Shenzhen Haoxinsheng Technology Co., Ltd.

haoxinshengic.com

DATASHEET

Whatsapp: +8615816789795

Skype: 8615816789795

WeChat: 15816789795

Viber: +8615816789795

E-mail: info@haoxinshengic.com

CIRRUS LOGIC®

EP93XX

ARM®9 Embedded Processor Family

EP93xx

User's Guide

©Copyright 2007 Cirrus Logic, Inc.

SEP 2007
DS785UM1

Contacting Cirrus Logic Support

For all product questions and inquiries contact a Cirrus Logic Sales Representative.

To find the one nearest to you go to **www.cirrus.com**

Cirrus Logic, Inc. and its subsidiaries ("Cirrus") believe that the information contained in this document is accurate and reliable. However, the information is subject to change without notice and is provided "AS IS" without warranty of any kind (express or implied). Customers are advised to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability. No responsibility is assumed by Cirrus for the use of this information, including use of this information as the basis for manufacture or sale of any items, or for infringement of patents or other rights of third parties. This document is the property of Cirrus and by furnishing this information, Cirrus grants no license, express or implied under any patents, mask work rights, copyrights, trademarks, trade secrets or other intellectual property rights. Cirrus owns the copyrights associated with the information contained herein and gives consent for copies to be made of the information only for use within your organization with respect to Cirrus integrated circuits or other products of Cirrus. This consent does not extend to other copying such as copying for general distribution, advertising or promotional purposes, or for creating any work for resale.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). CIRRUS PRODUCTS ARE NOT DESIGNED, AUTHORIZED OR WARRANTED FOR USE IN AIRCRAFT SYSTEMS, MILITARY APPLICATIONS, PRODUCTS SURGICALLY IMPLANTED INTO THE BODY, LIFE SUPPORT PRODUCTS OR OTHER CRITICAL APPLICATIONS (INCLUDING MEDICAL DEVICES, AIRCRAFT SYSTEMS OR COMPONENTS AND PERSONAL OR AUTOMOTIVE SAFETY OR SECURITY DEVICES). INCLUSION OF CIRRUS PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK AND CIRRUS DISCLAIMS AND MAKES NO WARRANTY, EXPRESS, STATUTORY OR IMPLIED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE, WITH REGARD TO ANY CIRRUS PRODUCT THAT IS USED IN SUCH A MANNER. IF THE CUSTOMER OR CUSTOMER'S CUSTOMER USES OR PERMITS THE USE OF CIRRUS PRODUCTS IN CRITICAL APPLICATIONS, CUSTOMER AGREES, BY SUCH USE, TO FULLY INDEMNIFY CIRRUS, ITS OFFICERS, DIRECTORS, EMPLOYEES, DISTRIBUTORS AND OTHER AGENTS FROM ANY AND ALL LIABILITY, INCLUDING ATTORNEYS' FEES AND COSTS, THAT MAY RESULT FROM OR ARISE IN CONNECTION WITH THESE USES.

Cirrus Logic, Cirrus, MaverickCrunch, MaverickKey, and the Cirrus Logic logo designs are trademarks of Cirrus Logic, Inc. All other brand and product names in this document may be trademarks or service marks of their respective owners.

Microsoft, Windows, and Windows CE are registered trademarks of Microsoft Corporation.

Microwire is a trademark of National Semiconductor Corp. National Semiconductor is a registered trademark of National Semiconductor Corp.

Texas Instruments is a registered trademark of Texas Instruments, Inc.

Motorola is a registered trademark of Motorola, Inc.

LINUX is a registered trademark of Linus Torvalds.

ARM and Thumb are registered trademarks of ARM Limited

Intel is a registered trademark of Intel Corporation

Hewlett-Packard is a registered trademark of Hewlett-Packard Corporation.

Compaq is a registered trademark of BV, a private Limited Liability Company in the Netherlands.

Contents

Chapter Figures xiv
Chapter Tables xvii
Revision History xxii

Preface P-1
P.1 About the EP93xx User's Guide P-1
P.2 Related Documents from Cirrus Logic P-3
P.3 Reference Documents P-3
P.4 Notational Conventions P-3
P.5 Register Example P-4

Chapter 1. Introduction 1-1
1.1 Introduction 1-1
1.2 EP93xx Features 1-1
1.3 EP93xx Processor Applications 1-7
1.4 EP93xx Processor Highlights 1-7
1.4.1 High-Performance ARM920T Core 1-7
1.4.2 MaverickCrunch™ Co-processor for Ultra-Fast Math Processing 1-7
1.4.3 MaverickKey™ Unique ID Secures Digital Content in OEM Designs 1-8
1.4.4 Integrated Multi-Port USB 2.0 Full Speed Hosts with Transceivers 1-8
1.4.5 Integrated Ethernet MAC Reduces BOM Costs 1-9
1.4.6 8x8 Keypad Interface Reduces BOM Costs 1-9
1.4.7 Multiple Booting Mechanisms Increase Flexibility 1-9
1.4.8 Abundant General Purpose I/Os Build Flexible Systems 1-9
1.4.9 General-Purpose Memory Interface (SDRAM, SRAM, ROM, FLASH) 1-9
1.4.10 12-Bit Analog-to-Digital Converter (ADC) Provides an Integrated Touch-Screen Interface or General ADC Functionality 1-10
1.4.11 Raster Analog / LCD Controller 1-10
1.4.12 Graphics Accelerator 1-10
1.4.13 PCMCIA Interface 1-10

