS
E13	VSS
E14	VSS
E16	AD11
E17	AD7
E2	VIPDATA4
E4	VSS
E5	VSS
E6	VSS
E7	VDDR1
E8	VDDR1
E9	VDDR1
F1	VIPDATA3
F13	VSS
F14	AD6
F16	AD8
F17	AD5
F2	VIPDATA2
F4	VSS
F5	VSS
G1	VIPDATA0
G13	VSS
G14	AD1
G16	AD3
G17	AD4
G2	VIPDATA1
G4	TESTEN
G5	VDDR4
H1	TDO
H10	VSS
H13	VDDR5
H16	AD2
H17	SDC0CD
H2	TDI
H5	VDDR4
H8	VSS
H9	VDDC
J1	TCLK
J10	VDDC

Table A-1 W2250 Pin Listing Sorted by Ball Reference (Continued)

Ball Ref	Signal Name
J13	VDDR5
J16	SDC0WP
J17	SDC0DQ1
J2	TMS
J5	VDDR4
J8	VDDC
J9	VDDC
K1	SDA
K10	VSS
K13	VDDR5
K16	SDC0DQ0
K17	SDC0CLK
K2	SCL
K5	VDDR4
K8	VSS
K9	VDDC
L1	VIPVSYNC
L13	VDDR5
L14	SDC0DQ3
L16	SDC0DQ2
L17	SDC0CMD
L2	VIPCLKO
L4	SPKG
L5	VSS
M1	VIPHREF
M13	VSS
M14	VSS
M16	CLI_CMD
M17	CLI_CSB
M2	VIPCLK
M4	SCVOLT0
M5	VSS
N1	GPIO19
N10	VDDR2
N11	VDDR2
N12	VDDR2
N13	VSS
N14	CLI_CLK
N16	CLI_DOUT
N17	CLI_RESETB
N2	GPIO18
N4	SCVOLT1
N5	VSS
N6	VDDR2
N7	VDDR2

Table A-1 W2250 Pin Listing Sorted by Ball Reference (Continued)

Ball Ref	Signal Name
N8	VDDR2
N9	VDDR2
P1	GPIO17
P11	LCDD0
P12	GOE
P13	VSS
P16	GCLK
P17	FRAME
P2	GPIO16
P5	VSS
P6	LCDD12
P7	LCDD6
R1	GPIO15
R16	DINV
R17	SS
R2	GPIO12
T1	GPIO14
T10	LCDD1
T11	LCDD4
T12	DCLK
T13	GS
T14	GPIO0
T15	GPIO4
T16	GPIO3
T17	GPIO2
T2	GPIO13
T3	GPIO10
T4	GPIO8
T5	LCDD13
T6	LCDD15
T7	LCDD17
T8	LCDD8
T9	LCDD10
U10	LCDD2
U11	LCDD3
U12	LCDD5
U13	LS
U14	GPIO5
U15	GPIO1
U16	GPIO6
U2	GPIO11
U3	GPIO9
U4	GPIO7
U5	LCDD14
U6	LCDD16

Table A-1 W2250 Pin Listing Sorted by Ball Reference (Continued)

Ball Ref	Signal Name
U7	LCDD7
U8	LCDD9
U9	LCDD11

A.3 W2250 Pin Listing Sorted by Signal Name

Table A-2 W2250 Pin Listing Sorted by Signal Name

Ball Ref	Signal Name
D6	AD0
G14	AD1
C16	AD10
E16	AD11
C17	AD12
D16	AD13
B17	AD14
B16	AD15
A16	AD16
A15	AD17
B15	AD18
B3	AD19
H16	AD2
A3	AD20
A2	AD21
D5	AD22
G16	AD3
G17	AD4
F17	AD5
F14	AD6
E17	AD7
F16	AD8
D17	AD9
A14	BE0
B14	BE1
N14	CLI_CLK
M16	CLI_CMD
M17	CLI_CSB
N16	CLI_DOUT
N17	CLI_RESETB
A5	CSB
T12	DCLK
R16	DINV
B13	DQ0
D12	DQ1
A9	DQ10
D7	DQ11
A8	DQ12
B7	DQ13
A7	DQ14
A6	DQ15
A12	DQ2
B11	DQ3

