PrimeCell® Color LCD Controller (PL111)

Revision: r0p2

Technical Reference Manual

ARM®

ARM DDI 0293C

PrimeCell Color LCD Controller (PL111)
Technical Reference Manual

Release Information

The following changes have been made to this document.

Change history

Date	Issue	Confidentiality	Change
14 October 2003	A	Non-Confidential	First release.
24 July 2006	B	Non-Confidential	Update to r0p1. Corrections to: • Table 2-3 on page 2-14 • Table A-7 on page A-6 • Table A-9 on page A-8. Improved explanations for: • pixels-per-line in Table 3-2 on page 3-5 • clocks-per-line in Table 3-4 on page 3-8.
22 December 2006	C	Non-Confidential	First release for r0p2.

Proprietary Notice

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Product Status

The information in this document is final, that is for a developed product.

Web Address

`http://www.arm.com`

Contents
PrimeCell Color LCD Controller (PL111) Technical Reference Manual

List of Tables
PrimeCell Color LCD Controller (PL111) Technical Reference Manual

List of Figures
PrimeCell Color LCD Controller (PL111) Technical Reference Manual

Preface

This preface introduces the *PrimeCell Color LCD Controller (PL111) Technical Reference Manual*. It contains the following sections:

About this manual

This is the *Technical Reference Manual* (TRM) for the *PrimeCell* (PL111) *Color LCD Controller* (CLCDC).

Product revision status

The r*n*p*n* identifier indicates the revision status of the product described in this manual, where:

r*n* Identifies the major revision of the product.

p*n* Identifies the minor revision or modification status of the product.

Intended audience

This manual is written for hardware and software engineers who have some experience of ARM products. Prior experience of the controller is not assumed.

Organization

This manual is organized into the following chapters:

Chapter 1 *Introduction*

Read this chapter for an introduction to the controller and its features.

Chapter 2 *Functional Overview*

Read this chapter for a description of the major functional blocks of the controller.

Chapter 3 *Programmer's Model*

Read this chapter for a description of the registers and programming details.

Chapter 4 *Programmer's Model for Test*

Read this chapter for a description of the additional logic for functional verification and production testing.

Appendix A *Signal Descriptions*

Read this appendix for a description of the signals.

Glossary Read the Glossary for definitions of terms used in this manual.

Conventions

Conventions that this manual can use are described in:

- *Typographical*
- *Timing diagrams*
- *Signals* on page xiv
- *Numbering* on page xiv.

Typographical

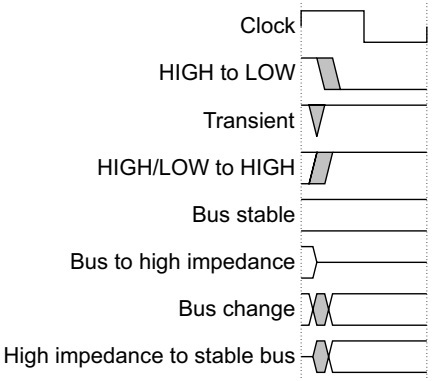
The typographical conventions are:

italic	Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`mono`space	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`monospace italic`	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.
< and >	Angle brackets enclose replaceable terms for assembler syntax where they appear in code or code fragments. They appear in normal font in running text. For example: • `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>` • The Opcode_2 value selects the register that is accessed.

Timing diagrams

The figure named *Key to timing diagram conventions* on page xiv explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.



Key to timing diagram conventions

Signals

The signal conventions are:

Signal level The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means HIGH for active-HIGH signals and LOW for active-LOW signals.

Lower-case n Denotes an active-LOW signal.

Prefix H Denotes *Advanced High-performance Bus* (AHB) signals.

Prefix P Denotes *Advanced Peripheral Bus* (APB) signals.

Numbering

The numbering convention is:

<size in bits>'<base><number>

This is a Verilog® method of abbreviating constant numbers. For example:

- 'h7B4 is an unsized hexadecimal value.
- 'o7654 is an unsized octal value.
- 8'd9 is an eight-bit wide decimal value of 9.

- 8’h3F is an eight-bit wide hexadecimal value of `0x3F`. This is equivalent to b00111111.
- 8’b1111 is an eight-bit wide binary value of b00001111.

Further reading

This section lists publications by ARM, and by third parties.

ARM periodically provides updates and corrections to its documentation. See `http://www.arm.com` for current errata sheets, addenda, and the Frequently Asked Questions list.

ARM publications

This manual contains information that is specific to the CLCDC. See the following documents for other relevant information:

- *AMBA® Specification (Rev 2.0)* (ARM IHI 0011)
- *ARM PrimeCell CLCDC PL111 Design Manual* (PL111 DDES 0000)
- *ARM PrimeCell CLCDC PL111 Integration Manual* (PL111 INTM 0000).

Feedback

ARM welcomes feedback on the CLCDC and its documentation.

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier giving:

- the product name
- a concise explanation of your comments.

Feedback on this manual

If you have any comments on this manual, send e-mail to `errata@arm.com` giving:

- the title
- the number
- the relevant page number(s) to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

This chapter introduces the CLCDC. It contains the following sections:

1.1 About the controller

The controller is an *Advanced Microcontroller Bus Architecture* (AMBA) master-slave module that connects to the *Advanced High-performance Bus* (AHB). It is an AMBA-compliant *System-on-Chip* (SoC) peripheral that is developed, tested, and licensed by ARM.

The controller is a reusable soft-IP block that has been developed with the principal aim of reducing time-to-market for *Application-Specific Integrated Circuit* (ASIC) development.

The controller provides all of the necessary control signals to interface directly to a variety of color and monochrome LCD panels.

1.1.1 Features of the controller

The principal features of the controller are:

- compliance to the *AMBA Specification (Rev 2.0)* for easy integration into SoC implementation
- 64-bit master AHB interface to access frame buffer
- dual 16-deep programmable 64-bit wide FIFOs for buffering incoming display data
- it supports single- and dual-panel mono *Super Twisted Nematic* (STN) displays with 4 or 8-bit interfaces
- it supports single- and dual-panel color STN displays
- it supports *Thin Film Transistor* (TFT) color displays
- resolution programmable up to 1024x768
- hardware cursor support for single-panel displays
- 15 gray-level mono, 3375 color STN, and 32K color palettized TFT support
- 1, 2, or 4 *bits-per-pixel* (bpp) palettized displays for mono STN
- 1, 2, 4, or 8bpp palettized color displays for color STN and TFT
- 16bpp true-color nonpalettized, for color STN and TFT
- 24bpp true-color nonpalettized, for color TFT
- programmable timing for different display panels

- 256 entry, 16-bit palette RAM, arranged as a 128x32-bit RAM physically
- frame, line, and pixel clock signals
- AC bias signal for STN, data enable signal for TFT panels
- patented gray scale algorithm
- supports little and big-endian, and Windows® CE data formats.

1.1.2 Programmable parameters

You can program the following key parameters:

- horizontal front and back porch
- horizontal synchronization pulse width
- number of pixels per line
- vertical front and back porch
- vertical synchronization pulse width
- number of lines per panel
- number of pixel clocks per line
- hardware cursor control
- signal polarity, active HIGH or LOW
- AC panel bias
- panel clock frequency
- bits-per-pixel
- display type, mono STN, color STN, or TFT
- STN 4 or 8-bit interface mode
- STN single- or dual-panel mode
- little-endian, big-endian, or Windows CE mode
- interrupt generation event.

1.1.3 Target markets

The controller addresses primarily the portable segment of the market. Typical applications for the controller include:

- *Personal Digital Assistants* (PDAs)
- ultrasub notebook computers
- smartphones
- handheld, portable color games terminals.

1.1.4 LCD panel resolution

You can program the controller to support a wide range of panel resolutions such as:

- 320x200, 320x240
- 640x200, 640x240, 640x480
- 800x600
- 1024x768.

1.1.5 Hardware cursor support

The hardware cursor extension reduces software overheads associated with maintaining a cursor image in the controller frame buffer.

Before the hardware cursor was introduced, moving the cursor required software to:

- save an image of the area under the next cursor position
- update the area with the cursor image
- repair the last cursor position with a previously saved image.

In addition, the LCD driver had to check whether the graphics operation had overwritten the cursor, and correct it. With a cursor size of 64x64 and 24-bit color, each cursor move involved reading and writing approximately 75KB of data.

The hardware cursor removes the requirement for this management by providing a completely separate image buffer for the cursor, and superimposing the cursor image on the LCD output stream at the current cursor (X,Y) coordinate.

To move the hardware cursor, the software driver supplies a new cursor coordinate. The frame buffer requires no modification. This significantly reduces the software overhead.

The cursor image is held in the controller in a 256x32-bit dual-port RAM. You program the contents using the AHB slave interface.

1.1.6 Types of LCD panel supported

The controller supports the following types of LCD panel:

- active matrix TFT panels with up to 24-bit bus interface
- single-panel monochrome STN panels (4-bit and 8-bit bus interface)
- dual-panel monochrome STN panels (4-bit and 8-bit bus interface per panel)
- single-panel color STN panels, 8-bit bus interface
- dual-panel color STN panels, 8-bit bus interface per panel.

1.1.7 Number of colors supported

The following sections describe the number of colors supported by the different types of panels:

- *TFT panels*
- *Color STN panels* on page 1-6
- *Mono STN panels* on page 1-6.

TFT panels

TFT panels support one or more of the following color modes:

- 1bpp, palettized, 2 colors selected from available colors.
- 2bpp, palettized, 4 colors selected from available colors.
- 4bpp, palettized, 16 colors selected from available colors.
- 8bpp, palettized, 256 colors selected from available colors.
- 12bpp, direct 4:4:4 *Red-Green-Blue* (RGB).
- 16bpp, direct 5:5:5 RGB, with 1bpp not normally used. This pixel is still output, and you can use it as a bright bit to connect to the *Least Significant Bit* (LSB) of RGB components of a 6:6:6 TFT panel.
- 16bpp, direct 5:6:5 RGB.
- 24bpp, direct 8:8:8 RGB, providing over 16 million colors.

Each 16-bit palette entry is composed of 5bpp (RGB), plus a common intensity bit. This provides better memory utilization and performance compared with a full 6bpp structure. The total number of colors supported can be doubled from 32K to 64K if you use the intensity bit and apply it to all three color components simultaneously. See *LCD panel signal multiplexing details* on page A-8 for more information.

Alternatively, you can use the 16 signals to drive a 5:6:5 panel, and only apply the extra bit to the green channel.

Color STN panels

Color STN panels support one or more of the following color modes:

- 1bpp, palettized, 2 colors selected from 3375
- 2bpp, palettized, 4 colors selected from 3375
- 4bpp, palettized, 16 colors selected from 3375
- 8bpp, palettized, 256 colors selected from 3375
- 16bpp, direct 4:4:4 RGB, with 4bpp not being used.

Mono STN panels

Mono STN panels support one or more of the following modes:

- 1bpp, palettized, 2 gray scales selected from 15
- 2bpp, palettized, 4 gray scales selected from 15
- 4bpp, palettized, 16 gray scales selected from 15.

You can program more than 4bpp for mono panels, but using these modes has no benefit because the maximum number of gray scales supported on the display is 15.

1.1.8 LCD powering up and powering down sequence support

The controller requires the following power-up sequence to be performed:

1. V_{dd} is simultaneously applied to the SoC that contains the controller and panel display driver logic. The following signals are held LOW:
   - **CLLP**
   - **CLCP**
   - **CLFP**
   - **CLAC**
   - **CLD[23:0]**
   - **CLLE**.
2. When V_{dd} is stabilized, a 1 is written to the LcdEn bit in the LCDControl Register. See *LCD Control Register* on page 3-12. This enables the following signals into their active states:
   - **CLLP**
   - **CLCP**
   - **CLFP**
   - **CLAC**
   - **CLLE**.

   The **CLD[23:0]** signals remain in an inactive state.

3. When the signals in step 2 have stabilized, where appropriate, the contrast voltage, V_{ee} (not controlled or supplied by the controller) is then applied.

4. If required, you can use a software timer routine to provide the minimum display specific delay time between application of the control signals and power to the panel display. On completion of the software timer routine, power is applied to the panel by writing a 1 to the LcdPwr bit in the LcdControl Register that, in turn, sets the **CLPOWER** signal HIGH and enables the **CLD[23:0]** signals into their active states. Normally, **CLPOWER** is used to gate the power to the LCD panel.

The power-down sequence is the reverse of the above four stages and you must follow it strictly, this time, writing the respective register bits with 0.

Figure 1-1 shows the power-up and power-down sequences.

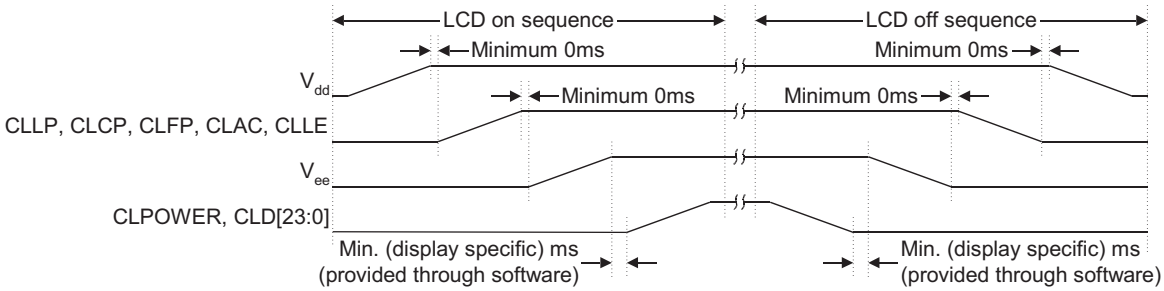


Figure 1-1 Power-up and power-down sequences

1.2 Product revisions

This section describes differences in functionality between product revisions:

r0p0-r0p1 Contains the following differences in functionality:

- CLCDPeriphID2 Register bits [7:4] now read back as `0x1`. See *Peripheral Identification Register 2* on page 3-29.
- There is no change to the functionality described in this manual. See the engineering errata that accompanies the product deliverables for more information.

r0p1-r0p2 Contains the following differences in functionality:

- CLCDPeriphID2 Register bits [7:4] now read back as `0x2`. See *Peripheral Identification Register 2* on page 3-29.
- There is no change to the functionality described in this manual. See the engineering errata that accompanies the product deliverables for more information.

Chapter 2
Functional Overview

This chapter describes the major functional blocks of the CLCDC. It contains the following sections:

2.1 Controller overview

The controller performs translation of pixel-coded data into the required formats and timings to drive a variety of single- or dual-panel mono and color LCDs.

Packets of pixel-coded data are fed using the AHB interface, to two independent, programmable, 32-bit wide, DMA FIFOs that act as input data flow buffers.

The buffered pixel coded-data is then unpacked using a pixel serializer.

Depending on the LCD type and mode, the unpacked data can represent:

- an actual true display gray or color value
- an address to a 256x16-bit wide palette RAM gray or color value.

In the case of STN displays, either a value obtained from the addressed palette location, or the true value is passed to the gray scaling generators. The hardware-coded gray scale algorithm logic sequences the addressed pixels' activity over a programmed number of frames to provide the effective display appearance.

For TFT displays, either an addressed palette value or true color value is passed directly to the output display drivers, bypassing the gray scaling algorithmic logic.

In addition to data formatting, the controller provides a set of programmable display control signals, including:

- LCD panel power enable
- pixel clock
- horizontal and vertical synchronization pulses
- display bias.

The controller generates individual interrupts for:

- upper or lower panel DMA FIFO underflow
- base address update signification
- vertical compare
- bus error.

There is also a single combined interrupt that is asserted when any of the individual interrupts become active.

Figure 2-1 on page 2-3 shows a simplified block diagram of the CLCDC.

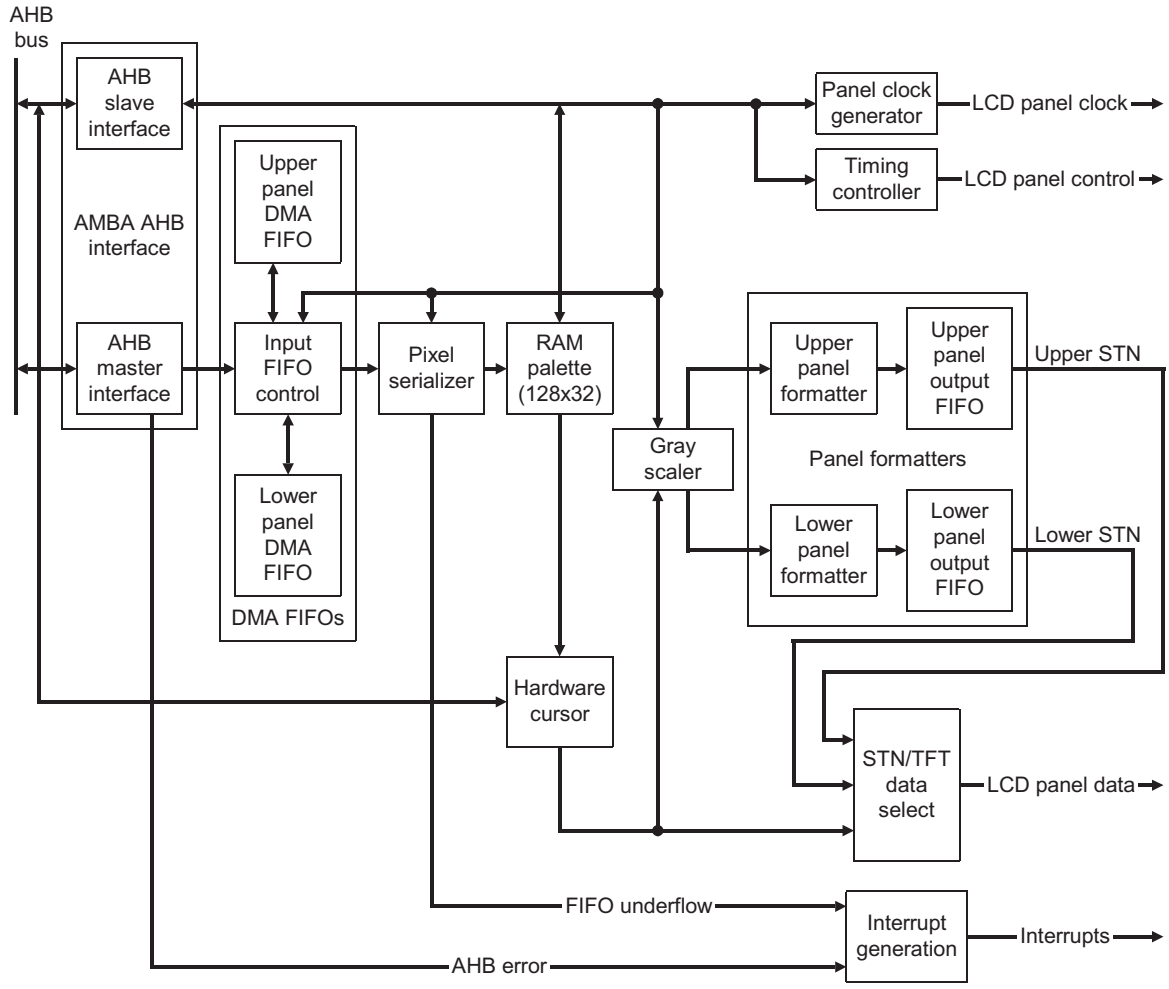


Figure 2-1 CLCDC block diagram

2.2 AHB interfaces

The following sections describe the AHB interfaces and bus architecture:

- *AHB slave interface*
- *AHB master interface*
- *Bus architecture* on page 2-5.