Chapter 2. ARM920T Core and Advanced High-Speed Bus (AHB) 2-1
2.1 Introduction 2-1
2.2 Overview: ARM920T Core 2-1
2.2.1 Features 2-1
2.2.2 Block Diagram 2-2
2.2.3 Operations 2-2
2.2.3.1 ARM9TDMI Core 2-3
2.2.3.2 Memory Management Unit 2-4
2.2.3.3 Cache and Write Buffer 2-5
2.2.4 Co-processor Interface 2-6
2.2.5 AMBA AHB Bus Interface Overview 2-6
2.2.6 AHB Implementation Details 2-7
2.2.7 Memory and Bus Access Errors 2-9
2.2.8 Bus Arbitration 2-9
2.2.8.1 Main AHB Bus Arbiter 2-10
2.2.8.2 SDRAM Slave Arbiter 2-11
2.2.8.3 EBI Bus Arbiter 2-11
2.3 AHB Decoder 2-11
2.3.1 AHB Slave 2-11

©Copyright 2007 Cirrus Logic, Inc.

2.3.2 AHB-to-APB Bridge 2-12
2.3.2.1 Function and Operation of the AHB-to-APB Bridge 2-12
2.3.3 APB Slave 2-13
2.3.4 Register Definitions 2-13
2.3.5 Memory Map 2-16
2.3.6 Internal Register Map 2-17
2.3.6.1 Memory Access Rules 2-17

Chapter 3. MaverickCrunch Co-Processor 3-1
3.1 Introduction 3-1
3.1.1 Features 3-1
3.1.2 Operational Overview 3-1
3.1.3 Pipelines and Latency 3-3
3.1.4 Data Registers 3-3
3.1.5 Integer Saturation Arithmetic 3-4
3.1.6 Comparisons 3-6
3.2 Programming Examples 3-8
3.2.1 Example 1 3-8
3.2.1.1 Setup Code 3-8
3.2.1.2 C Code 3-8
3.2.1.3 Accessing MaverickCrunch with ARM Co-Processor Instructions 3-8
3.2.1.4 MaverickCrunch Assembly Language Instructions 3-8
3.2.2 Example 2 3-9
3.2.2.1 C Code 3-9
3.2.2.2 MaverickCrunch Assembly Language Instructions 3-9
3.3 DSPSC Register 3-10
3.4 ARM Co-Processor Instruction Format 3-14
3.5 Instruction Set for the MaverickCrunch Co-Processor 3-17
3.5.1 Load and Store Instructions 3-21
3.5.2 Move Instructions 3-24
3.5.3 Accumulator and DSPSC Move Instructions 3-27
3.5.4 Copy and Conversion Instructions 3-31
3.5.5 Shift Instructions 3-35
3.5.6 Compare Instructions 3-36
3.5.7 Floating Point Arithmetic Instructions 3-38
3.5.8 Integer Arithmetic Instructions 3-41
3.5.9 Accumulator Arithmetic Instructions 3-45

Chapter 4. Boot ROM 4-1
4.1 Introduction 4-1
4.1.1 Boot ROM Hardware Operational Overview 4-1
4.1.1.1 Memory Map 4-1
4.1.2 Boot ROM Software Operational Overview 4-1
4.1.2.1 Image Header 4-2
4.1.2.2 Boot Algorithm 4-2
4.1.2.3 Flowchart 4-3
4.2 Boot Options 4-4
4.2.1 UART Boot 4-6
4.2.2 SPI Boot 4-6
4.2.3 FLASH Boot 4-6
4.2.4 SDRAM or SyncFLASH Boot 4-7

©Copyright 2007 Cirrus Logic, Inc.

4.2.5 Synchronous Memory Operation....4-7

Chapter 5. System Controller....5-1
5.1 Introduction....5-1
5.1.1 System Startup....5-1
5.1.2 System Reset....5-1
5.1.3 Hardware Configuration Control....5-2
5.1.4 Software System Configuration Options....5-4
5.1.5 Clock Control....5-4
5.1.5.1 Oscillators and Programmable PLLs....5-4
5.1.5.2 Bus and Peripheral Clock Generation....5-5
5.1.5.3 Steps for Clock Configuration....5-9
5.1.6 Power Management....5-9
5.1.6.1 Clock Gatings....5-9
5.1.6.2 System Power States....5-10
5.1.7 Interrupt Generation....5-12
5.2 Registers....5-13

Chapter 6. Vectored Interrupt Controller....6-1
6.1 Introduction....6-1
6.1.1 Interrupt Priority....6-2
6.1.2 Interrupt Configuration....6-3
6.1.3 Interrupt Details....6-4
6.2 Registers....6-8