Table A-2 W2250 Pin Listing Sorted by Signal Name (Continued)

Ball Ref	Signal Name
B10	DQ4
D11	DQ5
A11	DQ6
B9	DQ7
A10	DQ8
B8	DQ9
P17	FRAME
P16	GCLK
P12	GOE
T14	GPIO0
U15	GPIO1
T3	GPIO10
U2	GPIO11
R2	GPIO12
T2	GPIO13
T1	GPIO14
R1	GPIO15
P2	GPIO16
P1	GPIO17
N2	GPIO18
N1	GPIO19
T17	GPIO2
T16	GPIO3
T15	GPIO4
U14	GPIO5
U16	GPIO6
U4	GPIO7
T4	GPIO8
U3	GPIO9
T13	GS
A13	INTB
P11	LCDD0
T10	LCDD1
T9	LCDD10
U9	LCDD11
P6	LCDD12
T5	LCDD13
U5	LCDD14
T6	LCDD15
U6	LCDD16
T7	LCDD17
U10	LCDD2
U11	LCDD3
T11	LCDD4
U12	LCDD5

Table A-2 W2250 Pin Listing Sorted by Signal Name (Continued)

Ball Ref	Signal Name
P7	LCDD6
U7	LCDD7
T8	LCDD8
U8	LCDD9
U13	LS
B4	MR
A4	OEB
B2	PVDD
B1	PVSS
B12	RESETB
K2	SCL
M4	SCVOLT0
N4	SCVOLT1
K1	SDA
H17	SDC0CD
K17	SDC0CLK
L17	SDC0CMD
K16	SDC0DQ0
J17	SDC0DQ1
L16	SDC0DQ2
L14	SDC0DQ3
J16	SDC0WP
L4	SPKG
B6	SRCCLK
R17	SS
J1	TCLK
H2	TDI
H1	TDO
G4	TESTEN
J2	TMS
H9	VDDC
J8	VDDC
J9	VDDC
J10	VDDC
K9	VDDC
E7	VDDR1
E8	VDDR1
E9	VDDR1
E10	VDDR1
E11	VDDR1
N6	VDDR2
N7	VDDR2
N8	VDDR2
N9	VDDR2
N10	VDDR2

Table A-2 W2250 Pin Listing Sorted by Signal Name (Continued)

Ball Ref	Signal Name
N11	VDDR2
N12	VDDR2
G5	VDDR4
H5	VDDR4
J5	VDDR4
K5	VDDR4
H13	VDDR5
J13	VDDR5
K13	VDDR5
L13	VDDR5
M2	VIPCLK
L2	VIPCLKO
G1	VIPDATA0
G2	VIPDATA1
F2	VIPDATA2
F1	VIPDATA3
E2	VIPDATA4
E1	VIPDATA5
D1	VIPDATA6
D2	VIPDATA7
M1	VIPHREF
L1	VIPVSYNC
E4	VSS
E5	VSS
E6	VSS
E12	VSS
E13	VSS
E14	VSS
F4	VSS
F5	VSS
F13	VSS
G13	VSS
H8	VSS
H10	VSS
K8	VSS
K10	VSS
L5	VSS
M5	VSS
M13	VSS
M14	VSS
N5	VSS
N13	VSS
P5	VSS
P13	VSS
D13	WAITB

Table A-2 W2250 Pin Listing Sorted by Signal Name (Continued)

Ball Ref	Signal Name
B5	WEB
C2	XTALIN
C1	XTALOUT

A.4 Pin Listing to Pad Picture

The following is the W2250 pin listing to pad picture.