2.2.1 AHB slave interface

The AHB slave interface connects the controller to the AHB bus and provides CPU accesses to the registers, and palette RAM. See the *AMBA Specification (Rev 2.0)* for more information about AHB slave interfaces.

The AHB slave interface supports the following features:

- standard write and read AHB accesses
- INCR4, INCR8, and undefined length WORD bursts only
- 32-bit data interface
- OKAY response only.

Note

WRAP bursts produce unpredictable behavior.

2.2.2 AHB master interface

The AHB master interface transfers display data from a selected slave (memory) to the internal DMA FIFOs. This bus has a 64-bit data path, and is read-only. You can connect it directly to the AHB system bus, or to the AHB interface of a memory controller, such as an SDRAM controller.

In dual-panel mode, the DMA FIFOs are filled up in an alternating fashion from a single **HBUSREQM** request, and subsequent **HGRANTM**. In single-panel mode, the DMA FIFOs are filled up in a sequential fashion from a single **HBUSREQM** request and subsequent **HGRANTM**.

The inherent AHB master interface state machine performs the following functions:

- Loads the upper panel base address into the AHB address incrementer on recognition of a new frame.
- Monitors the upper and lower DMA FIFO levels and asserts **HBUSREQM** to request display data from memory, filling them to above the programmed watermark. **HBUSREQM** is reasserted when there are at least four locations available in either FIFO (dual-panel mode).

- Checks for 1KB boundaries during fixed-length bursts, appropriately adjusting the address in such occurrences.
- Generates the address sequences for fixed-length and undefined bursts.
- Controls the handshaking between the memory and DMA FIFOs. It inserts BUSY cycles if the FIFOs have not completed their synchronization and updating sequence.
- Fills up the DMA FIFOs, in dual-panel mode, in an alternating fashion from a single **HBUSREQM** request, and subsequent **HGRANTM**.
- Asserts the **CLCDMBEINTR** interrupt if an error occurs during an active burst.
- Responds to retry commands by restarting the failed access. This introduces some BUSY cycles while it resynchronizes.

2.2.3 Bus architecture

The controller incorporates a master interface and you can connect it directly onto the main system AHB bus, or alternatively to an AHB interface of a memory controller, such as an SDRAM controller.

In addition to the AHB master interface, there is an AHB slave interface for programming registers in the controller. The slave interface and the master interface are separate AHB slaves and masters. This means that you can connect the controller in one of two ways:

- You can build it so that the master interface and the slave interface connect to a single multi-master 32-bit AHB bus interface. This requires the following additional components:
  - — A downsizer to convert the 64 bit interface to 32 bits. This results in some performance degradation.
  - — An AHB-Lite to AHB bridge.
- The master interface can connect directly to a memory controller, such as an SDRAM controller, with an AHB slave interface. The slave interface connects to the AHB bus.

Figure 2-2 on page 2-6 and Figure 2-3 on page 2-6 show these two arrangements.

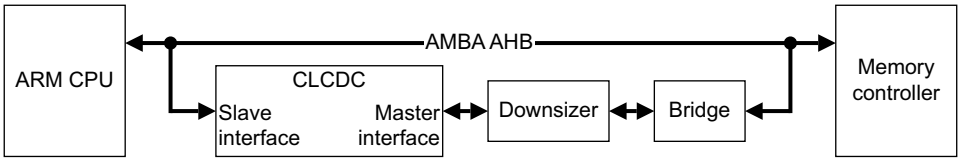


Figure 2-2 Single-bus AHB architecture

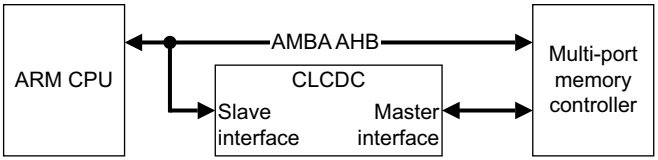


Figure 2-3 Dual-bus AHB architecture

AHB supports a wide range of on-chip bus sizes, from eight bits, up to 1024 bits. The master and slave interfaces are implemented as 64-bit and 32-bit data bus devices respectively.

2.3 Dual DMA FIFOs and associated control logic

Two DMA FIFOs buffer the pixel data accessed from memory two DMA FIFOs that can be independently controlled to cover single- and dual-panel LCD types. Each FIFO is 16 words deep by 32 bits wide and can be cascaded to form an effective 32-word deep FIFO in single-panel mode. The input to the FIFOs is 64-bit, and the output to the pixel serializer is 32-bit. Each FIFO value is output in two stages, first bits [31:0] followed by bits [63:32]. Endianness setting do not affect this behavior, and it assumes a word-invariant memory system.

Synchronization logic transfers the pixel data from the AHB **HCLK** domain to the **CLCDCLK** clock domain, and **HCLK** clocks the DMA FIFOs.

The water level marks in each FIFO are set so that each FIFO requests data when at least four locations become available.

An interrupt signal is asserted if an attempt is made to read either of the two DMA FIFOs when they are empty. In other words, an underflow condition has occurred.

2.4 Pixel serializer

This block reads the 32-bit wide LCD data from the output interface of the DMA FIFO and extracts 24, 16, 8, 4, 2, or 1bpp data, depending on the current mode of operation. The controller supports big-endian, little-endian, and Windows CE data formats. Data is always read from the DMA FIFO one word at a time, first bits [31:0] followed by bits [63:32].

In dual-panel mode, words are alternately read from the upper and lower DMA FIFOs to interleave the two separate data streams that are later split out to their respective panels.

Depending on the mode of operation, you can use the extracted data to point to a color or gray scale value in the palette RAM, or you can actually apply a true color value to an LCD panel input.

Figure 2-4 to Figure 2-9 on page 2-11, and Table 2-1 on page 2-11 show the structure of the data in each DMA FIFO word corresponding to the endianness and bpp combinations. For each of the three supported data formats, you must extract the required data for each panel display pixel from the data word.

The nomenclature that the figures use is:

- *Little-endian Byte, Little-endian Pixel* (LBLP) order.
- *Big-endian Byte, Big-endian Pixel* (BBBP) order.
- *Little-endian Byte, Big-endian Pixel* (LBBP) order. This is the Windows CE format.

bpp	DMA FIFO output bits															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	p31	p30	p29	p28	p27	p26	p25	p24	p23	p22	p21	p20	p19	p18	p17	p16
2	p15		p14		p13		p12		p11		p10		p9		p8	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p7				p6				p5				p4			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p3								p2							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p1															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24	p0															
									23	22	21	20	19	18	17	16

Figure 2-4 LBLP, DMA FIFO output bits [31:16]

bpp	DMA FIFO output bits															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	p15	p14	p13	p12	p11	p10	p9	p8	p7	p6	p5	p4	p3	p2	p1	p0
2	p7		p6		p5		p4		p3		p2		p1		p0	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p3				p2				p1				p0			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p1								p0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 2-5 LBLP, DMA FIFO output bits [15:0]

bpp	DMA FIFO output bits															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15
2	p0		p1		p2		p3		p4		p5		p6		p7	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p0				p1				p2				p3			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p0								p1							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24	p0															
									23	22	21	20	19	18	17	16

Figure 2-6 BBBP, DMA FIFO output bits [31:16]

bpp	DMA FIFO output bits															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	p16	p17	p18	p19	p20	p21	p22	p23	p24	p25	p26	p27	p28	p29	p30	p31
2	p8		p9		p10		p11		p12		p13		p14		p15	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p4				p5				p6				p7			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p2								p3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p1															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24	p0															
									23	22	21	20	19	18	17	16

Figure 2-7 BBBP, DMA FIFO output bits [15:0]

bpp	DMA FIFO output bits															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	p24	p25	p26	p27	p28	p29	p30	p31	p16	p17	p18	p19	p20	p21	p22	p23
2	p12		p13		p14		p15		p8		p9		p10		p11	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p6				p7				p4				p5			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p3								p2							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p1															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24	p0															
									23	22	21	20	19	18	17	16

Figure 2-8 LBBP, DMA FIFO output bits [31:16]

bpp	DMA FIFO output bits															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	p8	p9	p10	p11	p12	p13	p14	p15	p0	p1	p2	p3	p4	p5	p6	p7
2	p8		p9		p10		p11		p12		p13		p14		p15	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p2				p3				p0				p1			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p1								p0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24	p0															
									23	22	21	20	19	18	17	16

Figure 2-9 LBBP, DMA FIFO output bits [15:0]

Table 2-1 RGB mode data format

DMA FIFO output bits	24-bit RGB	16-bit 1:5:5:5 RGB	16-bit 5:6:5 RGB	16-bit 4:4:4 RGB
[31]	0	p1 Intensity bit	p1 Blue4	0
[30]	0	p1 Blue4	p1 Blue3	0
[29]	0	p1 Blue3	p1 Blue2	0
[28]	0	p1 Blue2	p1 Blue1	0
[27]	0	p1 Blue1	p1 Blue0	p1 Blue3
[26]	0	p1 Blue0	p1 Green5	p1 Blue2
[25]	0	p1 Green4	p1 Green4	p1 Blue1
[24]	0	p1 Green3	p1 Green3	p1 Blue0
[23]	p0 Blue7	p1 Green2	p1 Green2	p1 Green3
[22]	p0 Blue6	p1 Green1	p1 Green1	p1 Green2
[21]	p0 Blue5	p1 Green0	p1 Green0	p1 Green1
[20]	p0 Blue4	p1 Red4	p1 Red4	p1 Green0
[19]	p0 Blue3	p1 Red3	p1 Red3	p1 Red3

Table 2-1 RGB mode data format (continued)

DMA FIFO output bits	24-bit RGB	16-bit 1:5:5:5 RGB	16-bit 5:6:5 RGB	16-bit 4:4:4 RGB
[18]	p0 Blue2	p1 Red2	p1 Red2	p1 Red2
[17]	p0 Blue1	p1 Red1	p1 Red1	p1 Red1
[16]	p0 Blue0	p1 Red0	p1 Red0	p1 Red0
[15]	p0 Green7	p1 Intensity bit	p0 Blue4	0
[14]	p0 Green6	p0 Blue4	p0 Blue3	0
[13]	p0 Green5	p0 Blue3	p0 Blue2	0
[12]	p0 Green4	p0 Blue2	p0 Blue1	0
[11]	p0 Green3	p0 Blue1	p0 Blue0	p0 Blue3
[10]	p0 Green2	p0 Blue0	p0 Green5	p0 Blue2
[9]	p0 Green1	p0 Green4	p0 Green4	p0 Blue1
[8]	p0 Green0	p0 Green3	p0 Green3	p0 Blue0
[7]	p0 Red7	p0 Green2	p0 Green2	p0 Green3
[6]	p0 Red6	p0 Green1	p0 Green1	p0 Green2
[5]	p0 Red5	p0 Green0	p0 Green0	p0 Green1
[4]	p0 Red4	p0 Red4	p0 Red4	p0 Green0
[3]	p0 Red3	p0 Red3	p0 Red3	p0 Red3
[2]	p0 Red2	p0 Red2	p0 Red2	p0 Red2
[1]	p0 Red1	p0 Red1	p0 Red1	p0 Red1
[0]	p0 Red0	p0 Red0	p0 Red0	p0 Red0

2.5 RAM palette

The RAM-based palette is a 256x16 bit dual-port RAM, physically structured as 128x32-bit. You can write two entries into the palette from a single word write access. The LSB of the serialized pixel data selects between upper and lower halves of the palette RAM. The half that is selected depends on the byte ordering mode. In little-endian mode, setting the LSB selects the upper half, but in big-endian mode, it selects the lower half of the palette.

Note

Windows CE byte ordering is little-endian, so the former case applies.

You can write and verify pixel data values through the AHB slave interface. For information about the numbers of colors supported, see *Number of colors supported* on page 1-5.

The palette RAM is a dual-port RAM with independent controls and addresses for each interface. Port1 is a read/write interface and connects to the AHB slave interface. You can write and verify the palette entries through this interface. Port2 is a read-only interface and connects to the unpacker and gray scaler. For color modes of less than 16bpp, the palette enables each pixel value to be mapped to a 16-bit color:

- for TFT displays, the 16-bit value is passed directly to the pixel serializer
- for STN displays, the 16-bit value is first converted by the gray scaler.

Table 2-2 lists the bit representation of the palette data. The palette 16-bit output uses the TFT 1:5:5:5 data format.

Table 2-2 Palette data storage mode

Palette data	RGB format		BGR format	
Bit	**Name**	**Description**	**Name**	**Description**
[31]	I	Intensity/unused	I	Intensity/unused
[30:26]	B[4:0]	Blue palette data	R[4:0]	Red palette data
[25:21]	G[4:0]	Green palette data	G[4:0]	Green palette data
[20:16]	R[4:0]	Red palette data	B[4:0]	Blue palette data
[15]	I	Intensity/unused	I	Intensity/unused

Table 2-2 Palette data storage mode (continued)

Palette data	RGB format		BGR format	
Bit	**Name**	**Description**	**Name**	**Description**
[14:10]	B[4:0]	Blue palette data	R[4:0]	Red palette data
[9:5]	G[4:0]	Green palette data	G[4:0]	Green palette data
[4:0]	R[4:0]	Red palette data	B[4:0]	Blue palette data

For mono STN mode, only the red palette field bits [4:1] are used. However, in STN color mode, the green and blue [4:1] are also used. Only 4 bits per color are used, because the gray scaler only supports 16 different shades per color.

You can swap the red and blue pixel data to support BGR data format using a control register bit (bit 8 = BGR). See *LCD Control Register* on page 3-12 for more information. Table 2-3 lists the bit representation of the palette data for the STN mode.

Table 2-3 Palette data storage for STN modes

Palette data	Mono format		RGB format		BGR format	
Bit	**Name**	**Description**	**Name**	**Description**	**Name**	**Description**
[31]	-	Unused	-	Unused	-	Unused
[30:27]	-	Unused	B[3:0]	Blue palette data	R[3:0]	Red palette data
[26]	-	Unused	-	Unused	-	Unused
[25:22]	-	Unused	G[3:0]	Green palette data	G[3:0]	Green palette data
[21]	-	Unused	-	Unused	-	Unused
[20:17]	Y[3:0]	Intensity data	R[3:0]	Red palette data	B[3:0]	Blue palette data
[16]	-	Unused	-	Unused	-	Unused
[15]	-	Unused	I	Unused	I	Unused
[14:11]	-	Unused	B[4:1]	Blue palette data	R[4:1]	Red palette data
[10]	-	Unused	B[0]	Unused	R[0]	Unused
[9:6]	-	Unused	G[4:1]	Green palette data	G[4:1]	Green palette data

Table 2-3 Palette data storage for STN modes (continued)

Palette data	Mono format		RGB format		BGR format	
Bit	**Name**	**Description**	**Name**	**Description**	**Name**	**Description**
[5]	-	Unused	G[0]	Unused	G[0]	Unused
[4:1]	Y[3:0]	Intensity data	R[4:1]	Red palette data	B[4:1]	Blue palette data
[0]	-	Unused	R[0]	Unused	B[0]	Unused

Note

In 16 and 24bpp TFT mode, the palette is bypassed, and the output of the pixel serializer is used as the TFT panel data.

2.6 Hardware cursor

The following sections describe the hardware cursor:

- *Integration with the controller*
- *Operation*
- *Supported cursor sizes* on page 2-17
- *Movement* on page 2-17
- *Image format* on page 2-20.

2.6.1 Integration with the controller

The hardware cursor is an integral part of the controller. It uses the LCD timing module to provide an indication of the current scan position coordinate, and intercepts the pixel stream between the palette logic and the gray scale/output multiplexer.

The AHB slave interface enables access to all the cursor programming registers and provides a read/write interface to the cursor image RAM.

Figure 2-10 shows a block diagram of the hardware cursor.