Chapter 7. Raster Engine With Analog/LCD Integrated Timing and Interface....7-1
7.1 Introduction....7-1
7.2 Features....7-3
7.3 Raster Engine Features Overview....7-3
7.3.1 Hardware Blinking....7-3
7.3.2 Color Look-Up Tables....7-4
7.3.3 Grayscale/Color Generation for Monochrome/Passive Low Color Displays....7-4
7.3.4 Frame Buffer Organization....7-4
7.3.5 Frame Buffer Memory Size....7-6
7.3.6 Pulse Width Modulated Brightness....7-6
7.3.7 Hardware Cursor....7-7
7.4 Functional Details....7-7
7.4.1 VILOSATI (Video Image Line Output Scanner and Transfer Interface)....7-8
7.4.2 Video FIFO....7-9
7.4.3 Video Pixel MUX....7-10
7.4.4 Blink Function....7-10
7.4.5 Color Look-Up-Tables....7-11
7.4.6 Color RGB Mux....7-11
7.4.7 Pixel Shift Logic....7-12
7.4.8 Grayscale/Color Generator for Monochrome/Passive Low Color Displays....7-15
7.4.8.1 HORZ_CNT3, HORZ_CNT4 Counters....7-16
7.4.8.2 VERT_CNT3, VERT_CNT4 Counters....7-16
7.4.8.3 FRAME_CNT3, FRAME_CNT4 Counters....7-16
7.4.8.4 HORZ_CNTx (pixel) timing....7-16
7.4.8.5 VERT_CNTx (line) timing....7-16

©Copyright 2007 Cirrus Logic, Inc.

7.4.8.6 FRAME_CNTx timing 7-16
7.4.8.7 Grayscale Look-Up Table (GrySclLUT) 7-17
7.4.8.8 GrySclLUT Timing Diagram 7-18
7.4.9 Hardware Cursor 7-24
7.4.9.1 Registers Used for Cursor 7-26
7.4.10 Video Timing 7-28
7.4.10.1 Setting the Video Memory Parameters 7-31
7.4.10.2 PixelMode 7-32
7.4.11 Blink Logic 7-32
7.4.11.1 BlinkRate 7-32
7.4.11.2 Defining Blink Pixels 7-32
7.4.11.3 Types of Blinking 7-33
7.4.12 Color Mode Definition 7-35
7.4.12.1 Pixel Look-up Table Mode 7-35
7.4.12.2 Triple 8-bit Color Definition Mode 7-35
7.4.12.3 16-bit 565 Color Definition Mode 7-35
7.4.12.4 16-bit 555 Color Definition Mode 7-35
7.5 Registers 7-36

Chapter 8. Graphics Accelerator 8-1
8.1 Overview 8-1
8.2 Block Processing Modes 8-1
8.2.1 Copy 8-2
8.2.1.1 Transparency 8-2
8.2.1.2 Logical Mask 8-2
8.2.1.3 Logical Destination 8-2
8.2.1.4 Operation Precedence 8-2
8.2.2 Remapping 8-3
8.2.3 Block Fills 8-3
8.2.4 Packed Memory Transfer 8-3
8.3 Line Draws 8-3
8.3.1 Breshenham Line Draws 8-4
8.3.2 Pixel Step Line Draws 8-4
8.4 Memory Organization for Graphics Accelerator 8-4
8.4.1 Memory Organization for 1 Bit Per Pixel (bpp) 8-5
8.4.2 Memory Organization for 4-Bits Per Pixel 8-5
8.4.3 Memory Organization for 8-Bits Per Pixel 8-5
8.4.4 Memory Organization for 16-Bits Per Pixel 8-6
8.4.5 Memory Organization for 24-Bits Per Pixel 8-7
8.4.6 Memory Map Access 8-8
8.5 Register Programming 8-8
8.5.1 Word Count 8-8
8.5.1.1 Example: 8 BPP mode 8-8
8.5.1.2 Example: 24 BPP (packed) mode 8-9
8.5.2 Pixel End and Start 8-9
8.5.2.1 4 BPP Word Layout 8-10
8.5.2.2 8 BPP Word Layout 8-11
8.5.2.3 16 BPP WORD Layout 8-11
8.5.2.4 24 BPP mode 8-12
8.6 Register Usage 8-13
8.6.1 Breshenham's Algorithm Line Draw 8-13
8.6.2 Example of Breshenham's Algorithm Line Draw 8-15
8.6.3 Block Fill Function 8-16

©Copyright 2007 Cirrus Logic, Inc.