Top side (left to right):

Group	Pad	Cell	Name	Pin
corner	uPAD_CORNER_LT	PVRGRGW1DA1	VSS	
BUS	uPAD_AD21	PGPIOCW1DA2	AD21	132
BUS	uPAD_AD19	PGPIOCW1DA2	AD19	131
BUS	uPAD_AD20	PGPIOCW1DA2	AD20	130
BUS	uPAD_AD22	PGPIOCW1DA2	AD22	129
BUS	uPAD_VDDR1_128	PVGVDRW1DA1	VDDR1	128
BUS	uPAD_VDDR1_127	PVGVDRW1DA1	VDDR1	127
BUS	uPAD_WEB	PGPIOCW1DA2	WEB	126
BUS	uPAD_OEB	PGPIOCW1DA2	OEB	125
BUS	uPAD_MR	PGPIOCW1DA2	MR	124
BUS	uPAD_CSB	PGPIOCW1DA2	CSB	123
BUS	uPAD_AD0	PGPIOCW1DA2	AD0	122
BUS	uPAD_VDD_121	PVGVDCW1DA1	VDD	121
BUS	uPAD_SRCCLK	PGPIOCW1DA2	SRCCLK	120
BUS	uPAD_DQ15	PGPIOCW1DA2	DQ15	119
BUS	uPAD_DQ14	PGPIOCW1DA2	DQ14	118
BUS	uPAD_DQ13	PGPIOCW1DA2	DQ13	117
BUS	uPAD_DQ12	PGPIOCW1DA2	DQ12	116
BUS	uPAD_VSS_115	PVGVSCW1DA1	VSS	115
BUS	uPAD_VDD_114	PVGVDCW1DA1	VDD	114
BUS	uPAD_DQ11	PGPIOCW1DA2	DQ11	113
BUS	uPAD_DQ10	PGPIOCW1DA2	DQ10	112
BUS	uPAD_DQ9	PGPIOCW1DA2	DQ9	111
BUS	uPAD_DQ8	PGPIOCW1DA2	DQ8	110
BUS	uPAD_DQ7	PGPIOCW1DA2	DQ7	109
BUS	uPAD_VSS_108	PVGVSCW1DA1	VSS	108
BUS	uPAD_DQ6	PGPIOCW1DA2	DQ6	107
BUS	uPAD_DQ5	PGPIOCW1DA2	DQ5	106
BUS	uPAD_DQ4	PGPIOCW1DA2	DQ4	105
BUS	uPAD_DQ3	PGPIOCW1DA2	DQ3	104
BUS	uPAD_DQ2	PGPIOCW1DA2	DQ2	103
BUS	uPAD_VDDR1_102	PVGVDRW1DA1	VDDR1	102
BUS	uPAD_VDDR1_101	PVGVDRW1DA1	VDDR1	101
BUS	uPAD_DQ1	PGPIOCW1DA2	DQ1	100
BUS	uPAD_DQ0	PGPIOCW1DA2	DQ0	99
BUS	uPAD_RESETB	PGPIOCW1DA2	RESETB	98
BUS	uPAD_INTB	PGPIOCW1DA2	INTB	97
BUS	uPAD_BE1	PGPIOCW1DA2	BE1	96
BUS	uPAD_VDD_95	PVGVDCW1DA1	VDD	95
BUS	uPAD_VDD_94	PVGVDCW1DA1	VDD	94
BUS	uPAD_BE0	PGPIOCW1DA2	BE0	93
BUS	uPAD_WAITB	PGPIOCW1DA2	WAITB	92
BUS	uPAD_AD18	PGPIOCW1DA2	AD18	91
BUS	uPAD_AD17	PGPIOCW1DA2	AD17	90
BUS	uPAD_AD16	PGPIOCW1DA2	AD16	89
corner	uPAD_CORNER_RT	PVRGRGW1DA1	VSS	

Left side (top to bottom):