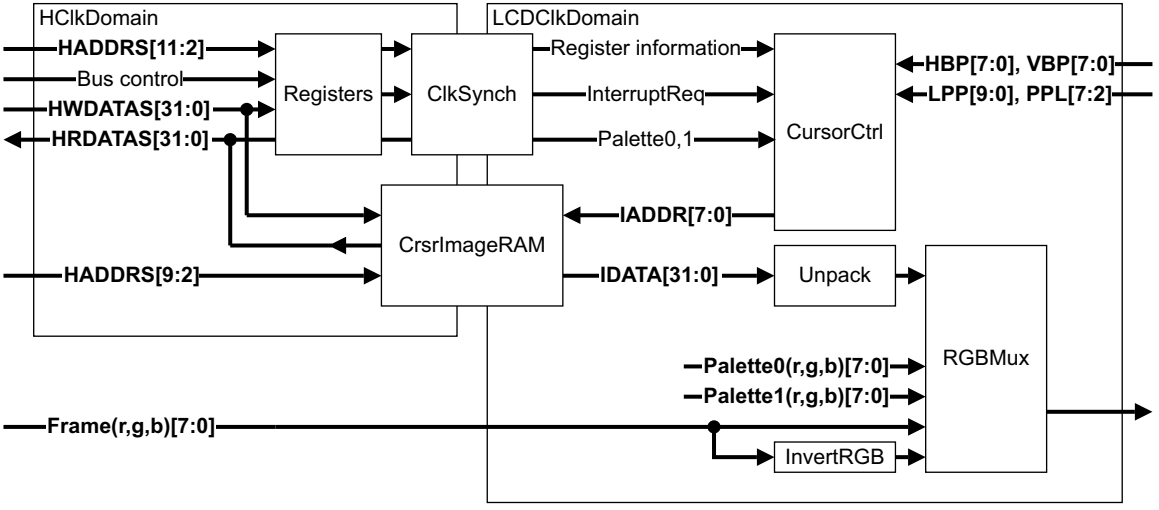


Figure 2-10 Hardware cursor block diagram

2.6.2 Operation

A dual-port RAM contains the hardware cursor. Software programs it through the AHB slave interface. The AHB slave interface also provides access to the hardware cursor control registers. These registers enable you to modify the cursor position and perform

various other functions. When enabled, the hardware cursor uses the horizontal and vertical synchronization signals, along with a pixel clock enable, and various display parameters, to calculate the current scan coordinate such as:

- *Horizontal Back Porch* (HBP)
- *Vertical Back Porch* (VBP)
- *Horizontal ACtive line* (HAC)
- *Vertical Active Column* (VAC)
- *Pixels-Per-Line* (PPL)
- *Lines-Per-Panel* (LPP).

When the display point is inside the bounds of the cursor image, the cursor replaces frame buffer pixels with cursor pixels.

The image RAM offers single-cycle performance to enable the read to take place in the clock cycle preceding its use, and maintains the data output until the next access. This removes the requirement for a holding register.

When the last cursor pixel is displayed, an interrupt is generated that software can use as an indication that it is safe to modify the cursor image. This enables you to perform software-controlled animations without flickering for frame-synchronized cursors.

2.6.3 Supported cursor sizes

Table 2-4 lists the two supported cursor sizes.

Table 2-4 Supported cursor images

X Pixels	Y Pixels	Bits per pixel	Words per line	Words in cursor image
32	32	2	2	64
64	64	2	4	256

2.6.4 Movement

The following descriptions assume that both the screen and cursor origins are at the top left of the visible screen (the first visible pixel scanned each frame). Figure 2-11 on page 2-18 shows how each pixel coordinate is assumed to be the top left corner of the pixel.

The following sections describe cursor positioning:

- *Cursor XY positioning* on page 2-18
- *Cursor clipping* on page 2-19.

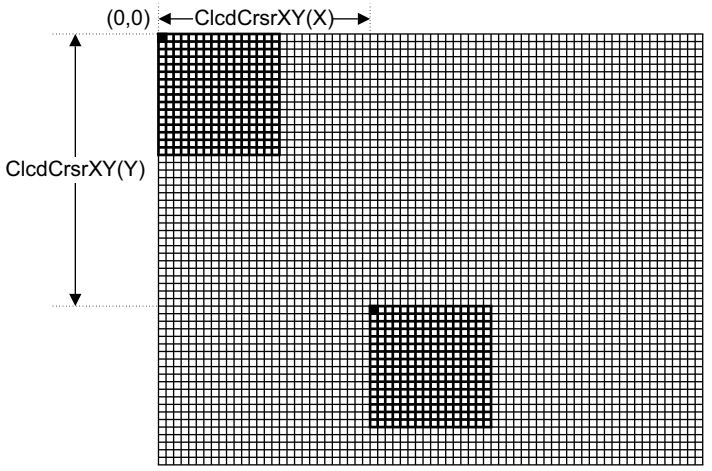


Figure 2-11 Hardware cursor movement

Cursor XY positioning

The ClcdCrsrXY Register controls the cursor position on the cursor overlay. See *Cursor XY Position Register* on page 3-23. This provides separate fields for X and Y ordinates.

The ClcdCrsrConfig Register provides a FrameSync bit controlling the visible behavior of the cursor. See *Cursor Configuration Register* on page 3-21.

With FrameSync inactive, the cursor responds immediately to any change in the programmed ClcdCrsrXY value.

Note

You might see some transient smearing effects if you move the cursor across the LCD scan line.

With FrameSync active, the cursor only updates its position after a vertical synchronization has occurred. This provides clean cursor movement, but the cursor position only updates once a frame.

Cursor clipping

The ClcdCrsrXY Register (see *Cursor XY Position Register* on page 3-23) is programmed with positive binary values that enable you to locate the cursor image anywhere on the visible screen image. The cursor image is clipped automatically at the screen limits when it extends beyond the screen image to the right or bottom. See X1,Y1 in Figure 2-12. The checked pattern shows the visible portion of the cursor.

Because the ClcdCrsrXY Register values are positive integers, to emulate cursor clipping on the left and top of screen, a Clip Index Register, ClcdCrsrClipXY, is provided. This controls the point of the cursor image that is positioned at the ClcdCrsrXY coordinate. For clipping functions on the Y axis, ClcdCrsrXY(X) is zero, and Clip(X) is programmed to provide the offset into the cursor image (X2 and X3). The equivalent function is provided to clip on the X axis at the top of the display (Y2).

For cursors that are not clipped at the X=0 or Y=0 lines, program the Clip Index Register X and Y fields with zero to display the cursor correctly. See Clip(X4,Y4) in Figure 2-12 for the effect of incorrect programming.

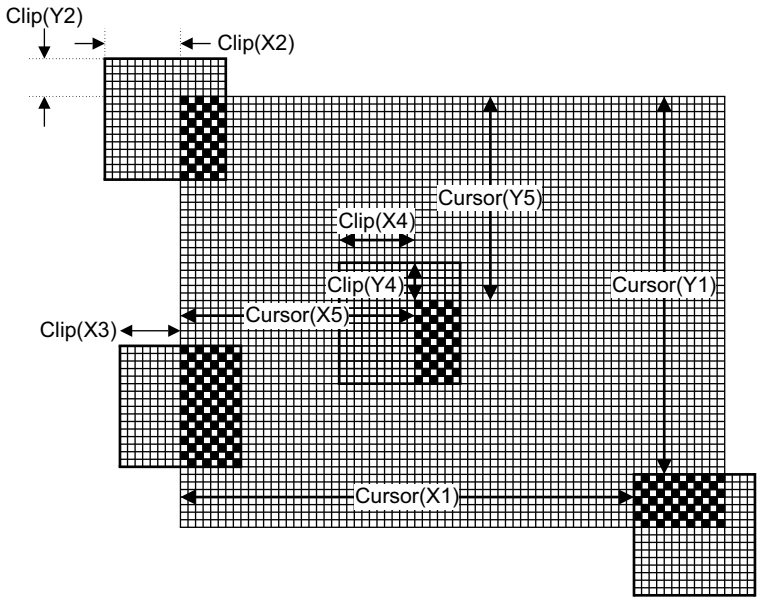


Figure 2-12 Hardware cursor clipping

2.6.5 Image format

The LCD frame buffer supports three packing formats, but the hardware cursor image requirement has been simplified to support only LBBP. This is little-endian byte, big-endian pixel for Windows CE mode.

The Image RAM start address (IBase) is offset by `0x800` from the LCD base address. The displayed cursor coordinate system is expressed in terms of (X,Y). 64x64 is an extension of the 32x32 format that Figure 2-13 shows.

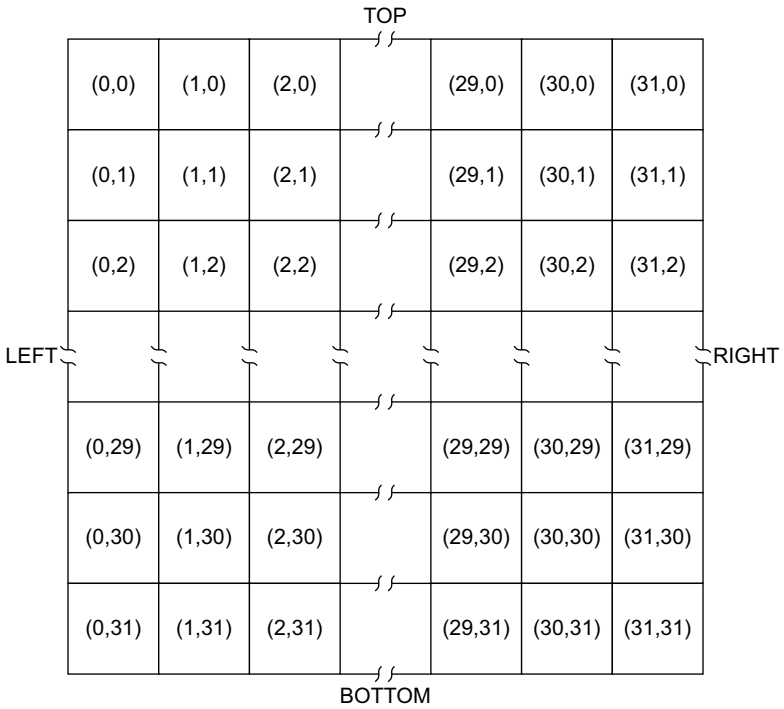


Figure 2-13 Hardware cursor image format

The following sections describe the hardware cursor image formats:

- *32x32 pixels* on page 2-21
- *64x64 pixels* on page 2-22
- *Software format for Windows CE* on page 2-23
- *Pixel encoding* on page 2-25.

32x32 pixels

Four cursors are held in memory, each with the same pixel format. Table 2-5 lists the base addresses for the four cursors.

Table 2-5 32x32 cursor base addresses

Address	Description
IBase	Cursor0 start address
IBase+0x100	Cursor1 start address
IBase+0x200	Cursor2 start address
IBase+0x300	Cursor3 start address

Table 2-6 and Table 2-7 on page 2-22 list the LBBP buffer to pixel mapping for Cursor0.

Table 2-6 LBBP buffer to pixel mapping 32x32 Cursor0 for datawords [31:16]

Address from CLCDC base	Dataword							
IBase=0x800	[31:30]	[29:28]	[27:26]	[25:24]	[23:22]	[21:20]	[19:18]	[17:16]
IBase	(12,0)	(13,0)	(14,0)	(15,0)	(8,0)	(9,0)	(10,0)	(11,0)
IBase+4	(28,0)	(29,0)	(30,0)	(31,0)	(24,0)	(25,0)	(26,0)	(27,0)
IBase+(8*y)	(12,y)	(13,y)	(14,y)	(15,y)	(8,y)	(9,y)	(10,y)	(11,y)
IBase+(8*y)+4	(28,y)	(29,y)	(30,y)	(31,y)	(24,y)	(25,y)	(26,y)	(27,y)
IBase+0xF8	(12,31)	(13,31)	(14,31)	(15,31)	(8,31)	(9,31)	(10,31)	(11,31)
IBase+0xFC	(28,31)	(29,31)	(30,31)	(31,31)	(24,31)	(25,31)	(26,31)	(27,31)

Table 2-7 LBBP buffer to pixel mapping 32x32 Cursor0 for datawords [15:0]

Address from CLCDC base	Dataword							
IBase=0x800	**[15:14]**	**[13:12]**	**[11:10]**	**[9:8]**	**[7:6]**	**[5:4]**	**[3:2]**	**[1:0]**
IBase	(4,0)	(5,0)	(6,0)	(7,0)	(0,0)	(1,0)	(2,0)	(3,0)
IBase+4	(20,0)	(21,0)	(22,0)	(23,0)	(16,0)	(17,0)	(18,0)	(19,0)
IBase+(8*y)	(4,y)	(5,y)	(6,y)	(7,y)	(0,y)	(1,y)	(2,y)	(3,y)
IBase+(8*y)+4	(20,y)	(21,y)	(22,y)	(23,y)	(16,y)	(17,y)	(18,y)	(19,y)
IBase+0xF8	(4,31)	(5,31)	(6,31)	(7,31)	(0,31)	(1,31)	(2,31)	(3,31)
IBase+0xFC	(20,31)	(21,31)	(22,31)	(23,31)	(16,31)	(17,31)	(18,31)	(19,31)

64x64 pixels

Table 2-8 and Table 2-9 on page 2-23 list the format that a single 64-bit cursor can be held at.

Table 2-8 LBBP buffer to pixel mapping 64x64 for datawords [31:16]

Address from CLCDC base	Dataword							
IBase=0x800	**[31:30]**	**[29:28]**	**[27:26]**	**[25:24]**	**[23:22]**	**[21:20]**	**[19:18]**	**[17:16]**
IBase	(12,0)	(13,0)	(14,0)	(15,0)	(8,0)	(9,0)	(10,0)	(11,0)
IBase+0x4	(28,0)	(29,0)	(30,0)	(31,0)	(24,0)	(25,0)	(26,0)	(27,0)
IBase+0x8	(44,0)	(45,0)	(46,0)	(47,0)	(40,0)	(41,0)	(42,0)	(43,0)
IBase+0xC	(60,0)	(61,0)	(62,0)	(63,0)	(56,0)	(57,0)	(58,0)	(59,0)
IBase+(16*y)	(12,y)	(13,y)	(14,y)	(15,y)	(8,y)	(9,y)	(10,y)	(11,y)
IBase+(16*y)+4	(28,y)	(29,y)	(30,y)	(31,y)	(24,y)	(25,y)	(26,y)	(27,y)
IBase+(16*y)+8	(44,y)	(45,y)	(46,y)	(47,y)	(40,y)	(41,y)	(42,y)	(43,y)
IBase+(16*y)+12	(60,y)	(61,y)	(62,y)	(63,y)	(56,y)	(57,y)	(58,y)	(59,y)
IBase+0x3FC	(60,63)	(61,63)	(62,63)	(63,63)	(56,63)	(57,63)	(58,63)	(59,63)

Table 2-9 LBBP buffer to pixel mapping 64x64 for datawords [15:0]

Address from CLCDC base	Dataword							
IBase=0x800	**[15:14]**	**[13:12]**	**[11:10]**	**[9:8]**	**[7:6]**	**[5:4]**	**[3:2]**	**[1:0]**
IBase	(4,0)	(5,0)	(6,0)	(7,0)	(0,0)	(1,0)	(2,0)	(3,0)
IBase+0x4	(20,0)	(21,0)	(22,0)	(23,0)	(16,0)	(17,0)	(18,0)	(19,0)
IBase+0x8	(36,0)	(37,0)	(38,0)	(39,0)	(32,0)	(33,0)	(34,0)	(35,0)
IBase+0xC	(52,0)	(53,0)	(54,0)	(55,0)	(48,0)	(49,0)	(50,0)	(51,0)
IBase+(16*y)	(4,y)	(5,y)	(6,y)	(7,y)	(0,y)	(1,y)	(2,y)	(3,y)
IBase+(16*y)+4	(20,y)	(21,y)	(22,y)	(23,y)	(16,y)	(17,y)	(18,y)	(19,y)
IBase+(16*y)+8	(36,y)	(37,y)	(38,y)	(39,y)	(32,y)	(33,y)	(34,y)	(35,y)
IBase+(16*y)+12	(52,y)	(53,y)	(54,y)	(55,y)	(48,y)	(49,y)	(50,y)	(51,y)
IBase+0x3FC	(52,63)	(53,63)	(54,63)	(55,63)	(48,63)	(49,63)	(50,63)	(51,63)

Software format for Windows CE

The software low-level cursor mask is split into two memory areas, providing separate AND and XOR buffers. These buffers are held as single bit memory images, and software must combine them to form a unified cursor image that is then copied to the cursor image buffer when required.

The hardware cursor does not support full color cursor images because of practical limitations on the size of the image dual-port RAM.

The software AND and XOR masks both follow the same format. Table 2-10 on page 2-24 lists the format for 32x32 pixels, and Table 2-11 on page 2-24 lists the format for 64x64 pixels.

Table 2-10 32x32 software mask storage

External memory offset	Data bit to pixel correspondence							
	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
+0	(0,0)	(1,0)	(2,0)	(3,0)	(4,0)	(5,0)	(6,0)	(7,0)
+1	(8,0)	(9,0)	(10,0)	(11,0)	(12,0)	(13,0)	(14,0)	(15,0)
+2	(16,0)	(17,0)	(18,0)	(19,0)	(20,0)	(21,0)	(22,0)	(23,0)
+3	(24,0)	(25,0)	(26,0)	(27,0)	(28,0)	(29,0)	(30,0)	(31,0)
…	-	-	-	-	-	-	-	-
+127	(24,31)	(25,31)	(26,31)	(27,31)	(28,31)	(29,31)	(30,31)	(31,31)

Table 2-11 64x64 software mask storage

External memory offset	Data bit to pixel correspondence							
	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
+0	(0,0)	(1,0)	(2,0)	(3,0)	(4,0)	(5,0)	(6,0)	(7,0)
+1	(8,0)	(9,0)	(10,0)	(11,0)	(12,0)	(13,0)	(14,0)	(15,0)
+2	(16,0)	(17,0)	(18,0)	(19,0)	(20,0)	(21,0)	(22,0)	(23,0)
+3	(24,0)	(25,0)	(26,0)	(27,0)	(28,0)	(29,0)	(30,0)	(31,0)
+4	(32,0)	(33,0)	(34,0)	(35,0)	(36,0)	(37,0)	(38,0)	(39,0)
+5	(40,0)	(41,0)	(42,0)	(43,0)	(44,0)	(45,0)	(46,0)	(47,0)
+6	(48,0)	(49,0)	(50,0)	(51,0)	(52,0)	(53,0)	(54,0)	(55,0)
+7	(56,0)	(57,0)	(58,0)	(59,0)	(60,0)	(61,0)	(62,0)	(63,0)
…	-	-	-	-	-	-	-	-
+511	(56,63)	(57,63)	(58,63)	(59,63)	(60,63)	(61,63)	(62,63)	(63,63)

Pixel encoding

Each pixel of the cursor requires two bits of information. These are interpreted as Color0, Color1, Transparent, and Transparent inverted.

In the coding scheme, bit 1 selects between color and transparent (AND mask) and bit 0 selects variant (XOR mask).

Table 2-12 lists the pixel encoding bit assignments.