8.6.4 Block Copy Function....8-18
8.6.4.1 Example of Block Copy....8-21
8.7 Registers....8-22

Chapter 9. 1/10/100 Mbps Ethernet LAN Controller....9-1
9.1 Introduction....9-1
9.1.1 Detailed Description....9-1
9.1.1.1 Host Interface and Descriptor Processor....9-1
9.1.1.2 Reset and Initialization....9-2
9.1.1.3 Power-down Modes....9-2
9.1.1.4 Address Space....9-2
9.1.2 MAC Engine....9-3
9.1.2.1 Data Encapsulation....9-3
9.1.3 Packet Transmission Process....9-5
9.1.3.1 Carrier Deference....9-5
9.1.4 Transmit Back-Off....9-7
9.1.4.1 Transmission....9-7
9.1.4.2 The FCS Field....9-7
9.1.4.3 Bit Order....9-8
9.1.4.4 Destination Address (DA) Filter....9-8
9.1.4.5 Perfect Address Filtering....9-8
9.1.4.6 Hash Filter....9-9
9.1.4.7 Flow Control....9-10
9.1.4.8 Receive Flow Control....9-10
9.1.4.9 Transmit Flow Control....9-10
9.1.4.10 Rx Missed and Tx Collision Counters....9-11
9.1.4.11 Accessing the MII....9-11
9.2 Descriptor Processor....9-13
9.2.1 Receive Descriptor Processor Queues....9-13
9.2.2 Receive Descriptor Queue....9-13
9.2.3 Receive Status Queue....9-16
9.2.3.1 Receive Status Format....9-18
9.2.3.2 Receive Flow....9-21
9.2.3.3 Receive Errors....9-22
9.2.3.4 Receive Descriptor Data/Status Flow....9-23
9.2.3.5 Receive Descriptor Example....9-24
9.2.3.6 Receive Frame Pre-Processing....9-25
9.2.3.7 Transmit Descriptor Processor Queues....9-26
9.2.3.8 Transmit Descriptor Queue....9-26
9.2.3.9 Transmit Descriptor Format....9-28
9.2.3.10 Transmit Status Queue....9-30
9.2.3.11 Transmit Status Format....9-32
9.2.3.12 Transmit Flow....9-34
9.2.3.13 Transmit Errors....9-35
9.2.3.14 Transmit Descriptor Data/Status Flow....9-36
9.2.4 Interrupts....9-37
9.2.4.1 Interrupt Processing....9-37
9.2.5 Initialization....9-37
9.2.5.1 Interrupt Processing....9-38
9.2.5.2 Receive Queue Processing....9-38
9.2.5.3 Transmit Queue Processing....9-38
9.2.5.4 Other Processing....9-38
9.2.5.5 Transmit Restart Process....9-39
9.3 Registers....9-40

©Copyright 2007 Cirrus Logic, Inc.

Chapter 10. DMA Controller 10-1
10.1 Introduction 10-1
10.1.1 DMA Features List 10-1
10.1.2 Managing Data Transfers Using a DMA Channel 10-2
10.1.3 DMA Operations 10-3
10.1.3.1 Memory-to-Memory Channels 10-3
10.1.3.2 Memory-to-Peripheral Channels 10-4
10.1.4 Internal M2P or P2M AHB Master Interface Functional Description 10-4
10.1.5 M2M AHB Master Interface Functional Description 10-5
10.1.5.1 Software Trigger Mode 10-5
10.1.5.2 Hardware Trigger Mode for Internal Peripherals (SSP and IDE) and for External Peripherals without Handshaking Signals 10-6
10.1.5.3 Hardware Trigger Mode for External Peripherals with Handshaking Signals 10-6
10.1.6 AHB Slave Interface Limitations 10-6
10.1.7 Interrupt Interface 10-6
10.1.8 Internal M2P/P2M Data Unpacker/Packer Functional Description 10-6
10.1.9 Internal M2P/P2M DMA Functional Description 10-7
10.1.9.1 Internal M2P/P2M DMA Buffer Control Finite State Machine 10-7
10.1.9.2 Data Transfer Initiation and Termination 10-9
10.1.10 M2M DMA Functional Description 10-10
10.1.10.1 M2M DMA Control Finite State Machine 10-10
10.1.10.2 M2M Buffer Control Finite State Machine 10-12
10.1.10.3 Data Transfer Initiation 10-13
10.1.10.4 Data Transfer Termination 10-15
10.1.10.5 Memory Block Transfer 10-16
10.1.10.6 Bandwidth Control 10-16
10.1.10.7 External DMA Request (DREQ) Mode 10-16
10.1.11 DMA Data Transfer Size Determination 10-17
10.1.11.1 Software Initiated M2M and M2P/P2M Transfers 10-17
10.1.11.2 Hardware-Initiated M2M Transfers 10-18
10.1.12 Buffer Descriptors 10-18
10.1.12.1 Internal M2P/P2M Channel Rx Buffer Descriptors 10-19
10.1.12.2 Internal M2P/P2M Channel Tx Buffer Descriptors 10-19
10.1.12.3 M2M Channel Buffer Descriptors 10-19
10.1.13 Bus Arbitration 10-19
10.2 Registers 10-20
10.2.1 DMA Controller Memory Map 10-20
10.2.2 Internal M2P/P2M Channel Register Map 10-21

Chapter 11. Universal Serial Bus Host Controller 11-1
11.1 Introduction 11-1
11.1.1 Features 11-1
11.2 Overview 11-1
11.2.1 Data Transfer Types 11-2
11.2.2 Host Controller Interface 11-3
11.2.2.1 Communication Channels 11-3
11.2.2.2 Data Structures 11-4
11.2.3 Host Controller Driver Responsibilities 11-6
11.2.3.1 Host Controller Management 11-6
11.2.3.2 Bandwidth Allocation 11-6
11.2.3.3 List Management 11-7
11.2.3.4 Root Hub 11-7

©Copyright 2007 Cirrus Logic, Inc.