Group	Pad	Cell	Name	Pin
corner	uPAD_CORNER_LT	PVRGRGW1DA1	VSS	
PBRK	uPAD_VDD_133	PVPBRKW1DA1	VDD	133
XTAL	uPAD_XTALIN	PGOSCLW1DA4	XTALIN	134
XTAL	uPAD_XTALIN	PGOSCLW1DA4	XTALOUT	135
PLL	uPAD_PVDD_0	PGAVDCW1DB2	PVDD	136
PLL	uPAD_PVSS_0	PGAVSCW1DA2	PVSS	137
PLL	uPAD_PVDD_1	PGAVDCW1DA2	PVDD	138
PLL	uPAD_PVSS_1	PGAVSCW1DA2	PVSS	139
VIN	uPAD_VSS_140	PVGVSCW1DA1	VSS	140
VIN	uPAD_VIPDATA7	PGPIODW1DA2	VIPDATA7	141
VIN	uPAD_VIPDATA6	PGPIODW1DA2	VIPDATA6	142
VIN	uPAD_VIPDATA5	PGPIODW1DA2	VIPDATA5	143
VIN	uPAD_VIPDATA4	PGPIODW1DA2	VIPDATA4	144
VIN	uPAD_VDD_145	PVGVDCW1DA1	VDD	145
VIN	uPAD_VIPDATA3	PGPIODW1DA2	VIPDATA3	146
VIN	uPAD_VIPDATA2	PGPIODW1DA2	VIPDATA2	147
VIN	uPAD_VIPDATA1	PGPIODW1DA2	VIPDATA1	148
VIN	uPAD_VIPDATA0	PGPIODW1DA2	VIPDATA0	149
JTAG	uPAD_VDDR4_150	PVGVDRW1DB1	VDDR4	150
JTAG	uPAD_VDDR4_151	PVGVDRW1DB1	VDDR4	151
JTAG	uPAD_TDI	PGPIODW1DA2	TDI	152
JTAG	uPAD_TDO	PGPIODW1DA2	TDO	153
JTAG	uPAD_TESTEN	PGPIODW1DA2	TESTEN	154
JTAG	uPAD_VSS_155	PVGVSCW1DA1	VSS	155
JTAG	uPAD_TCLK	PGPIODW1DA2	TCLK	156
VIN	uPAD_VIPCLKO	PGPIODW1DA2	VIPCLKO	157
I2C	uPAD_SCL	PGPIODW1DA2	SCL	158
I2C	uPAD_SDA	PGPIODW1DA2	SDA	159
VIN	uPAD_VDDR4_160	PVGVDRW1DB1	VDDR4	160
JTAG	uPAD_TMS	PGPIODW1DA2	TMS	161
VIN	uPAD_VIPVSYNC	PGPIODW1DA2	VIPVSYNC	162
VIN	uPAD_VIPHREF	PGPIODW1DA2	VIPHREF	163
VIN	uPAD_VIPCLK	PGPIODW1DA2	VIPCLK	164
PBRK	uPAD_VDD_165	PVPBRKW1DA1	VDD	165
STRP	uPAD_SPKG	PGPIODW1DA2	SPKG	166
STRP	uPAD_SDISSDC	PGPIODW1DA2	SDISSDC	167
STRP	uPAD_SCVOLT0	PGPIODW1DA2	SCVOLT0	168
STRP	uPAD_SCVOLT1	PGPIODW1DA2	SCVOLT1	169
GPIO	uPAD_VSS_170	PVGVSCW1DA1	VSS	170
GPIO	uPAD_GPIO19	PGPIODW1DA2	GPIO19	171
GPIO	uPAD_GPIO18	PGPIODW1DA2	GPIO18	172
GPIO	uPAD_VDDR2_173	PVGVDRW1DA1	VDDR2	173
GPIO	uPAD_GPIO17	PGPIODW1DA2	GPIO17	174
GPIO	uPAD_GPIO16	PGPIODW1DA2	GPIO16	175
GPIO	uPAD_GPIO15	PGPIODW1DA2	GPIO15	176
corner	uPAD_CORNER_LB	PVRGRGW1DA1	VSS	

Right side (top to bottom):