Table 2-12 Pixel encoding

Bits	Description
[00]	Color0. The cursor color is displayed according to the *Red-Green-Blue* (RGB) value programmed into the CrsrPalette0 Register.
[01]	Color1. The cursor color is displayed according to the RGB value programmed into the CrsrPalette1 Register.
[10]	Transparent. The cursor pixel is transparent, so is displayed unchanged. This enables the visible cursor to assume shapes that are not square.
[11]	Transparent inverted. The cursor pixel assumes the complementary color of the frame pixel that is displayed. You can use it to ensure that the cursor is visible regardless of the color of the frame buffer image.

2.7 Gray scaler

A patented gray scale algorithm drives mono and color STN panels. This provides 15 gray scales for mono displays. For STN color displays, the three color components (RGB) are gray scaled simultaneously. This results in 3375 (15x15x15) colors being available. The gray scaler transforms each 4-bit gray value into a sequence of activity-per-pixel over several frames, relying to some degree on the display characteristics, to give the representation of gray scales and color.

2.8 Upper and lower panel formatters

Formatters are used in STN mode to convert the gray scaler output to a parallel format that the display requires. For mono displays, this is either 4- or 8-bits wide, and for color displays, it is 8-bits wide. Table 2-13 shows a color display driven with $2^2/_3$ pixels worth of data in a repeating sequence.

Table 2-13 Color display driven with $2^2/_3$ pixel data in repeating sequence

Byte	CLD[7]	CLD[6]	CLD[5]	CLD[4]	CLD[3]	CLD[2]	CLD[1]	CLD[0]
0	P2[Green]	P2[Red]	P1[Blue]	P1[Green]	P1[Red]	P0[Blue]	P0[Green]	P0[Red]
1	P5[Red]	P4[Blue]	P4[Green]	P4[Red]	P3[Blue]	P3[Green]	P3[Red]	P2[Blue]
2	P7[Blue]	P7[Green]	P7[Red]	P6[Blue]	P6[Green]	P6[Red]	P5[Blue]	P5[Green]

Each formatter consists of three 3-bit (RGB) shift left registers. RGB pixel data bit values from the gray scaler are concurrently shifted into the respective registers. When enough data is available, a byte is constructed by multiplexing the registered data to the correct bit position to satisfy the RGB data pattern of LCD panel. The byte is transferred to the 3-byte FIFO. This has enough space to store eight color pixels.

2.9 Panel clock generator

The output of the panel clock generator block is the panel clock. This is a divided down version of **CLCDCLK**. You can program it in the range **CLCDCLK**/2 to **CLCDCLK**/1025 to match the bpp data rate of the LCD panel.

2.10 Timing controller

The primary function of the timing controller block is to generate the horizontal and vertical timing panel signals. It also provides panel bias and enable signals. You can configure these timings by using the AHB slave interface to program the relevant registers.

2.11 STN and TFT data select

Support is provided for passive STN and active TFT panel types:

STN displays STN display panels require algorithmic pixel pattern generation to provide pseudo gray scaling on mono, or color creation on color displays.

TFT displays TFT display panels require you to apply the digital color value of each pixel to the display data inputs.

2.12 Interrupt generation

The controller provides four interrupts, **CLCDFUFINTR**, **CLCDLNBUINTR**, **CLCDMBEINTR**, **CLCDVCOMPINTR**, and a single combined interrupt, **CLCDINTR**, that you can mask individually. The single combined interrupt is asserted if any of the combined interrupts are asserted and unmasked. For more information, see *Interrupts* on page 3-33.

Chapter 3
Programmer's Model

This chapter describes the CLCDC registers and provides information for programming the controller. It contains the following sections:

3.1 About the programmer's model

The following applies to the registers that the controller uses:

- The base address of the controller is not fixed and can be different for any particular system implementation. However, the offset of any particular register from the base address is fixed.
- You must not access reserved or unused address locations because this can result in unpredictable behavior of the controller.
- You must write reserved or unused bits of registers as zero, and ignore them on read unless otherwise stated in the relevant text.
- A system or power-on reset resets all register bits to a logic 0 unless otherwise stated in the relevant text.
- All registers support read/write accesses unless otherwise stated in the relevant text. A write updates the contents of a register and a read returns the contents of the register.

3.2 Register summary

All register addresses in the controller are fixed relative to its base address. Table 3-1 lists the registers in base offset order.

Table 3-1 Register summary

Name	Base offset	Type	Reset value	Description
LCDTiming0	0x000	R/W	0x00000000	*Horizontal Axis Panel Control Register* on page 3-5
LCDTiming1	0x004	R/W	0x00000000	*Vertical Axis Panel Control Register* on page 3-6
LCDTiming2	0x008	R/W	0x00000000	*Clock and Signal Polarity Control Register* on page 3-8
LCDTiming3	0x00C	R/W	0x00000	*Line End Control Register* on page 3-10
LCDUPBASE	0x010	R/W	0x00000000	*Upper and Lower Panel Frame Base Address Registers* on page 3-10
LCDLPBASE	0x014	R/W	0x00000000	*Upper and Lower Panel Frame Base Address Registers* on page 3-10
LCDControl	0x018	R/W	0x0000	*LCD Control Register* on page 3-12
LCDIMSC	0x01C	R/W	0x0	*Interrupt Mask Set/Clear Register* on page 3-14
LCDRIS	0x020	RO	0x0	*Raw Interrupt Status Register* on page 3-15
LCDMIS	0x024	RO	0x0	*Masked Interrupt Status Register* on page 3-16
LCDICR	0x028	WO	0x0	*LCD Interrupt Clear Register* on page 3-17
LCDUPCURR	0x02C	RO	0x00000000	*LCD Upper and Lower Panel Current Address Value Registers* on page 3-18
LCDLPCURR	0x030	RO	0x00000000	*LCD Upper and Lower Panel Current Address Value Registers* on page 3-18
-	0x034-0x1FC	-	-	Reserved
LCDPalette	0x200-0x3FC	R/W	0x00000000	*256x16-bit Color Palette Registers* on page 3-19
-	0x400-0x7FC	-	-	Reserved
CursorImage	0x800-0xBFC	R/W	0x00000000	*Cursor Image RAM Register* on page 3-20
ClcdCrsrCtrl	0xC00	R/W	0x00	*Cursor Control Register* on page 3-20
ClcdCrsrConfig	0xC04	R/W	0x0	*Cursor Configuration Register* on page 3-21

Table 3-1 Register summary (continued)

Name	Base offset	Type	Reset value	Description
ClcdCrsrPalette0	0xC08	R/W	0x000000	*Cursor Palette Registers* on page 3-22
ClcdCrsrPalette1	0xC0C	R/W	0x000000	*Cursor Palette Registers* on page 3-22
ClcdCrsrXY	0xC10	R/W	0x00000000	*Cursor XY Position Register* on page 3-23
ClcdCrsrClip	0xC14	R/W	0x0000	*Cursor Clip Position Register* on page 3-24
-	0xC18-0xC1C	-	-	Reserved
ClcdCrsrIMSC	0xC20	R/W	0x0	*Cursor Interrupt Mask Set/Clear Register* on page 3-25
ClcdCrsrICR	0xC24	WO	0x0	*Cursor Interrupt Clear Register* on page 3-25
ClcdCrsrRIS	0xC28	RO	0x0	*Cursor Raw Interrupt Status Register* on page 3-26
ClcdCrsrMIS	0xC2C	RO	0x0	*Cursor Masked Interrupt Status Register* on page 3-26
-	0xC30-0xDFC	-	-	Reserved
-	0xF00-0xF08	-	-	Chapter 4 *Programmer's Model for Test*
-	0xF0C-0xFDC	-	-	Reserved
CLCDPeriphID0	0xFE0	RO	0x11	*Peripheral Identification Register 0* on page 3-28
CLCDPeriphID1	0xFE4	RO	0x11	*Peripheral Identification Register 1* on page 3-28
CLCDPeriphID2	0xFE8	RO	0x-4[a]	*Peripheral Identification Register 2* on page 3-29
CLCDPeriphID3	0xFEC	RO	0x00	*Peripheral Identification Register 3* on page 3-30
CLCDPCellID0	0xFF0	RO	0x0D	*PrimeCell Identification Register 0* on page 3-31
CLCDPCellID1	0xFF4	RO	0xF0	*PrimeCell Identification Register 1* on page 3-31
CLCDPCellID2	0xFF8	RO	0x05	*PrimeCell Identification Register 2* on page 3-32
CLCDPCellID3	0xFFC	RO	0xB1	*PrimeCell Identification Register 3* on page 3-32

a. This value depends on the revision status of the controller.

3.3 Register descriptions

This section describes the registers with the exception of the test registers that Chapter 4 *Programmer's Model for Test* describes. Table 3-1 on page 3-3 provides cross-references to individual registers.

3.3.1 Horizontal Axis Panel Control Register

The LCDTiming0 Register controls:

- *Horizontal Synchronization pulse Width* (HSW)
- *Horizontal Front Porch* (HFP) period
- *Horizontal Back Porch* (HBP) period
- *Pixels-Per-Line* (PPL).

Figure 3-1 shows the bit assignments.

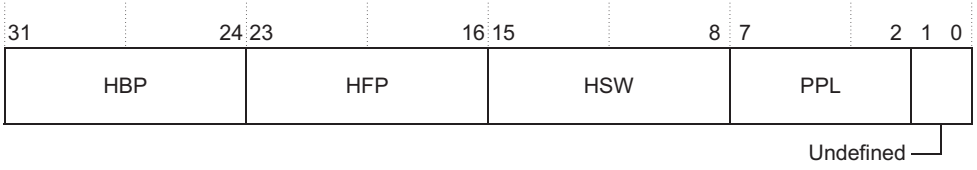


Figure 3-1 LCDTiming0 Register bit assignments

Table 3-2 lists the bit assignments.

Table 3-2 LCDTiming0 Register bit assignments

Bit	Name	Description
[31:24]	HBP	Horizontal back porch. This is the number of **CLCP** periods between the falling edge of **CLLP** and the start of active data. Program with value minus 1. Use the 8-bit HBP field to specify the number of pixel clock periods inserted at the beginning of each line or row of pixels. After the line clock for the previous line has been deasserted, the value in HBP counts the number of pixel clocks to wait before starting the next display line. HBP can generate a delay of 1-256 pixel clock cycles.
[23:16]	HFP	Horizontal front porch. This is the number of **CLCP** periods between the end of active data and the rising edge of **CLLP**. Program with value minus 1. The 8-bit HFP field sets the number of pixel clock intervals at the end of each line or row of pixels, before the LCD line clock is pulsed. When a complete line of pixels is transmitted to the LCD driver, the value in HFP counts the number of pixel clocks to wait before asserting the line clock. HFP can generate a period of 1-256 pixel clock cycles.

Table 3-2 LCDTiming0 Register bit assignments (continued)

Bit	Name	Description
[15:8]	HSW	Horizontal synchronization pulse width. This is the width of the **CLLP** signal in **CLCP** periods. Program with value minus 1. The 8-bit HSW field specifies the pulse width of the line clock in passive mode, or the horizontal synchronization pulse in active mode.
[7:2]	PPL	Pixels-per-line. Actual pixels-per-line = 16 * (PPL + 1). This field specifies the number of pixels in each line of the screen. PPL is a 6-bit value that represents between 16 and 1024 pixels-per-line. PPL counts the number of pixel clocks that occur before the HFP is applied. Program the value required divided by 16, minus 1. For example, if you require an actual pixels-per-line value of 320 then calculate and program PPL as (320/16) - 1 = 19 or b010011.
[1:0]	-	Undefined.

Horizontal timing restrictions

DMA requests new data at the start of a horizontal display line. You must allocate some time for the DMA transfer and for the data to propagate down the FIFO path in the LCD interface. The data path latency forces some restrictions on the usable minimum values for horizontal porch width in STN mode. The minimum values are HSW = 2 and HBP = 2.

Single-panel mode:

- HSW = 3
- HBP = 5
- HFP = 5
- *Panel Clock Divisor* (PCD) = 1 (**CLCDCLK**/3).

Dual-panel mode:

- HSW = 3
- HBP = 5
- HFP = 5
- PCD = 5 (**CLCDCLK**/7).

If enough time is given at the start of the line, for example, setting HSW = 6, HBP = 10, data does not corrupt for PCD = 4, the minimum value.

3.3.2 Vertical Axis Panel Control Register

The LCDTiming1 Register controls:

- number of *Lines-Per-Panel* (LPP)
- *Vertical Synchronization pulse Width* (VSW)

- *Vertical Front Porch* (VFP) period
- *Vertical Back Porch* (VBP) period.

Figure 3-2 shows the bit assignments.

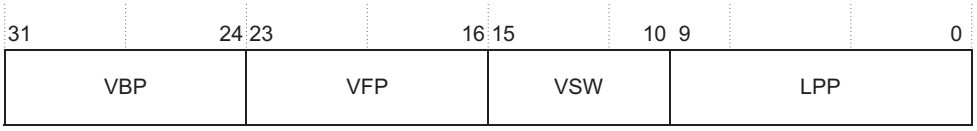


Figure 3-2 LCDTiming1 Register bit assignments

Table 3-3 lists the bit assignments.

Table 3-3 LCDTiming1 Register bit assignments

Bit	Name	Description
[31:24]	VBP	Vertical back porch. This is the number of inactive lines at the start of a frame, after the vertical synchronization period. Program to zero on passive displays or reduced contrast results. The 8-bit VBP field specifies the number of line clocks inserted at the beginning of each frame. The VBP count starts immediately after the vertical synchronization signal for the previous frame has been negated for active mode, or the extra line clocks have been inserted as specified by the VSW bit field in passive mode. After this has occurred, the count value in VBP sets the number of line clock periods inserted before the next frame. VBP generates 0–255 extra line clock cycles.
[23:16]	VFP	Vertical front porch. This is the number of inactive lines at the end of frame, before the vertical synchronization period. Program to zero on passive displays or reduced contrast results. The 8-bit VFP field specifies the number of line clocks to insert at the end of each frame. When a complete frame of pixels is transmitted to the LCD display, the value in VFP counts the number of line clock periods to wait. After the count has elapsed, the vertical synchronization signal, **CLFP**, is asserted in active mode, or extra line clocks are inserted as specified by the VSW bit-field in passive mode. VFP generates 0–255 line clock cycles.
[15:10]	VSW	Vertical synchronization pulse width. This is the number of horizontal synchronization lines. This value must be small (for example, program to zero) for passive STN LCDs. Program to the number of lines required, minus one. The higher the value, the worse the contrast on STN LCDs. The 6-bit VSW field specifies the pulse width of the vertical synchronization pulse. Program the register with the number of line clocks in VSync, minus one. The number of horizontal synchronization lines must be small (for example, program to zero) for passive STN LCDs. Program the register with the number of lines required, minus one. The higher the value, the worse the contrast on STN LCDs.
[9:0]	LPP	Lines per panel. This is the number of active lines per screen. Program to number of lines required, minus 1. The LPP field specifies the total number of lines or rows on the LCD panel being controlled. LPP is a 10-bit value enabling between 1 and 1024 lines. Program the register with the number of lines per LCD panel, minus 1. For dual-panel displays, program the register with the number of lines on each of the upper and lower panels.

3.3.3 Clock and Signal Polarity Control Register

The LCDTiming2 Register controls the CLCDC timing. Figure 3-3 shows the bit assignments.

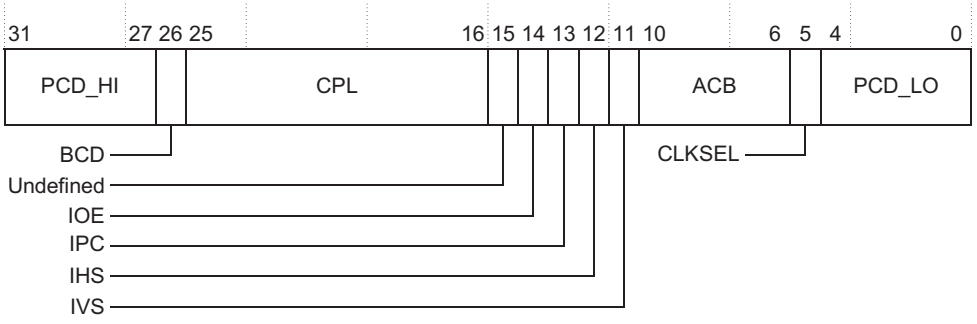


Figure 3-3 LCDTiming2 Register bit assignments

Table 3-4 lists the bit assignments.

Table 3-4 LCDTiming2 Register bit assignments

Bit	Name	Description
[31:27]	PCD_HI	Upper five bits of panel clock divisor.[a] The ten-bit PCD field, comprising PCD_HI and PCD_LO (bits [4:0]), derives the LCD panel clock frequency **CLCP** from the **CLCDCLK** frequency, **CLCP = CLCDCLK**/(PCD+2). For mono STN displays with a 4 or 8-bit interface, the panel clock is a factor of four and eight down on the actual individual pixel clock rate. For color STN displays, $2^2/_3$ pixels are output per **CLCP** cycle, therefore the panel clock is 0.375 times. For TFT displays, you can bypass the pixel clock divider by setting the LCDTiming2[26] BCD bit.
[26]	BCD	Bypass pixel clock divider. Setting this to 1 bypasses the pixel clock divider logic. You use this mainly for TFT displays.
[25:16]	CPL	Clocks per line. This field specifies the number of actual **CLCP** clocks to the LCD panel on each line. For each display type, this is: **TFT** (actual pixels-per-line/1) - 1. **Mono passive** (actual pixels-per-line/4) - 1 or (actual pixels-per-line/8) - 1. **Color passive** (actual pixels-per-line/$2^2/_3$) - 1. You must program this correctly in addition to the PPL bit described in the *Horizontal Axis Panel Control Register* on page 3-5.
[15]	-	Undefined.