11.2.4 Host Controller Responsibilities....11-8
11.2.4.1 USB States....11-8
11.2.4.2 Frame Management....11-8
11.2.4.3 List Processing....11-8
11.2.5 USB Host Controller Blocks....11-9
11.2.5.1 AHB Slave....11-9
11.2.5.2 AHB Master....11-9
11.2.5.3 HCI Slave Block....11-9
11.2.5.4 HCI Master Block....11-10
11.2.5.5 USB State Control....11-10
11.2.5.6 Data FIFO....11-10
11.2.5.7 List Processor....11-10
11.2.5.8 Root Hub and Host SIE....11-10
11.3 Registers....11-11

Chapter 12. Static Memory Controller....12-1
12.1 Introduction....12-1
12.2 Static Memory Controller Operation....12-2
12.3 PCMCIA Interface (EP9315 Processor Only)....12-5
12.4 PC Card Memory-Mode Enable Signals....12-8
12.5 PC Card Memory Mapping....12-8
12.6 Registers....12-10
12.6.1 Bank Configuration Registers....12-10
12.6.2 PCMCIA Configuration Registers (EP9315 Processor Only)....12-13

Chapter 13. SDRAM, SyncROM, and SyncFLASH Controller....13-1
13.1 Introduction....13-1
13.2 Booting from SyncROM or SyncFLASH....13-1
13.3 Address Pin Usage....13-3
13.4 SDRAM Initialization....13-4
13.5 Programming Mode Register: SDRAM Or SyncROM Device....13-6
13.6 SDRAM Self Refresh....13-8
13.6.1 Entering Self Refresh Mode....13-8
13.6.2 Exiting Self Refresh Mode....13-8
13.7 Programming Registers: SyncFLASH Device....13-8
13.8 External Synchronous Memory System....13-9
13.8.1 Chip Select SDCSN[3:0] Decoding....13-9
13.8.2 Address/Data/Control Required by Memory System....13-10
13.9 Registers....13-17

Chapter 14. UART1 With HDLC and Modem Control Signals....14-1
14.1 Introduction....14-1
14.2 UART Overview....14-1
14.2.1 UART Functional Description....14-2
14.2.1.1 AMBA APB Interface....14-2
14.2.1.2 DMA Block....14-2
14.2.1.3 Register Block....14-2
14.2.1.4 Baud Rate Generator....14-4
14.2.1.5 Transmit FIFO....14-4
14.2.1.6 Receive FIFO....14-4
14.2.1.7 Transmit Logic....14-4
14.2.1.8 Receive Logic....14-4

©Copyright 2007 Cirrus Logic, Inc.

14.2.1.9 Interrupt Generation Logic 14-4
14.2.1.10 Synchronizing Registers and Logic 14-5
14.2.2 UART Operation 14-5
14.2.2.1 Error Bits 14-6
14.2.2.2 Disabling the FIFOs 14-6
14.2.2.3 System/diagnostic Loopback Testing 14-6
14.2.2.4 UART Character Frame 14-6
14.2.3 Interrupts 14-7
14.2.3.1 UARTMSINTR 14-7
14.2.3.2 UARTRXINTR 14-7
14.2.3.3 UARTTXINTR 14-7
14.2.3.4 UARTRTINTR 14-8
14.2.3.5 UARTINTR 14-8
14.3 Modem 14-8
14.4 HDLC 14-8
14.4.1 Overview of HDLC Modes 14-9
14.4.2 Selecting HDLC Modes 14-9
14.4.3 HDLC Transmit 14-11
14.4.4 HDLC Receive 14-11
14.4.5 CRCs 14-12
14.4.6 Address Matching 14-12
14.4.7 Aborts 14-13
14.4.8 DMA 14-14
14.4.9 Writing Configuration Registers 14-14
14.5 UART1 Package Dependency 14-14
14.5.1 Clocking Requirements 14-15
14.5.2 Bus Bandwidth Requirements 14-16
14.1 Registers 14-17

Chapter 15. UART2 15-1
15.1 Introduction 15-1
15.2 IrDA SIR Block 15-1
15.2.1 IrDA SIR Encoder/decoder Functional Description 15-1
15.2.1.1 IrDA SIR Transmit Encoder 15-2
15.2.1.2 IrDA SIR Receive Decoder 15-2
15.2.2 IrDA SIR Operation 15-3
15.2.2.1 System/diagnostic Loopback Testing 15-4
15.2.3 IrDA Data Modulation 15-4
15.2.4 Enabling Infrared (Ir) Modes 15-5
15.3 UART2 Package Dependency 15-5
15.3.1 Clocking Requirements 15-5
15.3.2 Bus Bandwidth Requirements 15-6
15.4 Registers 15-7

Chapter 16. UART3 With HDLC Encoder 16-1
16.1 Introduction 16-1
16.2 Implementation Details 16-1
16.2.1 UART3 Package Dependency 16-1
16.2.2 Clocking Requirements 16-2
16.2.3 Bus Bandwidth Requirements 16-2
16.3 Registers 16-3

 ©Copyright 2007 Cirrus Logic, Inc.