Pin	Name	Cell	Pad	Group
	VSS	PVRGRGW1DA1	uPAD_CORNER_RT	corner
88	AD15	PGPIOCW1DA2	uPAD_AD15	BUS
87	AD14	PGPIOCW1DA2	uPAD_AD14	BUS
86	AD13	PGPIOCW1DA2	uPAD_AD13	BUS
85	AD12	PGPIOCW1DA2	uPAD_AD12	BUS
84	AD11	PGPIOCW1DA2	uPAD_AD11	BUS
83	VDDR1	PVGVDRW1DA1	uPAD_VDDR1_83	BUS
82	VDDR1	PVGVDRW1DA1	uPAD_VDDR1_82	BUS
81	AD10	PGPIOCW1DA2	uPAD_AD10	BUS
80	AD9	PGPIOCW1DA2	uPAD_AD9	BUS
79	AD8	PGPIOCW1DA2	uPAD_AD8	BUS
78	AD7	PGPIOCW1DA2	uPAD_AD7	BUS
77	AD6	PGPIOCW1DA2	uPAD_AD6	BUS
76	VSS	PVGVSCW1DA1	uPAD_VSS_76	BUS
75	VDD	PVGVDCW1DA1	uPAD_VDD_75	BUS
74	AD5	PGPIOCW1DA2	uPAD_AD5	BUS
73	AD4	PGPIOCW1DA2	uPAD_AD4	BUS
72	AD3	PGPIOCW1DA2	uPAD_AD3	BUS
71	AD2	PGPIOCW1DA2	uPAD_AD2	BUS
70	AD1	PGPIOCW1DA2	uPAD_AD1	BUS
69	VSS	PVGBRKW1DA1	uPAD_VSS_69	PBRK
68	SDC0WP	PGPIODW1DA2	uPAD_SDC0WP	SD
67	SDC0CD	PGPIODW1DA2	uPAD_SDC0CD	SD
66	SDC0DQ1	PGPIODW1DA2	uPAD_SDC0DQ1	SD
65	SDC0DQ0	PGPIODW1DA2	uPAD_SDC0DQ0	SD
64	VDDR5	PVGVDRW1DA1	uPAD_VDDR5_64	SD
63	VDDR5	PVGVDRW1DA1	uPAD_VDDR5_63	SD
62	SDC0CLK	PGPIODW1DA2	uPAD_SDC0CLK	SD
61	SDC0CMD	PGPIODW1DA2	uPAD_SDC0CMD	SD
60	SDC0DQ3	PGPIODW1DA2	uPAD_SDC0DQ3	SD
59	SDC0DQ2	PGPIODW1DA2	uPAD_SDC0DQ2	SD
58	VSS	PVGBRKW1DA1	uPAD_VSS_58	PBRK
57	CLI_CSB	PGPIODW1DA2	uPAD_CLI_CSB	SER
56	CLI_RESETB	PGPIODW1DA2	uPAD_CLI_RESETB	SER
55	CLI_CMD	PGPIODW1DA2	uPAD_CLI_CMD	SER
54	CLI_CLK	PGPIODW1DA2	uPAD_CLI_CLK	SER
53	CLI_DOUT	PGPIODW1DA2	uPAD_CLI_DOUT	SER
52	VDDR2	PVGVDRW1DA1	uPAD_VDDR2_52	LCD
51	FRAME	PGPIODW1DA2	uPAD_FRAME	LCD
50	GCLK	PGPIODW1DA2	uPAD_GCLK	LCD
49	SS	PGPIODW1DA2	uPAD_SS	LCD
48	DINV	PGPIODW1DA2	uPAD_DINV	LCD
47	VDD	PVGVDCW1DA1	uPAD_VDD_47	LCD
46	GPIO2	PGPIODW1DA2	uPAD_GPIO2	GPIO
45	GPIO3	PGPIODW1DA2	uPAD_GPIO3	GPIO
	VSS	PVRGRGW1DA1	uPAD_CORNER_RB	corner

Bottom side (left to right):