Table 3-4 LCDTiming2 Register bit assignments (continued)

Bit	Name	Description
[14]	IOE	Invert output enable: 0 = **CLAC** output pin is active HIGH in TFT mode 1 = **CLAC** output pin is active LOW in TFT mode. This bit selects the active polarity of the output enable signal in TFT mode. In this mode, the **CLAC** pin is an enable that indicates to the LCD panel when valid display data is available. In active display mode, data is driven onto the LCD data lines at the programmed edge of **CLCP** when **CLAC** is in its active state.
[13]	IPC	Invert panel clock: 0 = Data is driven on the LCD data lines on the rising edge of **CLCP** 1 = Data is driven on the LCD data lines on the falling edge of **CLCP**. The IPC bit selects the edge of the panel clock that pixel data is driven on, out onto the LCD data lines.
[12]	IHS	Invert horizontal synchronization: 0 = **CLLP** pin is active HIGH and inactive LOW 1 = **CLLP** pin is active LOW and inactive HIGH. The *Invert HSync* (IHS) bit inverts the polarity of the **CLLP** signal.
[11]	IVS	Invert vertical synchronization: 0 = **CLFP** pin is active HIGH and inactive LOW 1 = **CLFP** pin is active LOW and inactive HIGH. The *Invert VSync* (IVS) bit inverts the polarity of the **CLFP** signal.
[10:6]	ACB	AC bias pin frequency. The AC bias pin frequency is only applicable to STN displays. These require the pixel voltage polarity to periodically reverse to prevent damage caused by DC charge accumulation. Program this field with the required value, minus one, to apply the number of line clocks between each toggle of the AC bias pin, **CLAC**. This field has no effect if the controller is operating in TFT mode, when the **CLAC** pin is a data enable signal.
[5]	CLKSEL	This bit drives the **CLCDCLKSEL** signal. It is the select signal for the external LCD clock multiplexor.
[4:0]	PCD_LO	Lower five bits of panel clock divisor.[a] The ten-bit PCD field derives the LCD panel clock frequency **CLCP** from the **CLCDCLK** frequency, **CLCP** = **CLCDCLK**/(PCD+2). For mono STN displays with a 4 or 8-bit interface, the panel clock is a factor of four and eight down on the actual individual pixel clock rate. For color STN displays, $2^2/_3$ pixels are output per **CLCP** cycle, therefore the panel clock is 0.375 times. For TFT displays, you can bypass the pixel clock divider by setting the LCDTiming2[26] BCD bit.

a. The data path latency forces some restrictions on the usable minimum values for the panel clock divider in STN modes:
single-panel color mode, PCD = 1 (**CLCP** = **CLCDCLK**/3)
dual-panel color mode, PCD = 4 (**CLCP** = **CLCDCLK**/6)
single-panel mono 4-bit interface mode, PCD = 2 (**CLCP** = **CLCDCLK**/4)
dual-panel mono 4-bit interface mode, PCD = 6 (**CLCP** = **CLCDCLK**/8)
single-panel mono 8-bit interface mode, PCD = 6 (**CLCP** = **CLCDCLK**/8)
dual-panel mono 8-bit interface mode, PCD = 14 (**CLCP** = **CLCDCLK**/16).

3.3.4 Line End Control Register

The LCDTiming3 Register controls the enabling of line-end signal **CLLE**. When enabled, a positive pulse, four **CLCDCLK** periods wide, is output on **CLLE** after a programmable delay, LED, from the last pixel of each display line. If the line-end signal is disabled, it is held permanently LOW. Figure 3-4 shows the bit assignments.

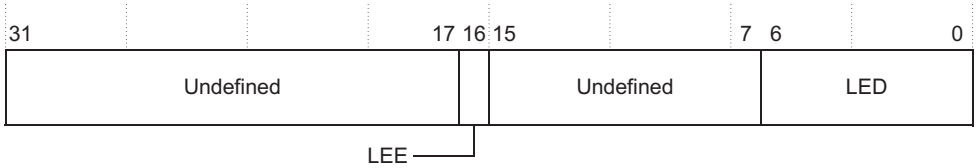


Figure 3-4 LCDTiming3 Register bit assignments

Table 3-5 lists the bit assignments.

Table 3-5 LCDTiming3 Register bit assignments

Bit	Name	Description
[31:17]	-	Undefined.
[16]	LEE	LCD Line end enable: 0 = **CLLE** disabled (held LOW) 1 = **CLLE** signal active.
[15:7]	-	Undefined.
[6:0]	LED	Line-end signal delay from the rising-edge of the last panel clock, **CLCP**. Program with number of **CLCDCLK** clock periods, minus 1.

3.3.5 Upper and Lower Panel Frame Base Address Registers

The LCDUPBASE and LCDLPBASE Registers are the color LCD DMA base address registers. They program the base address of the frame buffer. You must initialize LCDUPBASE (and LCDLPBASE for dual-panels) before you enable the controller.

Optionally, you can change the value mid-frame to create double-buffered video displays. These registers are copied to the corresponding current registers at each LCD vertical synchronization. This event causes the LNBU bit, and an optional interrupt, to be generated. You can use the interrupt to reprogram the base address when generating double-buffered video.

The base address must be doubleword-aligned. Bits [2:0] are ignored on write, and have a value of zero when read.

Upper Panel Frame Base Address Register

You use LCDUPBASE for:

- TFT displays
- single-panel STN displays
- the upper panel of dual-panel STN displays.

Figure 3-5 shows the bit assignments.

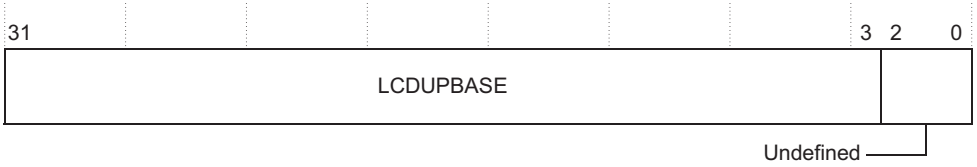


Figure 3-5 LCDUPBASE Register bit assignments

Table 3-6 lists the bit assignments.

Table 3-6 LCDUPBASE Register bit assignments

Bit	Name	Description
[31:3]	LCDUPBASE	LCD upper panel base address. This is the start address of the upper panel frame data in memory and is doubleword-aligned.
[2:0]	-	Undefined.

Lower Panel Frame Base Address Register

Use LCDLPBASE for the lower panel of dual-panel STN displays. Figure 3-6 on page 3-12 shows the bit assignments.

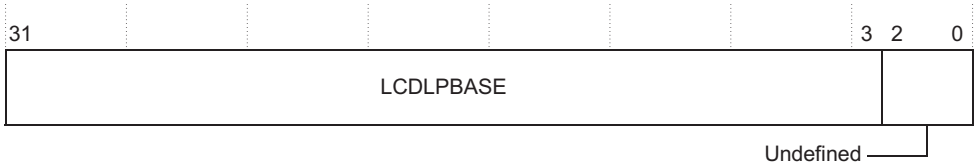


Figure 3-6 LCDLPBASE Register bit assignments

Table 3-7 lists the bit assignments.

Table 3-7 LCDLPBASE Register bit assignments

Bit	Name	Description
[31:3]	LCDLPBASE	LCD lower panel base address. This is the start address of the lower panel frame data in memory and is doubleword-aligned.
[2:0]	-	Undefined.

3.3.6 LCD Control Register

The LCDControl Register controls the operating mode, and the panel pixel parameters. Figure 3-7 shows the bit assignments.

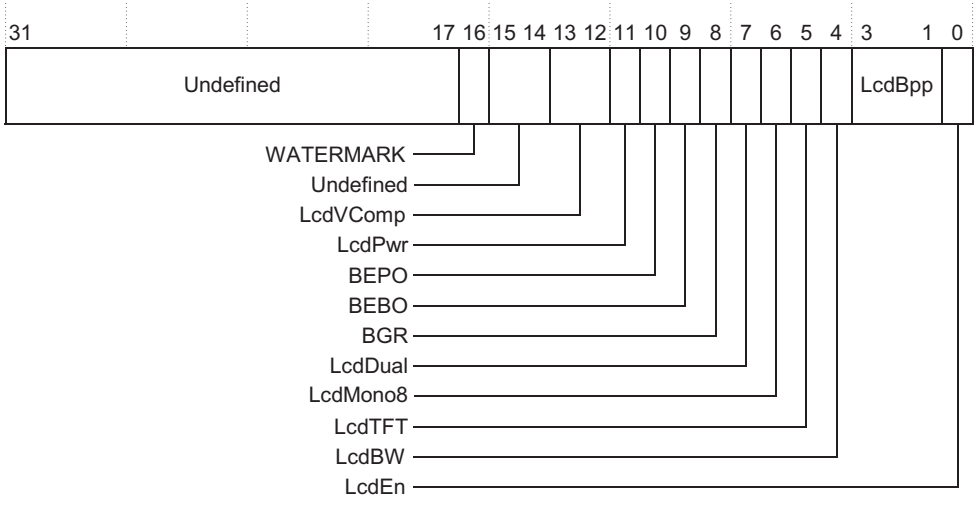


Figure 3-7 LCDControl Register bit assignments

Table 3-8 lists the bit assignments.

Table 3-8 LCDControl Register bit assignments

Bit	Name	Description
[31:17]	-	Undefined.
[16]	WATERMARK	LCD DMA FIFO watermark level: 0 = asserts **HBUSREQM** when either of the DMA FIFOs have four or more empty locations 1 = asserts **HBUSREQM** when either of the DMA FIFOs have eight or more empty locations.
[15:14]	-	Undefined.
[13:12]	LcdVComp	Generate interrupt at: b00 = start of vertical synchronization b01 = start of back porch b10 = start of active video b11 = start of front porch.
[11]	LcdPwr	LCD power enable: 0 = power not gated through to LCD panel and **CLD[23:0]** signals disabled, (held LOW) 1 = power gated through to LCD panel and **CLD[23:0]** signals enabled, (active). See *LCD powering up and powering down sequence support* on page 1-6 for information on LCD power sequencing.
[10]	BEPO	Big-endian pixel ordering within a byte: 0 = little-endian ordering within a byte 1 = big-endian pixel ordering within a byte. The BEPO bit selects between little and big-endian pixel packing for 1, 2, and 4bpp display modes and it has no effect on 8 or 16bpp pixel formats. See *Pixel serializer* on page 2-8 for more information about the data format.
[9]	BEBO	Big-endian byte order: 0 = little-endian byte order 1 = big-endian byte order.
[8]	BGR	RGB or BGR format selection: 0 = RGB normal output 1 = BGR red and blue swapped.
[7]	LcdDual	LCD interface is dual-panel STN: 0 = single-panel LCD is in use 1 = dual-panel LCD is in use.

Table 3-8 LCDControl Register bit assignments (continued)

Bit	Name	Description
[6]	LcdMono8	Monochrome LCD. This has an 8-bit interface. This bit controls whether monochrome STN LCD uses a 4 or 8-bit parallel interface. It has no meaning in other modes, and you must program it to zero. 0 = mono LCD uses 4-bit interface 1 = mono LCD uses 8-bit interface.
[5]	LcdTFT	LCD is TFT: 0 = LCD is an STN display. Use gray scaler 1 = LCD is a TFT display. Do not use gray scaler.
[4]	LcdBW	STN LCD is monochrome (black and white): 0 = STN LCD is color 1 = STN LCD is monochrome. This bit has no meaning in TFT mode.
[3:1]	LcdBpp	LCD bits per pixel: b000 = 1bpp b001 = 2bpp b010 = 4bpp b011 = 8bpp b100 = 16bpp b101 = 24bpp (TFT panel only) b110 = 16bpp 5:6:5 mode b111 = 12bpp 4:4:4 mode.
[0]	LcdEn	CLCDC enable: 0 = LCD signals **CLLP**, **CLCP**, **CLFP**, **CLAC**, and **CLLE** disabled (LOW) 1 = LCD signals **CLLP**, **CLCP**, **CLFP**, **CLAC**, and **CLLE** enabled (HIGH). See *LCD powering up and powering down sequence support* on page 1-6 for information on LCD power sequencing.

3.3.7 Interrupt Mask Set/Clear Register

Setting the LCDIMSC Register bits enables the corresponding raw interrupt LCDStatus bit values to be passed to the LCDInterrupt Register. Figure 3-8 on page 3-15 shows the bit assignments.

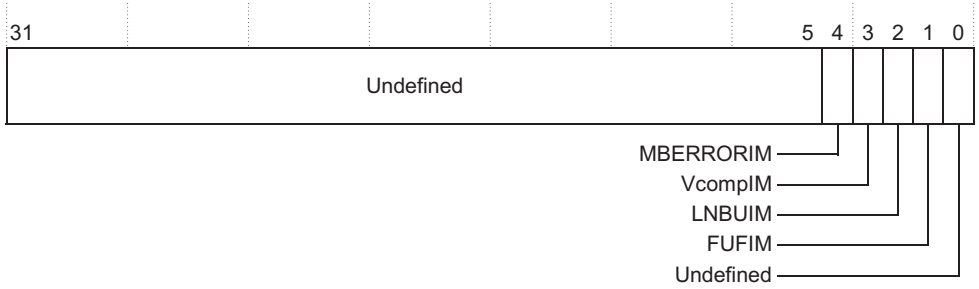


Figure 3-8 LCDIMSC Register bit assignments

Table 3-9 lists the bit assignments.

Table 3-9 LCDIMSC Register bit assignments

Bit	Name	Description
[31:5]	-	Undefined
[4]	MBERRORIM	AHB master error interrupt enable
[3]	VcompIM	Vertical compare interrupt enable
[2]	LNBUIM	LCD next base address update interrupt enable
[1]	FUFIM	FIFO underflow interrupt enable
[0]	-	Undefined

3.3.8 Raw Interrupt Status Register

The LCDRIS Register returns the bits that can generate interrupts when set. Figure 3-9 on page 3-16 shows the bit assignments.

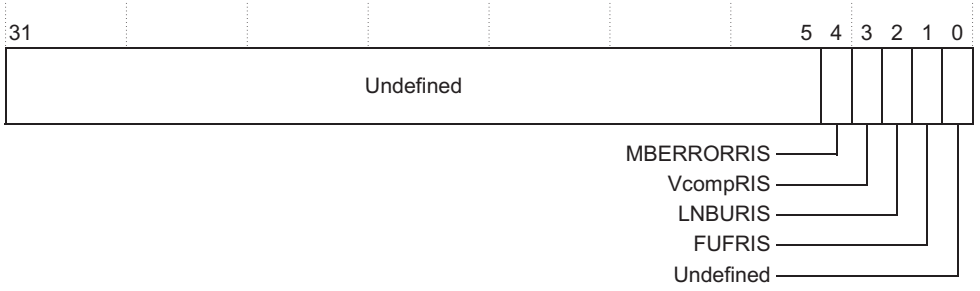


Figure 3-9 LCDRIS Register bit assignments

Table 3-10 lists the bit assignments.

Table 3-10 LCDRIS Register bit assignments

Bit	Name	Description
[31:5]	-	Undefined.
[4]	MBERRORRIS	AHB master bus error status, set when the AHB master encounters a bus error response from a slave.
[3]	VcompRIS	Vertical compare, set when one of the four vertical regions, selected through the LCDControl Register, is reached.
[2]	LNBURIS	LCD next address base update, mode dependent, set when the current base address registers have been successfully updated by the next address registers. Signifies that a new next address can be loaded if you use double buffering.
[1]	FUFRIS	FIFO underflow, set when either the upper or lower DMA FIFOs have been read accessed when empty, causing an underflow condition to occur.
[0]	-	Undefined.

3.3.9 Masked Interrupt Status Register

The LCDMIS Register is a bit-by-bit logical AND of the LCDRIS and LCDIMSC Registers. Interrupt lines correspond to each interrupt. A logical OR of all interrupts is provided to the system interrupt controller. Figure 3-10 on page 3-17 shows the bit assignments.

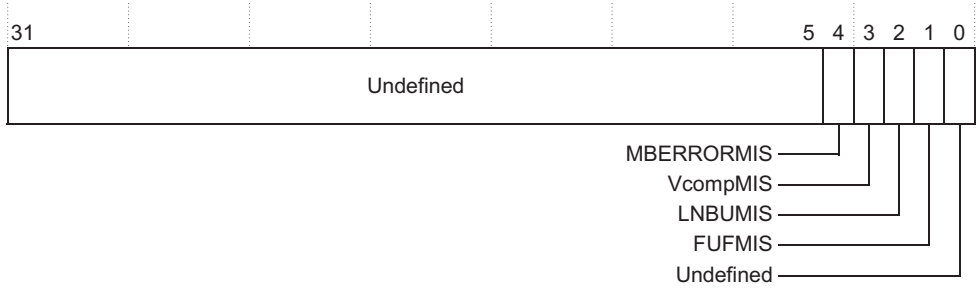


Figure 3-10 LCDMIS Register bit assignments

Table 3-11 lists the bit assignments.

Table 3-11 LCDMIS Register bit assignments

Bit	Name	Description
[31:5]	-	Undefined
[4]	MBERRORMIS	AHB master error interrupt status bit
[3]	VcompMIS	Vertical compare interrupt status bit
[2]	LNBUMIS	LCD next base address update interrupt status bit
[1]	FUFMIS	FIFO underflow interrupt status bit
[0]	-	Undefined

3.3.10 LCD Interrupt Clear Register

Writing a logic 1 to the relevant bit in the LCDICR Register clears the corresponding interrupt. Figure 3-11 on page 3-18 shows the bit assignments.

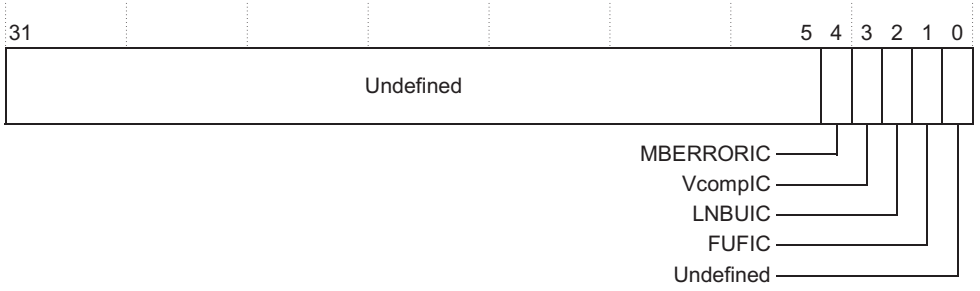


Figure 3-11 LCDICR Register bit assignments

Table 3-12 lists the bit assignments.