Chapter 17. IrDA 17-1
17.1 Introduction 17-1
17.2 IrDA Interfaces 17-1
17.3 Shared IrDA Interface Feature 17-2
17.3.1 Overview 17-2
17.3.2 Functional Description 17-2
17.3.2.1 General Configuration 17-3
17.3.2.2 Transmitting Data 17-3
17.3.2.3 Receiving Data 17-5
17.3.2.4 Special Conditions 17-7
17.3.3 Control Information Buffering 17-8
17.4 Medium IrDA Specific Features 17-8
17.4.1 Introduction 17-8
17.4.1.1 Bit Encoding 17-8
17.4.1.2 Frame Format 17-9
17.4.2 Functional Description 17-11
17.4.2.1 Baud Rate Generation 17-11
17.4.2.2 Receive Operation 17-11
17.4.2.3 Transmit Operation 17-13
17.5 Fast IrDA Specific Features 17-13
17.5.1 Introduction 17-14
17.5.1.1 4PPM Modulation 17-14
17.5.1.2 4.0 Mbps FIR Frame Format 17-15
17.5.2 Functional Description 17-17
17.5.2.1 Baud Rate Generation 17-17
17.5.2.2 Receive Operation 17-18
17.5.2.3 Transmit Operation 17-19
17.5.3 IrDA Connectivity 17-20
17.5.4 IrDA Integration Information 17-21
17.5.4.1 Enabling Infrared Modes 17-21
17.5.4.2 Clocking Requirements 17-21
17.5.4.3 Bus Bandwidth Requirements 17-22
17.6 Registers 17-23

Chapter 18. Timers 18-1
18.1 Introduction 18-1
18.1.1 Features 18-1
18.1.2 16 and 32-bit Timer Operation 18-1
18.1.2.1 Free Running Mode 18-2
18.1.2.2 Pre-load Mode 18-2
18.1.3 40-bit Timer Operation 18-2
18.2 Registers 18-2

Chapter 19. Watchdog Timer 19-1
19.1 Introduction 19-1
19.1.1 Watchdog Activation 19-2
19.1.2 Clocking Requirements 19-2
19.1.3 Reset Requirements 19-2
19.1.4 Watchdog Status 19-2
19.1 Registers 19-3

©Copyright 2007 Cirrus Logic, Inc.

Chapter 20. Real Time Clock With Software Trim 20-1
20.1 Introduction 20-1
20.1.1 Software Trim 20-1
20.1.1.1 Software Compensation 20-2
20.1.1.2 Oscillator Frequency Calibration 20-2
20.1.1.3 RTCSWComp Value Determination 20-2
20.1.1.4 Example - Measured Value Split Into Integer and Fractional Component 20-3
20.1.1.5 Maximum Error Calculation vs. Real Time Clock Accuracy 20-3
20.1.1.6 Real-Time Interrupt 20-3
20.1.2 Reset Control 20-4
20.1 Registers 20-4

Chapter 21. I²S Controller 21-1
21.1 Introduction 21-1
21.2 I²S Transmitter Channel Overview 21-2
21.3 I²S Receiver Channel Overview 21-5
21.3.1 Receiver FIFO's 21-6
21.4 I²S Master Clock Generation 21-7
21.5 I²S Bit Clock Rate Generation 21-9
21.5.1 Example of the Bit Clock Generation 21-9
21.5.2 Example of Right Justified LRCK format 21-10
21.6 Interrupts 21-10
21.7 Registers 21-12
21.7.1 I²S TX Registers 21-12
21.7.2 I²S RX Registers 21-19
21.7.3 I²S Configuration and Status Registers 21-25
21.7.4 I²S Global Status Registers 21-29

Chapter 22. AC'97 Controller 22-1
22.1 Introduction 22-1
22.2 Interrupts 22-3
22.2.1 Channel Interrupts 22-3
22.2.1.1 RIS 22-3
22.2.1.2 TIS 22-3
22.2.1.3 RTIS 22-4
22.2.1.4 TCIS 22-4
22.2.2 Global Interrupts 22-4
22.2.2.1 CODECREADY 22-4
22.2.2.2 WINT 22-4
22.2.2.3 GPIOINT 22-4
22.2.2.4 GPIOTXCOMPLETE 22-5
22.2.2.5 SLOT2INT 22-5
22.2.2.6 SLOT1TXCOMPLETE 22-5
22.2.2.7 SLOT2TXCOMPLETE 22-5
22.3 System Loopback Testing 22-5
22.4 Registers 22-5

Chapter 23. Synchronous Serial Port 23-1
23.1 Introduction 23-1
23.2 Features 23-1
23.3 SSP Functionality 23-2
23.4 SSP Pin Multiplex 23-2

©Copyright 2007 Cirrus Logic, Inc.