Pin	Name	Cell	Pad	Group
	VSS	PVRGRGW1DA1	uPAD_CORNER_LB	corner
1	GPIO14	PGPIODW1DA2	uPAD_GPIO14	GPIO
2	GPIO13	PGPIODW1DA2	uPAD_GPIO13	GPIO
3	GPIO12	PGPIODW1DA2	uPAD_GPIO12	GPIO
4	GPIO11	PGPIODW1DA2	uPAD_GPIO11	GPIO
5	GPIO10	PGPIODW1DA2	uPAD_GPIO10	GPIO
6	VDDR2	PVGVDRW1DA1	uPAD_VDDR2_6	GPIO
7	GPIO9	PGPIODW1DA2	uPAD_GPIO9	GPIO
8	GPIO8	PGPIODW1DA2	uPAD_GPIO8	GPIO
9	GPIO7	PGPIODW1DA2	uPAD_GPIO7	GPIO
10	LCDD12	PGPIODW1DA2	uPAD_LCDD12	LCD
11	LCDD13	PGPIODW1DA2	uPAD_LCDD13	LCD
12	VDD	PVGVDCW1DA1	uPAD_VDD_12	GPIO
13	LCDD14	PGPIODW1DA2	uPAD_LCDD14	LCD
14	LCDD15	PGPIODW1DA2	uPAD_LCDD15	LCD
15	LCDD16	PGPIODW1DA2	uPAD_LCDD16	LCD
16	LCDD17	PGPIODW1DA2	uPAD_LCDD17	LCD
17	LCDD6	PGPIODW1DA2	uPAD_LCDD6	LCD
18	VSS	PVGVSCW1DA1	uPAD_VSS_18	GPIO
19	VDD	PVGVDCW1DA1	uPAD_VDD_19	GPIO
20	LCDD7	PGPIODW1DA2	uPAD_LCDD7	LCD
21	LCDD8	PGPIODW1DA2	uPAD_LCDD8	LCD
22	LCDD9	PGPIODW1DA2	uPAD_LCDD9	LCD
23	LCDD10	PGPIODW1DA2	uPAD_LCDD10	LCD
24	LCDD11	PGPIODW1DA2	uPAD_LCDD11	LCD
25	VDDR2	PVGVDRW1DA1	uPAD_VDDR2_25	LCD
26	VDDR2	PVGVDRW1DA1	uPAD_VDDR2_26	LCD
27	LCDD0	PGPIODW1DA2	uPAD_LCDD0	LCD
28	LCDD1	PGPIODW1DA2	uPAD_LCDD1	LCD
29	LCDD2	PGPIODW1DA2	uPAD_LCDD2	LCD
30	LCDD3	PGPIODW1DA2	uPAD_LCDD3	LCD
31	VSS	PVGVSCW1DA1	uPAD_VSS_31	LCD
32	LCDD4	PGPIODW1DA2	uPAD_LCDD4	LCD
33	LCDD5	PGPIODW1DA2	uPAD_LCDD5	LCD
34	GOE	PGPIODW1DA2	uPAD_GOE	LCD
35	VDD	PVGVDCW1DA1	uPAD_VDD_35	LCD
36	DCLK	PGPIODW1DA2	uPAD_DCLK	LCD
37	LS	PGPIODW1DA2	uPAD_LS	LCD
38	GS	PGPIODW1DA2	uPAD_GS	LCD
39	VDDR2	PVGVDRW1DA1	uPAD_VDDR2_39	LCD
40	GPIO5	PGPIODW1DA2	uPAD_GPIO5	GPIO
41	GPIO4	PGPIODW1DA2	uPAD_GPIO4	GPIO
42	GPIO1	PGPIODW1DA2	uPAD_GPIO1	GPIO
43	GPIO6	PGPIODW1DA2	uPAD_GPIO6	GPIO
44	GPIO0	PGPIODW1DA2	uPAD_GPIO0	GPIO
	VSS	PVRGRGW1DA1	uPAD_CORNER_RB	corner

Figure A-2. W2250 Pin Listing to Pad Picture

Appendix B
Reference Schematics

B.1 Reference Schematics

To be supplied as a separate document, depending on the customer's need.

This page intentionally left blank.

Appendix C
GPIO Assignments

GPIO pins are multiplexed with LCD interface pins (*Table 3-12, "LCD Interface and GPIO Pin Multiplexing," on page 3-13*), CLI interface pins, JTAG pins (*Table 5-9, "JTAG Pins (powered on VDDR4)," on page 5-4*), and straps (*Table 5-13, "W2250 Straps," on page 5-4*).