Table 3-12 LCDICR Register bit assignments

Bit	Name	Description
[31:5]	-	Undefined
[4]	MBERRORIC	Clear AHB master error interrupt
[3]	VcompIC	Clear vertical compare interrupt
[2]	LNBUIC	Clear LCD next base address update interrupt
[1]	FUFIC	Clear FIFO underflow interrupt
[0]	-	Undefined

3.3.11 LCD Upper and Lower Panel Current Address Value Registers

The LCDUPCURR and LCDLPCURR Registers contain an approximate value of the upper and lower panel data DMA addresses when read:

- LCDUPCURR contains the approximate current upper panel data DMA address
- LCDLPCURR contains the approximate current lower panel data DMA address.

Note

These registers can change at any time and therefore, you can only use them as a mechanism for coarse delay.

3.3.12 256x16-bit Color Palette Registers

The LCDPalette Registers contain 256 palette entries organized as 128 locations of two entries per word.

Note

Only TFT displays use all of the palette entry bits.

Each word location contains two palette entries. This means that 128 word locations are used for the palette. When configured for little-endian byte ordering, bits [15:0] are the lower numbered palette entry and bits [31:16] are the higher numbered palette entry. When configured for big-endian byte ordering, this is reversed because bits [31:16] are the low numbered palette entry and bits [15:0] are the high numbered entry. Figure 3-12 shows the bit assignments for these registers.

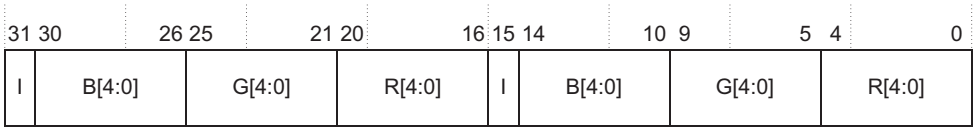


Figure 3-12 LCDPalette Register bit assignments

Table 3-13 lists the bit assignments.

Table 3-13 LCDPalette Register bit assignments

Bit	Name	Description
[31]	I	Intensity/unused.
[30:26]	B[4:0]	Blue palette data.
[25:21]	G[4:0]	Green palette data.
[20:16]	R[4:0]	Red palette data.
[15]	I	Intensity bit. You can use this as the LSB of the R, G, and B inputs to a 6:6:6 TFT display, doubling the number of colors to 64K, where each color has two different intensities.
[14:10]	B[4:0]	Blue palette data.
[9:5]	G[4:0]	Green palette data.
[4:0]	R[4:0]	Red palette data. For STN displays, only the four MSBs, bits [4:1], are used. For monochrome displays, only the red palette data is used. All the palette registers have the same bit fields.

3.3.13 Cursor Image RAM Register

The CursorImage Register contains 256-word wide values that define the image, or images, overlaid by the hardware cursor mechanism. You must always store the image in LBBP mode (little-endian byte, big-endian pixel) mode, as *Image format* on page 2-20 describes. You use two bits to encode color and transparency for each pixel in the cursor.

Depending on the state of bit 0 in the ClcdCrsrConfig Register (see *Cursor Configuration Register* on page 3-21), the cursor image RAM contains either four 32x32 cursor images, or a single 64x64 cursor.

The two colors defined for the cursor are mapped onto values from the ClcdCrsrPalette0 and ClcdCrsrPalette1 Registers. See *Cursor Palette Registers* on page 3-22.

3.3.14 Cursor Control Register

The ClcdCrsrCtrl Register provides access to cursor functions that you use frequently, such as display on/off control for the cursor, and the cursor number for 32x32 bit cursors. Figure 3-13 shows the bit assignments.

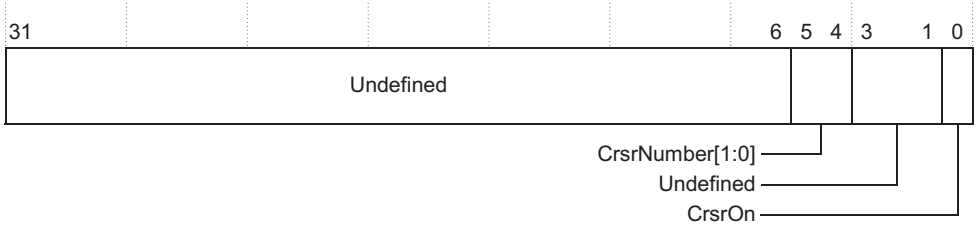


Figure 3-13 ClcdCrsrCtrl Register bit assignments

Table 3-14 lists the bit assignments

Table 3-14 ClcdCrsrCtrl Register bit assignments

Bit	Name	Description
[31:6]	-	Undefined.
[5:4]	CrsrNumber[1:0]	Cursor Image number. This field provides an offset into the cursor image buffer, to you to address enable one of four 32x32 cursors. The images each occupy one quarter of the image memory, with Cursor0 from location 0, followed by Cursor1 from address `0x100`, Cursor2 from `0x200` and Cursor3 from `0x300`. If the cursor size is 64x64, this field has no effect because there is only space for one cursor of this size in the image buffer. If the cursor size is 32x32: b11 = Cursor3 b10 = Cursor2 b01 = Cursor1 b00 = Cursor0. Frame Synchronization: Similar synchronization rules apply to the cursor number as apply to the cursor coordinates: • if CrsrFrameSync is 1, the displayed cursor image is only changed during the vertical frame blanking period. • if CrsrFrameSync is 0, the cursor image index is changed immediately, even if the cursor is currently being scanned.
[3:1]	-	Undefined.
[0]	CrsrOn	0 = Cursor not displayed 1 = Cursor is displayed.

3.3.15 Cursor Configuration Register

The ClcdCrsrConfig Register provides overall configuration information for the hardware cursor. Figure 3-14 shows the bit assignments.

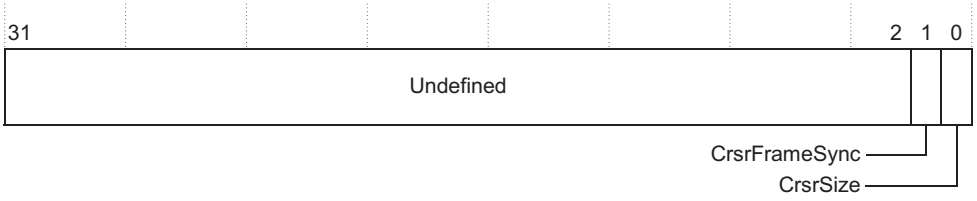


Figure 3-14 ClcdCrsrConfig Register bit assignments

Table 3-15 lists the bit assignments.

Table 3-15 ClcdCrsrConfig Register bit assignments

Bit	Name	Description
[31:2]	-	Undefined
[1]	CrsrFrameSync	0 = Cursor coordinates asynchronous 1 = Cursor coordinates synchronized to frame synchronization pulse
[0]	CrsrSize	0 = 32x32 pixel cursor 1 = 64x64 pixel cursor

3.3.16 Cursor Palette Registers

The ClcdCrsrPalette0 and ClcdCrsrPalette1 Registers provide color palette information for the visible colors of the cursor:

- Color0 is mapped through CrsrPalette0
- Color1 is mapped through CrsrPalette1.

The registers provide 24-bit RGB values that are displayed according to the abilities of the LCD panel, in the same way as the frame-buffers palette output is displayed.

In mono STN, only Red[7:4] are significant and, in STN color Red[7:4], Blue[7:4], and Green[7:4] are significant. In 24 bpp, all 24 bits of the palette registers are significant. Figure 3-15 shows the bit assignments.

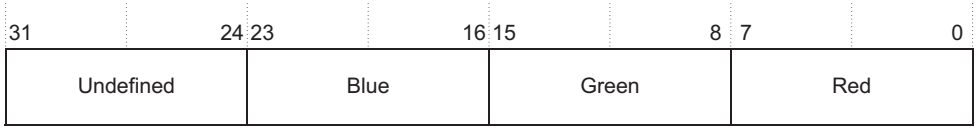


Figure 3-15 ClcdCrsrPalette0 and ClcdCrsrPalette1 Register bit assignments

Table 3-16 lists the bit assignments.

Table 3-16 ClcdCrsrPalette Register bit assignments

Bit	Name	Description
[31:24]	-	Undefined

Table 3-16 ClcdCrsrPalette Register bit assignments (continued)

Bit	Name	Description
[23:16]	Blue	Blue color component
[15:8]	Green	Green color component
[7:0]	Red	Red color component

3.3.17 Cursor XY Position Register

The ClcdCrsrXY Register defines the distance of the top-left edge of the cursor from the top-left side of the cursor overlay as shown in Figure 2-12 on page 2-19. Figure 3-16 shows the bit assignments.

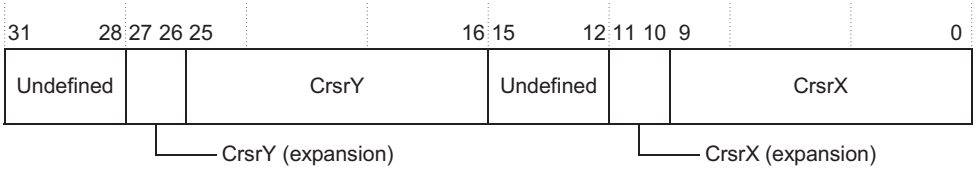


Figure 3-16 ClcdCrsrXY Register bit assignments

Table 3-17 lists the bit assignments.

Table 3-17 ClcdCrsrXY Register bit assignments

Bit	Name	Description
[31:28]	-	Undefined.
[27:26]	CrsrY (expansion)	Reserved for coordinate expansion. You must write this as zero.
[25:16]	CrsrY	Y ordinate of the cursor origin measured in pixels. When 0, the top edge of the cursor is at the top of the display.
[15:12]	-	Undefined.
[11:10]	CrsrX (expansion)	Reserved for coordinate expansion. You must write this as zero.
[9:0]	CrsrX	X ordinate of the cursor origin measured in pixels. When 0, the left edge of the cursor is at the left edge of the display.

Frame synchronization

If CrsrFrameSync is 0, the cursor position changes immediately, even if the cursor is currently being scanned. If CrsrFrameSync is 1, the cursor position only changes during the next vertical frame blanking period.

3.3.18 Cursor Clip Position Register

The ClcdCrsrClip Register defines the distance from the top-left edge of the cursor image, to the first displayed pixel in the cursor image as shown in Figure 2-12 on page 2-19. Figure 3-17 shows the bit assignments.

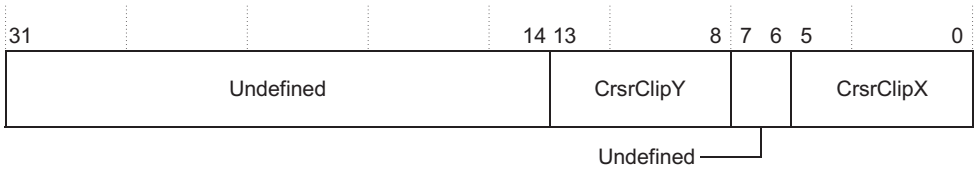


Figure 3-17 ClcdCrsrClip Register bit assignments

Table 3-18 lists the bit assignments.

Table 3-18 ClcdCrsrClip Register bit assignments

Bit	Name	Description
[31:14]	-	Undefined.
[13:8]	CrsrClipY	Distance from top of cursor image to the first displayed pixel in cursor. When 0, the first displayed pixel is from the top line of the cursor image.
[7:6]	-	Undefined.
[5:0]	CrsrClipX	Distance from left edge of cursor image to the first displayed pixel of cursor. When 0, the first pixel of the cursor line is displayed.

Frame synchronization

Different synchronization rules apply to the Cursor Clip Registers than apply to the cursor coordinates.

If CrsrFrameSync is 0, the cursor clip point changes immediately, even if the cursor is currently being scanned.

If CrsrFrameSync is 1, the displayed cursor image only changes during the vertical frame blanking period, providing that the cursor position has been updated since you programmed the Clip Register. Therefore, when programming, you must write the Clip Register before the Position Register (ClcdCrsrXY) to ensure that in a given frame, the clip and position information is coherent.

3.3.19 Cursor Interrupt Mask Set/Clear Register

The ClcdCrsrIMSC Register enables the cursor interrupt to the processor. Figure 3-18 shows the bit assignments.

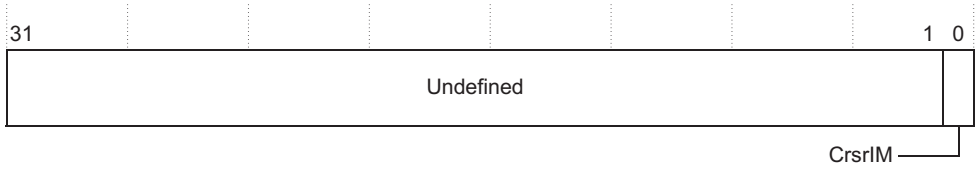


Figure 3-18 ClcdCrsrIMSC Register bit assignments

Table 3-19 lists the bit assignments.

Table 3-19 ClcdCrsrIMSC Register bit assignments

Bit	Name	Description
[31:1]	-	Undefined.
[0]	CrsrIM	When set, the cursor interrupts the processor immediately after reading of the last word of cursor image. When clear, the cursor never interrupts the processor.

3.3.20 Cursor Interrupt Clear Register

Software uses the ClcdCrsrICR Register to clear the cursor interrupt status, and the cursor interrupt signal to the processor. Figure 3-19 shows the bit assignments.

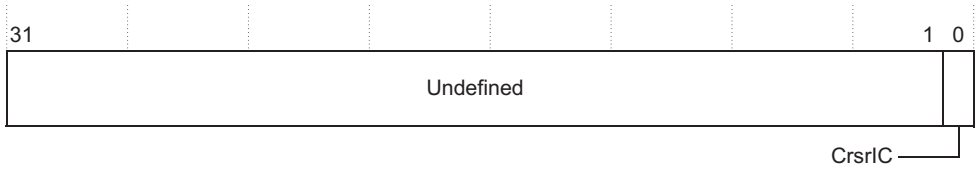


Figure 3-19 ClcdCrsrICR Register bit assignments

Table 3-20 lists the bit assignments.

Table 3-20 ClcdCrsrICR Register bit assignments

Bit	Name	Description
[31:1]	-	Undefined.
[0]	CrsrIC	When set, the cursor interrupt status is cleared. Writing 0 to this bit has no effect.

3.3.21 Cursor Raw Interrupt Status Register

Set the ClcdCrsrRIS Register to indicate a cursor interrupt. When enabled, it controls the state of the interrupt signal to the system interrupt controller. Figure 3-20 shows the bit assignments.

Note

The ClcdCrsrRIS Register is valid regardless of the state of the CrsrIM bit.

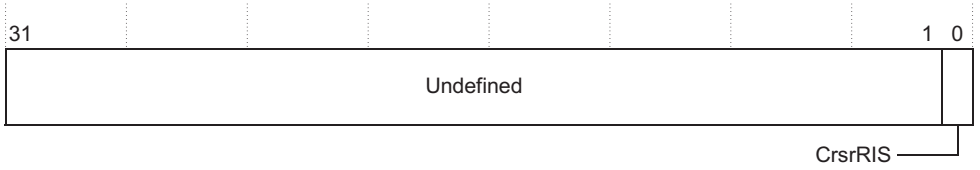


Figure 3-20 ClcdCrsrRIS Register bit assignments

Table 3-21 lists the bit assignments.

Table 3-21 ClcdCrsrRIS Register bit assignments

Bit	Name	Description
[31:1]	-	Undefined.
[0]	CrsrRIS	The cursor interrupt status is set immediately after the last data read from the cursor image for the current frame. You clear this bit by writing to the CrsrIC bit in the ClcdCrsrICR Register. See *Cursor Interrupt Clear Register* on page 3-25.

3.3.22 Cursor Masked Interrupt Status Register

Set the ClcdCrsrMIS Register to indicate a cursor interrupt, providing that the interrupt bit is not masked. Figure 3-21 on page 3-27 shows the bit assignments.

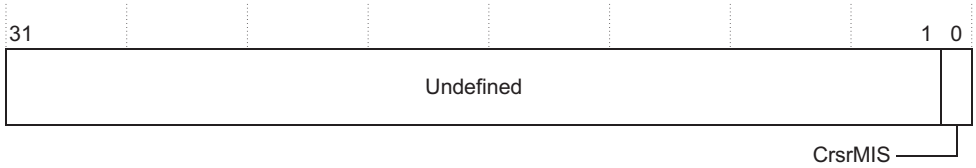


Figure 3-21 ClcdCrsrMIS Register bit assignments

Table 3-22 lists the bit assignments for this register.

Table 3-22 ClcdCrsrMIS Register bit assignments

Bit	Name	Description
[31:1]	-	Undefined.
[0]	CrsrMIS	The cursor interrupt status is set immediately after the last data read from the cursor image for the current frame, providing that you set the corresponding bit in the ClcdCrsrIMSC Register. The bit remains clear if the ClcdCrsrIMSC Register is clear. You clear this bit by writing to the ClcdCrsrICR Register. See *Cursor Interrupt Clear Register* on page 3-25.

3.3.23 Peripheral Identification Registers

The CLCDPeriphID0-3 Registers, are four 8-bit read-only registers that you can treat as a single 32-bit register. Table 3-23 lists the registers that provide the peripheral options.

Table 3-23 Peripheral Identification Register options

Bit	Description
PartNumber[11:0]	Identifies the peripheral. The three-digit product code is `0x111`.
DesignerID[19:12]	Identifies the designer. This is set to `0x41` to indicate that ARM designed the peripheral.
Revision[23:20]	Identifies the revision number of the peripheral. The revision number starts from 0 and is revision-dependent.
Configuration[31:24]	Identifies the configuration option of the peripheral. The configuration value is 0.

Figure 3-22 on page 3-28 shows the bit assignments for these registers.