23.5 Configuring the SSP 23-2
23.5.1 Enabling SSP Operation 23-2
23.5.2 Master/Slave Mode 23-3
23.5.3 Serial Bit Rate Generation 23-3
23.5.4 Frame Format 23-3
23.5.5 Texas Instruments® Synchronous Serial Frame Format 23-4
23.5.6 Motorola® SPI Frame Format 23-5
23.5.6.1 SPO Clock Polarity 23-5
23.5.6.2 SPH Clock Phase 23-5
23.5.7 Motorola SPI Format with SPO=0, SPH=0 23-5
23.5.8 Motorola SPI Format with SPO=0, SPH=1 23-7
23.5.9 Motorola SPI Format with SPO=1, SPH=0 23-8
23.5.10 Motorola SPI Format with SPO=1, SPH=1 23-9
23.5.11 National Semiconductor® Microwire™ Frame Format 23-10
23.5.11.1 Setup and Hold Time Requirements on SFRMIN with Respect to SCLKIN in Microwire Mode 23-12
23.6 Registers 23-13

Chapter 24. Pulse Width Modulator 24-1
24.1 Introduction 24-1
24.2 Theory of Operation 24-1
24.2.1 PWM Programming Examples 24-2
24.2.1.1 Example 24-2
24.2.1.2 Static Programming (PWM is Not Running) Example 24-2
24.2.1.3 Dynamic Programming (PWM is Running) Example 24-3
24.2.2 Programming Rules 24-3
24.3 Registers 24-3

Chapter 25. Analog Touch Screen Interface 25-1
25.1 Introduction 25-1
25.2 Touch Screen Controller Operation 25-1
25.2.1 Touch Screen Scanning: Four-wire and Eight-wire Operation 25-4
25.2.2 Five-wire and Seven-wire Operation 25-10
25.2.3 Direct Operation 25-12
25.2.4 Measuring Analog Input with the Touch Screen Controls Disabled 25-13
25.2.5 Measuring Touch Screen Resistance 25-15
25.2.6 Polled and Interrupt-Driven Modes 25-16
25.2.7 Touch Screen Package Dependency 25-16
25.3 Registers 25-17

Chapter 26. Keypad Interface 26-1
26.1 Introduction 26-1
26.2 Theory of Operation 26-2
26.2.1 Apparent Key Detection 26-3
26.2.2 Scan and Debounce 26-5
26.2.3 Interrupt Generation 26-5
26.2.4 Low Power Mode 26-6
26.2.5 Three-key Reset 26-6
26.3 Registers 26-6

©Copyright 2007 Cirrus Logic, Inc.

Chapter 27. IDE Interface 27-1
27.1 Introduction 27-1
27.2 Theory of Operation 27-1
27.2.1 Diagrams and State Machines 27-2
27.2.2 PIO Operations 27-3
27.2.3 MDMA Operations 27-4
27.2.4 UDMA Operations 27-5
27.2.5 Performance Considerations 27-5
27.2.6 UDMA Example 27-6
27.2.7 DMA Request Latency 27-7
27.2.7.1 DMA Request Deassertion 27-7
27.2.7.2 DMA Request Latency Overview 27-7
27.2.7.3 IDE DMA Programming Considerations 27-8
27.2.8 IDE Package Dependency 27-9
27.2.8.1 System Configuration Constraints 27-9
27.2.8.2 Bus Bandwidth Requirements 27-9
27.3 Registers 27-10

Chapter 28. GPIO Interface 28-1
28.1 Introduction 28-1
28.1.1 Memory Map 28-3
28.1.2 Functional Description 28-3
28.1.3 Reset 28-5
28.1.4 GPIO Pin Map 28-6
28.2 Registers 28-9

Chapter 29. Security 29-1
29.1 Introduction 29-1
29.2 Features 29-1
29.3 Contact Information 29-1
29.4 Registers 29-2

Chapter 30. Glossary 30-1

Chapter 31. EP93XX Register List 31-1

Figures

Figure 1-1. EP9301 Block Diagram 1-2
Figure 1-2. EP9302 Block Diagram 1-3
Figure 1-3. EP9307 Block Diagram 1-3
Figure 1-4. EP9312 Block Diagram 1-4
Figure 1-5. EP9315 Block Diagram 1-4
Figure 2-1. ARM920T Block Diagram 2-2
Figure 2-2. Typical AMBA AHB System 2-7
Figure 2-3. Main Data Paths 2-8

©Copyright 2007 Cirrus Logic, Inc.

Figure 4-1. Flow Chart of Boot ROM Software 4-4
Figure 4-2. Flow chart of Boot Sequence for 16-bit SDRAM Devices 4-7
Figure 5-1. Phase Locked Loop (PLL) Structure 5-4
Figure 5-2. Clock Generation System 5-6
Figure 5-3. Bus Clock Generation 5-7
Figure 5-4. Power States and Transitions 5-11
Figure 6-1. Vectored Interrupt Controller Block Diagram 6-2
Figure 7-1. Raster Engine Block Diagram 7-8
Figure 7-2. Video Buffer Diagram 7-9
Figure 7-3. Graphics Matrix for 50% Duty Cycle 7-20
Figure 7-4. Sample Matrix Causing Flickering 7-21
Figure 7-5.. Sample Matrix That Avoids Flickering 7-21
Figure 7-6. Programming for One-third Luminous Intensity 7-22
Figure 7-7. Creating Bit Patterns that Move to the Right 7-23
Figure 7-8. Three and Four Count Axis 7-24
Figure 7-9. Progressive/Dual Scan Video Signals 7-29
Figure 7-10. Interlaced Video Signals 7-30
Figure 9-1. 1/10/100 Mbps Ethernet LAN Controller Block Diagram 9-1
Figure 9-2. Ethernet Frame / Packet Format (Type II only) 9-4
Figure 9-3. Packet Transmission Process 9-5
Figure 9-4. Carrier Deference State Diagram 9-6
Figure 9-5. Data Bit Transmission Order 9-8
Figure 9-6. CRC Logic 9-9
Figure 9-7. Receive Descriptor Format and Data Fragments 9-14
Figure 9-8. Receive Status Queue 9-17
Figure 9-9. Receive Flow Diagram 9-21
Figure 9-10. Receive Descriptor Data/Status Flow 9-23
Figure 9-11. Receive Descriptor Example 9-24
Figure 9-12. Receive Frame Pre-processing 9-25
Figure 9-13. Transmit Descriptor Format and Data Fragments 9-27
Figure 9-14. Multiple Fragments Per Transmit Frame 9-28
Figure 9-15. Transmit Status Queue 9-31
Figure 9-16. Transmit Flow Diagram 9-34
Figure 9-17. Transmit Descriptor Data/Status Flow 9-36
Figure 10-1. DMA M2P/P2M Finite State Machine 10-7
Figure 10-2. M2M DMA Control Finite State Machine 10-10
Figure 10-3. M2M DMA Buffer Finite State Machine 10-12