Table C-1 Recommended GPIO Assignments

Pin Name	Recommended GPIO Assignment
GPIO 0	LCD power saving (PS)
GPIO 1	LCD reverse; REV (Vcom or POL)
GPIO 2	Spare
GPIO 3	Spare
GPIO 4	Spare
GPIO 5	Spare
GPIO 6	Spare
GPIO 7	CLI_CSB
GPIO 8	Panel revision ID
GPIO 9	Spare
GPIO 10	Panel revision ID or Secondary display ID
GPIO 11	Pulse width modulation
GPIO 12	Secondary display ID
GPIO 13	Spare
GPIO 14	SD Power On/Off
GPIO 15	Spare
GPIO 16	Spare
GPIO 17	Primary display ID
GPIO 18	Spare
GPIO 19	Primary display ID
TCLK/GPIO 20	CAM_PWNDN
TMS/GPIO 21	CAM_RESET
TDI/GPIO 22	Spare
TDO/GPIO 23	Spare

This page intentionally left blank.

Appendix D
Design Considerations

D.1 General Design Rules

To lower noise and ground bounce on the digital power sections, be sure to do the following:

- Keep the power traces as short and as wide as possible
- Place the decoupling caps for each rail as close as possible to the ASIC
- You can connect at any point the traces for each power rail (VSS, VDDC, VDDR1, VDDR2, VDDR3, VDDR4, VDDR5), but it is recommended that this be done as close as possible to the pad.

D.2 Correct Placement of PLL Decoupling Capacitor

The following diagram illustrates the correct placement of the PLL decoupling capacitor.

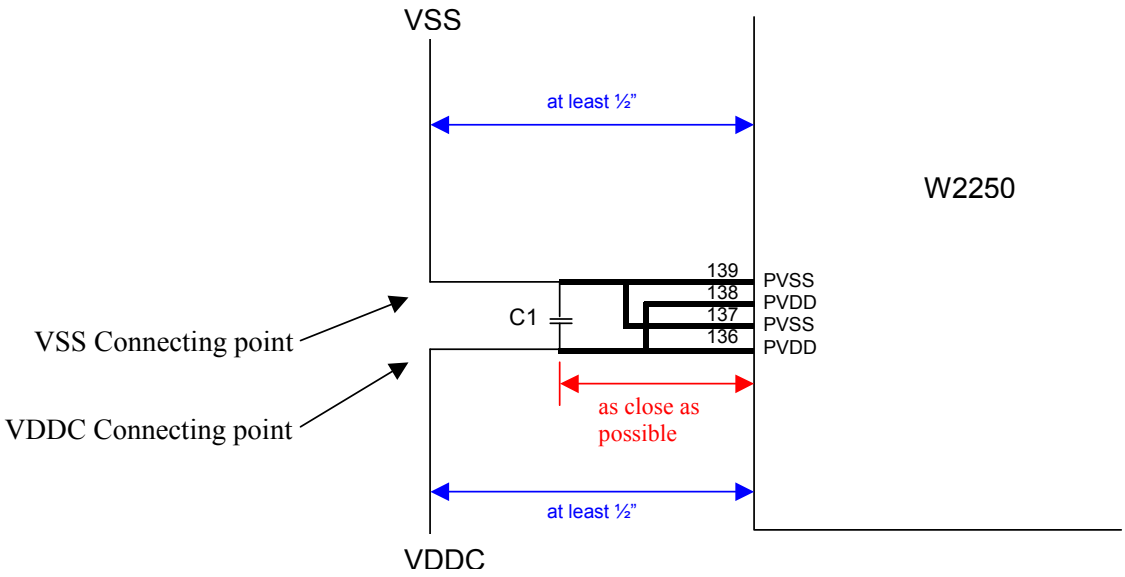

Figure D-1. Decoupling Capacitor Placement

- Connect the PVDD pins to VDDC and the PVSS pins to VSS at least ½" from the die or BGA.
- Hook up the decoupling capacitor, C1, as close as possible to the PVSS and PVDD pads.
- C1 can be as large as 0.1µF, depending on the design.

This page intentionally left blank.