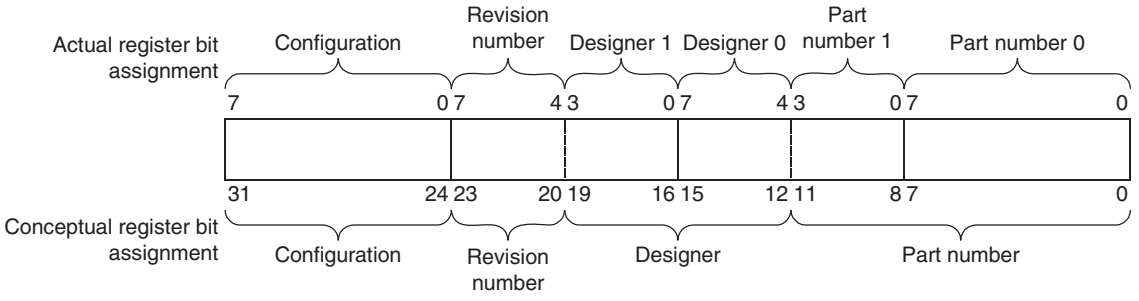


Figure 3-22 CLCDPeriphID0-3 Register bit assignments

The following sections describe the four 8-bit Peripheral Identification Registers:

- *Peripheral Identification Register 0*
- *Peripheral Identification Register 1*
- *Peripheral Identification Register 2* on page 3-29
- *Peripheral Identification Register 3* on page 3-30.

Peripheral Identification Register 0

The CLCDPeriphID0 Register is hard-coded and the fields in the register control the reset value. Figure 3-23 shows the bit assignments.

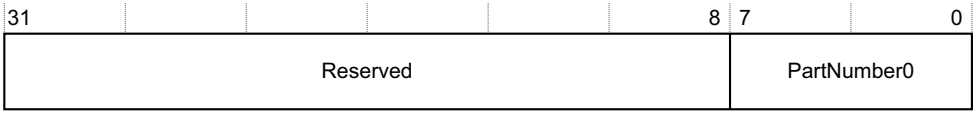


Figure 3-23 CLCDPeriphID0 Register bit assignments

Table 3-24 lists the bit assignments.

Table 3-24 CLCDPeriphID0 Register bit assignments

Bit	Name	Description
[31:8]	-	Undefined
[7:0]	PartNumber0	These bits read back as `0x11`

Peripheral Identification Register 1

The CLCDPeriphID1 Register is hard-coded and the fields in the register control the reset value. Figure 3-24 on page 3-29 shows the bit assignments.

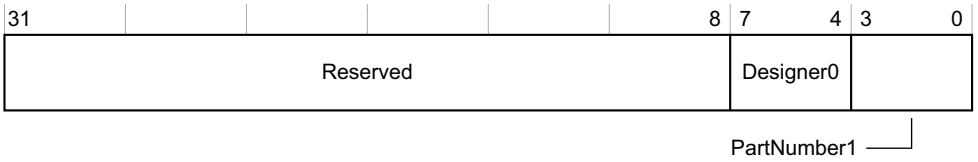


Figure 3-24 CLCDPeriphID1 Register bit assignments

Table 3-25 lists the bit assignments.

Table 3-25 CLCDPeriphID1 Register bit assignments

Bit	Name	Description
[31:8]	-	Undefined
[7:4]	Designer0	These bits read back as `0x1`
[3:0]	PartNumber1	These bits read back as `0x1`

Peripheral Identification Register 2

The CLCDPeriphID2 Register is hard-coded and the fields in the register control the reset value. Figure 3-25 shows the bit assignments.

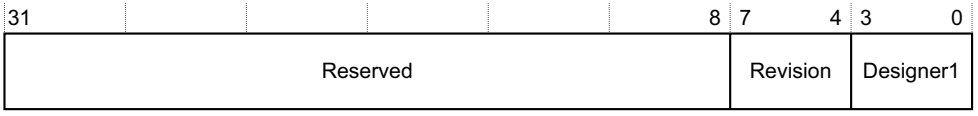


Figure 3-25 CLCDPeriphID2 Register bit assignments

Table 3-26 lists the bit assignments.

Table 3-26 CLCDPeriphID2 Register bit assignments

Bit	Name	Description
[31:8]	-	Undefined
[7:4]	Revision	These bits provide the revision number of the controller: `0x0` = revision r0p0 `0x1` = revision r0p1 `0x2` = revision r0p2.
[3:0]	Designer1	These bits read back as `0x4`

Peripheral Identification Register 3

The CLCDPeriphID3 Register is hard-coded and the fields in the register control the reset value. Figure 3-26 shows the bit assignments.

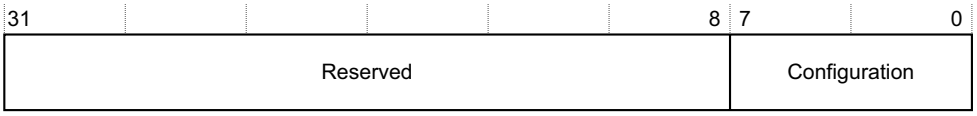


Figure 3-26 CLCDPeriphID3 Register bit assignments

Table 3-27 lists the bit assignments.

Table 3-27 CLCDPeriphID3 Register bit assignments

Bit	Name	Description
[31:8]	-	Undefined
[7:0]	Configuration	These bits read back as `0x0`

3.3.24 PrimeCell Identification Registers

The CLCDPCellID0-3 Registers are four 8-bit read-only registers that you can treat conceptually as a single 32-bit register. The register is a standard cross-peripheral identification system. Figure 3-27 shows the bit assignments.

The following subsections describe the four 8-bit PrimeCell Identification Registers:

- *PrimeCell Identification Register 0* on page 3-31
- *PrimeCell Identification Register 1* on page 3-31
- *PrimeCell Identification Register 2* on page 3-32
- *PrimeCell Identification Register 3* on page 3-32.

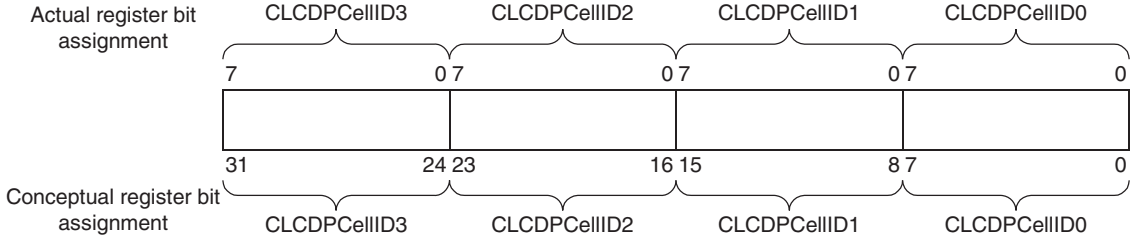


Figure 3-27 CLCDPCellID0-3 Register bit assignments

PrimeCell Identification Register 0

The CLCDPCellID0 Register is hard-coded and the fields in the register control the reset value. Figure 3-28 shows the bit assignments.

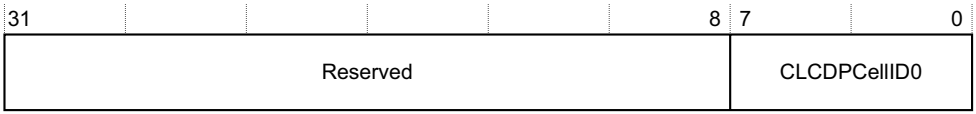


Figure 3-28 CLCDPCellID0 Register bit assignments

Table 3-28 lists the bit assignments.

Table 3-28 CLCDPCellID0 Register bit assignments

Bit	Name	Description
[31:8]	-	Undefined
[7:0]	CLCDPCellID0	These bits read back as `0x0D`

PrimeCell Identification Register 1

The CLCDPCellID1 Register is hard-coded, and the fields in the register control the reset value. Figure 3-29 shows the bit assignments.

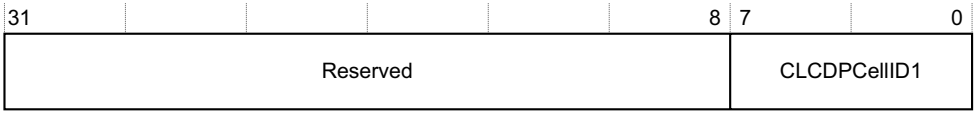


Figure 3-29 CLCDPCellID1 Register bit assignments

Table 3-29 lists the bit assignments.

Table 3-29 CLCDPCellID1 Register bit assignments

Bit	Name	Description
[31:8]	-	Undefined
[7:0]	CLCDPCellID1	These bits read back as `0xF0`

PrimeCell Identification Register 2

The CLCDPCellID2 Register is hard-coded and the fields in the register control the reset value. Figure 3-30 shows the bit assignments.

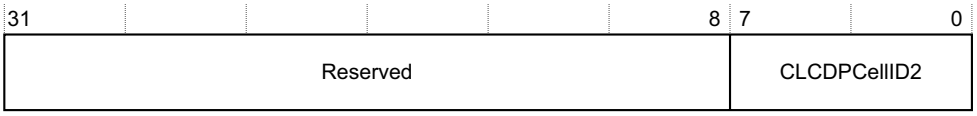


Figure 3-30 CLCDPCellID2 Register bit assignments

Table 3-30 lists the bit assignments.

Table 3-30 CLCDPCellID2 Register bit assignments

Bit	Name	Description
[31:8]	-	Undefined
[7:0]	CLCDPCellID2	These bits read back as `0x05`

PrimeCell Identification Register 3

The CLCDPCellID3 Register is hard-coded and the fields in the register control the reset value. Figure 3-31 shows the bit assignments.

Figure 3-31 CLCDPCellID3 Register bit assignments

Table 3-31 lists the bit assignments.

Table 3-31 CLCDPCellID3 Register bit assignments

Bit	Name	Description
[31:8]	-	Undefined
[7:0]	CLCDPCellID3	These bits read back as `0xB1`

3.4 Interrupts

The controller generates five interrupts. Four of these are individual maskable active HIGH interrupts:

- *Master bus error interrupt, CLCDMBEINTR*
- *Vertical compare interrupt, CLCDVCOMPINTR*
- *Next base address update interrupt, CLCDLNBUINTR* on page 3-34
- *FIFO underflow interrupt, CLCDFUFINTR* on page 3-34.

The outputs are also output as a combined single interrupt, **CLCDINTR**.

You can enable or disable each of the four individual maskable interrupts by changing the mask bits in the LCDIMSC Register. See *Interrupt Mask Set/Clear Register* on page 3-14.

Provision of individual outputs in addition to a combined interrupt output enable you to use either a global interrupt service routine, or modular device drivers to handle interrupts.

You can read the status of the individual interrupt sources from the LCDRIS Register. See *Raw Interrupt Status Register* on page 3-15.

3.4.1 Master bus error interrupt, CLCDMBEINTR

The master bus error interrupt is asserted when an ERROR response is received by the master interface during a transaction with a slave. When such an error is encountered, the master interface enters an error state and remains in this state until clearance of the error has been signaled to it. When the respective interrupt service routine is complete, you can clear the master bus error interrupt by writing a 1 to the MBERRORIC bit in the LCDICR Register. This action releases the master interface from its ERROR state to the start of FRAME state, and enables a fresh frame of data display to be initiated.

3.4.2 Vertical compare interrupt, CLCDVCOMPINTR

The vertical compare interrupt asserts when one of four vertical display regions, selected using the LCD Control Register, is reached. You can make the interrupt occur at the start of:

- vertical synchronization
- back porch
- active video
- front porch.

You can clear the interrupt by writing a 1 to the VcompIC bit in the LCDICR Register.

3.4.3 Next base address update interrupt, CLCDLNBUINTR

The LCD next base address update interrupt asserts when either the LCDUPBASE or LCDLPBASE values have been transferred to the LCDUPCURR or LCDLPCURR incrementers respectively. This signals to the system that it is safe to update the LCDUPBASE or the LCDLPBASE Registers with new frame base addresses if required.

You can clear the interrupt by writing a 1 to the LNBUIC bit in the LCDICR Register.

3.4.4 FIFO underflow interrupt, CLCDFUFINTR

The FIFO underflow interrupt asserts when internal data is requested from an empty DMA FIFO. Internally, individual upper and lower panel DMA FIFO underflow interrupt signals are generated and **CLCDFUFINTR** is the single combined version of these.

You can clear the interrupt by writing a 1 to the FUFIC bit in the LCDICR Register.

Chapter 4
Programmer's Model for Test

This chapter describes the additional logic for functional verification and production testing. It contains the following sections:

4.1 Scan testing

The controller enables:

- the automatic insertion of scan test cells
- the use of *Automatic Test Pattern Generation* (ATPG).

This is the recommended method for manufacturing test.

During scan testing, you must drive **SCANENABLE** HIGH. For normal use, you must drive **SCANENABLE** LOW.

4.2 Test registers

This section describes the test registers. All register addresses in the controller are fixed relative to its base address. Table 4-1 lists the registers in base offset order.

Table 4-1 Test register summary

Name	Base offset	Type	Reset value	Description
LCDTCR	0xF00	R/W	0x0	*Integration Test Control Register*
LCDITOP1	0xF04	R/W	0x00	*Integration Test Output Register 1* on page 4-4
LCDITOP2	0xF08	R/W	0x00000000	*Integration Test Output Register 2* on page 4-5

4.2.1 Integration Test Control Register

The LCDTCR Register controls the integration test mode of the controller for integration testing. Do not use this register during normal operation of the controller. Figure 4-1 shows the bit assignments.

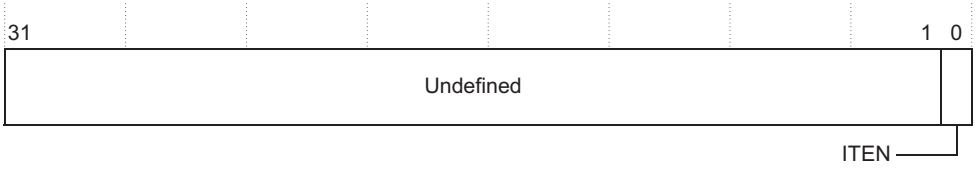


Figure 4-1 LCDTCR Register bit assignments

Table 4-2 lists the bit assignments.

Table 4-2 LCDTCR Register bit assignments

Bit	Name	Description
[31:1]	-	Undefined.
[0]	ITEN	Integration test enable: 0 = normal CLCDC operation 1 = integration test mode enabled.

4.2.2 Integration Test Output Register 1

The LCDITOP1 Register controls the on-chip outputs of the controller. See *On-chip signals* on page A-6. Figure 4-2 shows the bit assignments.

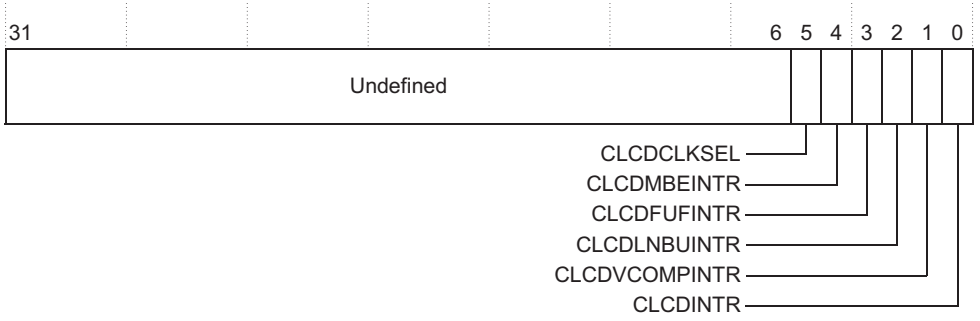


Figure 4-2 LCDITOP1 Register bit assignments

Table 4-3 lists the bit assignments.

Table 4-3 LCDITOP1 Register bit assignments

Bit	Name	Description
[31:6]	-	Undefined.
[5]	CLCDCLKSEL	Writes specify the value to drive on **CLCDCLKSEL**, when in integration test mode. A read returns the value of the **CLCDCLKSEL** signal at the output of the test multiplexor.
[4]	CLCDMBEINTR	Writes specify the value to drive on **CLCDMBEINTR**, when in integration test mode. A read returns the value of **CLCDMBEINTR** at the output of the test multiplexor.
[3]	CLCDFUFINTR	Writes specify the value to drive on **CLCDFUFINTR**, when in integration test mode. A read returns the value of **CLCDFUFINTR** at the output of the test multiplexor.
[2]	CLCDLNBUINTR	Writes specify the value to drive on **CLCDLNBUINTR**, when in integration test mode. A read returns the value of **CLCDLNBUINTR** at the output of the test multiplexor.
[1]	CLCDVCOMPINTR	Writes specify the value to drive on **CLCDVCOMPINTR**, when in integration test mode. A read returns the value of **CLCDVCOMPINTR** at the output of the test multiplexor.
[0]	CLCDINTR	Writes specify the value to drive on **CLCDINTR**, when in integration test mode. A read returns the value of **CLCDINTR** at the output of the test multiplexor.

4.2.3 Integration Test Output Register 2

The LCDITOP2 Register controls the primary outputs of the controller. See *External pad interface signals* on page A-5. Figure 4-3 shows the bit assignments.

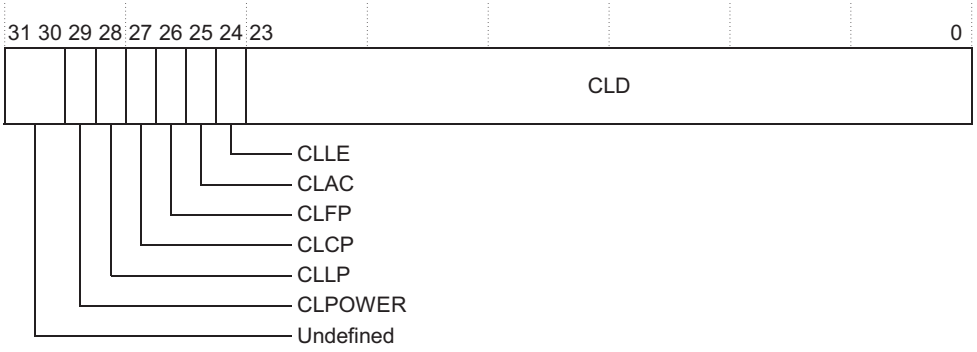


Figure 4-3 LCDITOP2 Register bit assignments

Table 4-4 lists the bit assignments.

Table 4-4 LCDITOP2 Register bit assignments

Bit	Name	Description
[31:30]	-	Undefined.
[29]	CLPOWER	Writes specify the value to drive on **CLPOWER**, when in integration test mode.
[28]	CLLP	Writes specify the value to drive on **CLLP**, when in integration test mode.
[27]	CLCP	Writes specify the value to drive on **CLCP**, when in integration test mode.
[26]	CLFP	Writes specify the value to drive on **CLFP**, when in integration test mode.
[25]	CLAC	Writes specify the value to drive on **CLAC**, when in integration test mode.
[24]	CLLE	Writes specify the value to drive on **CLLE**, when in integration test mode.
[23:0]	CLD	Writes specify the values to drive on **CLD**, when in integration test mode.

Appendix A
Signal Descriptions

This appendix describes the controller signals. It contains the following sections:

A.1 AHB slave interface signals

Table A-1 lists the AHB slave interface signals.

Table A-1 AHB slave interface signals

Signal	AMBA equivalent[a]
HADDRS[11:2]	**HADDR[31:0]**
HCLK	**HCLK**
HRDATAS[31:0]	**HRDATA[31:0]**
HREADYSin	**HREADYin**
HREADYSout	**HREADYout**
HRESETn	**HRESETn**
HRESPS[1:0]	**HRESP[1:0]**
HSELCLCD	**HSELx**
HTRANSS[1:0]	**HTRANS[1:0]**
HWDATAS[31:0]	**HWDATA[31:0]**
HWRITES	**HWRITE**

a. See the *AMBA Specification (Rev 2.0)* for a description of these signals.

HRESPS[1:0] deviates from the functionality described in the *AMBA Specification (Rev 2.0).* Table A-2 lists its functionality.

Table A-2 HRESPS bus

Signal	Type	Destination	Description
HRESPS[1:0]	Output	Multiplexor	The controller only supports the OKAY response. This bus is tied LOW.

A.2 AHB master interface signals

Table A-3 lists the AHB master interface signals.

Table A-3 AHB master interface signals

Signal	AMBA equivalent[a]
HADDRM[31:0]	**HADDR[31:0]**
HBURSTM[2:0][b]	**HBURST[2:0]**
HBUSREQM	**HBUSREQx**
HGRANTM	**HGRANTx**
HLOCK	**HLOCKx**
HPROT[3:0]	**HPROT[3:0]**
HRDATAM[63:0]	**HRDATA[63:0]**
HREADYMin	**HREADY**
HRESPM[1:0][b]	**HRESP[1:0]**
HSIZEM[2:0][c]	**HSIZE[2:0]**
HTRANSM[1:0]	**HTRANS[1:0]**
HWRITEM	**HWRITE**

a. See the *AMBA Specification (Rev 2.0)* for a description of these signals.
b. See *HBURSTM and HRESPM signals* on page A-4 for a description of these buses.
c. See *Steady state signals* on page A-4 for a description of these signals.

The following signals deviate from the functionality that the *AMBA Specification (Rev 2.0)* describes:

- **HBURSTM[2:0]**
- **HRESPM[1:0]**
- **HSIZEM[2:0]**
- **HWRITEM**.

The differences are described in:

- *HBURSTM and HRESPM signals* on page A-4
- *Steady state signals* on page A-4.

A.2.1 HBURSTM and HRESPM signals

The controller does not implement all of the functionality that the *AMBA Specification (Rev 2.0)* describes. Table A-4 lists them.

Table A-4 HBURSTM and HRESPM signals

Signal	Type	Destination	Description
HBURSTM[2:0]	Output	AHB bus	Supports the following burst transfers: • SINGLE • INCR • INCR4 • INCR8 • INCR16. **Note** The controller does not generate wrapping bursts.
HRESPM[1:0]	Input	AHB bus	Supports the following slave responses: • OKAY • ERROR • RETRY. **Note** The controller does not support SPLIT responses.

A.2.2 Steady state signals

Some AHB master interface signals are tied to a steady state. Table A-5 lists them.

Table A-5 Steady state signals

Signal	Type	Destination	Description
HSIZEM[2:0]	Output	AHB bus	The controller only supports doubleword read accesses and therefore it: • ties **HSIZEM[2]** LOW • ties **HSIZEM[1:0]** HIGH.
HWRITEM	Output	AHB bus	The controller only supports read accesses and therefore it ties this signal LOW.

A.3 External pad interface signals

Table A-6 lists the external pad interface signals.

Table A-6 External pad interface signals

Signal	Type	Source/ destination	Description
CLAC	Output	Pad	AC bias drive (STN) or Data enable output (TFT)
CLCP	Output	Pad	LCD panel clock
CLD[23:0]	Output	Pad	LCD panel data
CLFP	Output	Pad	Frame pulse (STN) or Vertical synchronization pulse (TFT)
CLLE	Output	Pad	Line end signal
CLLP	Output	Pad	Line synchronization pulse (STN) or Horizontal synchronization pulse (TFT)
CLPOWER	Output	Pad	LCD panel power enable

A.4 On-chip signals

You must provide a free-running reference clock, **CLCDCLK**. By default, it is assumed to be asynchronous to **HCLK**.

The reset inputs are asynchronously asserted, but synchronously removed for each of the clock domains in the controller. This ensures that logic is reset even if clocks are not present, to avoid any static power consumption problems at power-up. Each clock domain has an individual reset to simplify the process of inserting scan test cells.

Table A-7 lists the on-chip signals required in addition to the AHB signals.

Table A-7 On-chip signals

Signal	Type	Source/ destination	Description
CLCDCLK	Input	Clock multiplexor	Reference clock for the controller.
CLCDCLKSEL	Output	Clock multiplexor	Reference clock select signal. The CLKSEL bit of the LCDTiming2 Register drives the **CLCDCLKSEL** signal which selects the source for the reference clocks: **CLKSEL = 0** **HCLK** selected. **CLKSEL = 1** **CLCDCLK** selected.
CLCDFUFINTR	Output	Interrupt controller	FIFO underflow interrupt, active HIGH. A combined interrupt generated when either of the upper or lower panel DMA FIFOs underflow.
CLCDINTR	Output	Interrupt controller	Interrupt, active HIGH. A single combined interrupt generated as an OR function of the four individually maskable interrupts.
CLCDLNBUINTR	Output	Interrupt controller	Next base address update interrupt, active HIGH.
CLCDMBEINTR	Output	Interrupt controller	Master bus error interrupt, active HIGH.
CLCDVCOMPINTR	Output	Interrupt controller	Vertical region compare interrupt, active HIGH.
nCLCLKRESET	Input	Reset multiplexor	Reset signal to the **CLCDCLK** domain, active LOW. The reset controller must use **HRESETn** to assert **nCLCLKRESET** asynchronously, but negate it synchronously with **CLCDCLK**.
nCLCDCLK	Input	Clock multiplexor	Inverse of **CLCDCLK**.

A.5 Scan signals

Table A-8 lists the scan signals.

Table A-8 Scan signals

Signal	Type	Source/ destination	Description
SCANENABLE	Input	Test controller	Scan test enable. You must assert this signal HIGH to bypass the DMA FIFOs during scan testing.
SCANINHCLK	Input	Test controller	Input scan signal for the **HCLK** domain.
SCANINCLCDCLK	Input	Test controller	Input scan signal for the **CLCDCLK** domain.
SCANINnCLCDCLK	Input	Test controller	Input scan signal for the **nCLCDCLK** domain.
SCANOUTHCLK	Output	Test controller	Output scan signal for the **HCLK** domain.
SCANOUTCLCDCLK	Output	Test controller	Output scan signal for the **CLCDCLK** domain.
SCANOUTnCLCDCLK	Output	Test controller	Output scan signal for the **nCLCDCLK** domain.

A.6 LCD panel signal multiplexing details

The **CLLP**, **CLAC**, **CLFP**, **CLCP**, and **CLLE** signals are common, but the **CLD[23:0]** bus has ten modes of operation corresponding to:

- TFT 24-bit interface
- TFT 16-bit interface, 1:5:5:5
- TFT 16-bit interface, 5:6:5
- TFT 12-bit interface, 4:4:4
- color STN single-panel
- color STN dual-panel
- 4-bit mono STN single-panel
- 4-bit mono STN dual-panel
- 8-bit mono STN single-panel
- 8-bit mono STN dual-panel.

Table A-9 lists the **CLD[23:0]** pins that supply the pixel data to the STN panel for each of the STN modes of operation.

Note

Table A-9 uses the following terms:

CLSTN Color lower panel STN, single-panel.

CUSTN Color upper panel STN, dual- and/or single-panel.

MLSTN Mono lower panel STN, single-panel.

MUSTN Mono upper panel STN, dual- and/or single-panel.

Table A-9 STN panel signal multiplexing

External pin	Color STN single-panel	Color STN dual-panel	4-bit mono STN single-panel	4-bit mono STN dual-panel	8-bit mono STN single-panel	8-bit mono STN dual-panel
CLD[23]	0	0	0	0	0	0
CLD[22]	0	0	0	0	0	0
CLD[21]	0	0	0	0	0	0
CLD[20]	0	0	0	0	0	0
CLD[19]	0	0	0	0	0	0

Table A-9 STN panel signal multiplexing (continued)

External pin	Color STN single-panel	Color STN dual-panel	4-bit mono STN single-panel	4-bit mono STN dual-panel	8-bit mono STN single-panel	8-bit mono STN dual-panel
CLD[18}	0	0	0	0	0	0
CLD[17]	0	0	0	0	0	0
CLD[16]	0	0	0	0	0	0
CLD[15]	0	**CLSTN[7]**	0	0	0	**MLSTN[7]**
CLD[14]	0	**CLSTN[6]**	0	0	0	**MLSTN[6]**
CLD[13]	0	**CLSTN[5]**	0	0	0	**MLSTN[5]**
CLD[12]	0	**CLSTN[4]**	0	0	0	**MLSTN[4]**
CLD[11]	0	**CLSTN[3]**	0	**MLSTN[3]**	0	**MLSTN[3]**
CLD[10]	0	**CLSTN[2]**	0	**MLSTN[2]**	0	**MLSTN[2]**
CLD[9]	0	**CLSTN[1]**	0	**MLSTN[1]**	0	**MLSTN[1]**
CLD[8]	0	**CLSTN[0]**	0	**MLSTN[0]**	0	**MLSTN[0]**
CLD[7]	**CUSTN[7]**	**CUSTN[7]**	0	0	**MUSTN[7]**	**MUSTN[7]**
CLD[6]	**CUSTN[6]**	**CUSTN[6]**	0	0	**MUSTN[6]**	**MUSTN[6]**
CLD[5]	**CUSTN[5]**	**CUSTN[5]**	0	0	**MUSTN[5]**	**MUSTN[5]**
CLD[4]	**CUSTN[4]**	**CUSTN[4]**	0	0	**MUSTN[4]**	**MUSTN[4]**
CLD[3]	**CUSTN[3]**	**CUSTN[3]**	**MUSTN[3]**	**MUSTN[3]**	**MUSTN[3]**	**MUSTN[3]**
CLD[2]	**CUSTN[2]**	**CUSTN[2]**	**MUSTN[2]**	**MUSTN[2]**	**MUSTN[2]**	**MUSTN[2]**
CLD[1]	**CUSTN[1]**	**CUSTN[1]**	**MUSTN[1]**	**MUSTN[1]**	**MUSTN[1]**	**MUSTN[1]**
CLD[0]	**CUSTN[0]**	**CUSTN[0]**	**MUSTN[0]**	**MUSTN[0]**	**MUSTN[0]**	**MUSTN[0]**

Table A-10 on page A-10 lists the **CLD[23:0]** pins that supply the pixel data to the TFT panel for each of the TFT modes of operation.

Table A-10 TFT panel signal multiplexing

External pin	TFT 24-bit	TFT 16-bit (1:5:5:5)	TFT 16-bit (5:6:5)	TFT 12-bit (4:4:4)
CLD[23]	**BLUE7**	**BLUE4**	**BLUE4**	**BLUE3**
CLD[22]	**BLUE6**	**BLUE3**	**BLUE3**	**BLUE2**
CLD[21]	**BLUE5**	**BLUE2**	**BLUE2**	**BLUE1**
CLD[20]	**BLUE4**	**BLUE1**	**BLUE1**	**BLUE0**
CLD[19]	**BLUE3**	**BLUE0**	**BLUE0**	Reserved
CLD[18]	**BLUE2**	Intensity bit	Reserved	Reserved
CLD[17]	**BLUE1**	Reserved	Reserved	Reserved
CLD[16]	**BLUE0**	Reserved	Reserved	Reserved
CLD[15]	**GREEN7**	**GREEN4**	**GREEN5**	**GREEN3**
CLD[14]	**GREEN6**	**GREEN3**	**GREEN4**	**GREEN2**
CLD[13]	**GREEN5**	**GREEN2**	**GREEN3**	**GREEN1**
CLD[12]	**GREEN4**	**GREEN1**	**GREEN2**	**GREEN0**
CLD[11]	**GREEN3**	**GREEN0**	**GREEN1**	Reserved
CLD[10]	**GREEN2**	Intensity bit	**GREEN0**	Reserved
CLD[9]	**GREEN1**	Reserved	Reserved	Reserved
CLD[8]	**GREEN0**	Reserved	Reserved	Reserved
CLD[7]	**RED7**	**RED4**	**RED4**	**RED3**
CLD[6]	**RED6**	**RED3**	**RED3**	**RED2**
CLD[5]	**RED5**	**RED2**	**RED2**	**RED1**
CLD[4]	**RED4**	**RED1**	**RED1**	**RED0**
CLD[3]	**RED3**	**RED0**	**RED0**	Reserved
CLD[2]	**RED2**	Intensity bit	Reserved	Reserved
CLD[1]	**RED1**	Reserved	Reserved	Reserved
CLD[0]	**RED0**	Reserved	Reserved	Reserved

Glossary

This glossary describes some of the terms used in technical documents from ARM.

Advanced High-performance Bus (AHB)
A bus protocol with a fixed pipeline between address/control and data phases. It only supports a subset of the functionality provided by the AMBA AXI protocol. The full AMBA AHB protocol specification includes a number of features that are not commonly required for master and slave IP developments and ARM recommends only a subset of the protocol is usually used. This subset is defined as the AMBA AHB-Lite protocol.

See also Advanced Microcontroller Bus Architecture and AHB-Lite.

Advanced Microcontroller Bus Architecture (AMBA)
A family of protocol specifications that describe a strategy for the interconnect. AMBA is the ARM open standard for on-chip buses. It is an on-chip bus specification that details a strategy for the interconnection and management of functional blocks that make up a *System-on-Chip* (SoC). It aids in the development of embedded processors with one or more CPUs or signal processors and multiple peripherals. AMBA complements a reusable design methodology by defining a common backbone for SoC modules.

Advanced Peripheral Bus (APB)
A simpler bus protocol than AXI and AHB. It is designed for use with ancillary or general-purpose peripherals such as timers, interrupt controllers, UARTs, and I/O ports. Connection to the main system bus is through a system-to-peripheral bus bridge that helps to reduce system power consumption.

AHB *See* Advanced High-performance Bus.

AHB-Lite A subset of the full AMBA AHB protocol specification. It provides all of the basic functions required by the majority of AMBA AHB slave and master designs, particularly when used with a multi-layer AMBA interconnect. In most cases, the extra facilities provided by a full AMBA AHB interface are implemented more efficiently by using an AMBA AXI protocol interface.

Aligned A data item stored at an address that is divisible by the number of bytes that defines the data size is said to be aligned. Aligned words and halfwords have addresses that are divisible by four and two respectively. The terms word-aligned and halfword-aligned therefore stipulate addresses that are divisible by four and two respectively.

AMBA *See* Advanced Microcontroller Bus Architecture.

APB *See* Advanced Peripheral Bus.

Application Specific Integrated Circuit (ASIC)
An integrated circuit that has been designed to perform a specific application function. It can be custom-built or mass-produced.

ASIC *See* Application Specific Integrated Circuit.

ATPG *See* Automatic Test Pattern Generation.

Automatic Test Pattern Generation (ATPG)
The process of automatically generating manufacturing test vectors for an ASIC design, using a specialized software tool.

Beat Alternative word for an individual transfer within a burst. For example, an INCR4 burst comprises four beats.

See also Burst.

Big-endian Byte ordering scheme in which bytes of decreasing significance in a data word are stored at increasing addresses in memory.

See also Little-endian and Endianness.

Burst A group of transfers to consecutive addresses. Because the addresses are consecutive, there is no requirement to supply an address for any of the transfers after the first one. This increases the speed at which the group of transfers can occur. Bursts over AMBA are controlled using signals to indicate the length of the burst and how the addresses are incremented.

See also Beat.

Byte An 8-bit data item.

Direct Memory Access (DMA)
An operation that accesses main memory directly, without the processor performing any accesses to the data concerned.

DMA *See* Direct Memory Access.

Doubleword A 64-bit data item. The contents are taken as being an unsigned integer unless otherwise stated.

Doubleword-aligned A data item having a memory address that is divisible by eight.

Endianness Byte ordering. The scheme that determines the order that successive bytes of a data word are stored in memory. An aspect of the system's memory mapping.

See also Little-endian and Big-endian

Little-endian Byte ordering scheme in which bytes of increasing significance in a data word are stored at increasing addresses in memory.

See also Big-endian and Endianness.

Multi-layer An interconnect scheme similar to a cross-bar switch. Each master on the interconnect has a direct link to each slave, The link is not shared with other masters. This enables each master to process transfers in parallel with other masters. Contention only occurs in a multi-layer interconnect at a payload destination, typically the slave.

Multi-master AHB Typically a shared, not multi-layer, AHB interconnect scheme. More than one master connects to a single AMBA AHB link. In this case, the bus is implemented with a set of full AMBA AHB master interfaces. Masters that use the AMBA AHB-Lite protocol must connect through a wrapper to supply full AMBA AHB master signals to support multi-master operation.

Processor A processor is the circuitry in a computer system required to process data using the computer instructions. It is an abbreviation of microprocessor. A clock source, power supplies, and main memory are also required to create a minimum complete working computer system.

Reserved A field in a control register or instruction format is reserved if the field is to be defined by the implementation, or produces Unpredictable results if the contents of the field are not zero. These fields are reserved for use in future extensions of the architecture or are implementation-specific. All reserved bits not used by the implementation must be written as 0 and read as 0.

Word A 32-bit data item.

Word-invariant In a word-invariant system, the address of each byte of memory changes when switching between little-endian and big-endian operation, in such a way that the byte with address A in one endianness has address A EOR 3 in the other endianness. As a result, each aligned word of memory always consists of the same four bytes of memory in the same order, regardless of endianness. The change of endianness occurs because of the change to the byte addresses, not because the bytes are rearranged.