©Copyright 2007 Cirrus Logic, Inc.

Figure 10-4. Edge-triggered DREQ Mode 10-17
Figure 11-1. USB Focus Areas 11-2
Figure 11-2. Communication Channels 11-3
Figure 11-3. Typical List Structure 11-4
Figure 11-4. Interrupt Endpoint Descriptor Structure 11-5
Figure 11-5. Sample Interrupt Endpoint Schedule 11-6
Figure 11-6. USB Host Controller Block Diagram 11-9
Figure 12-1. 32-bit Read, 32-bit Memory, 0 Wait Cycles, RBLE = 1, WAITn Inactive 12-3
Figure 12-2. 32-bit Write, 32-bit Memory, 0 Wait Cycles, RBLE = 1, WAITn Inactive 12-3
Figure 12-3. 16-bit Read, 16-bit Memory, RBLE = 1, WAITn Active 12-4
Figure 12-4. 16-bit Write, 16-bit Memory, RBLE = 1, WAITn Active 12-4
Figure 12-5. Single PC Card Interface 12-7
Figure 14-1. UART Block Diagram 14-3
Figure 14-2. UART Character Frame 14-6
Figure 14-3. UART Character Frame 14-6
Figure 15-1. IrDA SIR Encoder/decoder Block Diagram 15-2
Figure 15-2. IrDA Data Modulation (3/16) 15-4
Figure 17-1. RZ1/NRZ Bit Encoding Example 17-9
Figure 17-2. 4PPM Modulation Encoding 17-14
Figure 17-3. 4PPM Modulation Example 17-15
Figure 17-4. IrDA (4.0 Mbps) Transmission Format 17-15
Figure 21-1. Architectural Overview of the I^2S Controller 21-1
Figure 21-2. Bit Clock Generation Example 21-10
Figure 21-3. Frame Format for Right Justified Data 21-10
Figure 23-1. Texas Instruments Synchronous Serial Frame Format (Single Transfer) 23-4
Figure 23-2. TI Synchronous Serial Frame Format (Continuous Transfer) 23-4
Figure 23-3. Motorola SPI Frame Format (Single Transfer) with SPO=0 and SPH=0 23-5
Figure 23-4. Motorola SPI Frame Format (Continuous Transfer) with SPO=0 and SPH=0 23-6
Figure 23-5. Motorola SPI Frame Format with SPO=0 and SPH=1 23-7
Figure 23-6. Motorola SPI Frame Format (Single Transfer) with SPO=1 and SPH=0 23-8
Figure 23-7. Motorola SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0 23-8
Figure 23-8. Motorola SPI Frame Format with SPO=1 and SPH=1 23-9
Figure 23-9. Microwire Frame Format (Single Transfer) 23-10
Figure 23-10. Microwire Frame Format (Continuous Transfers) 23-12
Figure 23-11. Microwire Frame Format, SFRMIN Input Setup and Hold Requirements 23-12
Figure 24-1. PWM_INV Example 24-6

 ©Copyright 2007 Cirrus Logic, Inc.

Figure 25-1. Different Types of Touch Screens 25-2
Figure 25-2. 8-Wire Resistive Interface Switching Diagram 25-5
Figure 25-3. 4-Wire Analog Resistive Interface Switching Diagram 25-6
Figure 25-4. Analog Resistive Touch Screen Scan Flow Chart 25-9
Figure 25-5. 5-Wire Analog Resistive Interface Switching Diagram 25-11
Figure 25-6. 5-Wire Feedback (7-Wire) Analog Resistive Interface Switching Diagram 25-12
Figure 25-7. Power Down Detect Press Switching Diagram 25-13
Figure 25-8. Other Switching Diagrams 25-14
Figure 25-9. Measure Resistance Switching Diagram 25-15
Figure 26-1. Key Array Block Diagram 26-1
Figure 26-2. 8 x 8 Key Array Diagram 26-3
Figure 26-3. Apparent Key 00H 26-4
Figure 27-1. IDE Interface Signal Connections 27-2
Figure 28-1. System Level GPIO Connectivity 28-2
Figure 28-2. Signal Connections Within the Standard GPIO Port Control Logic (Ports C, D, E, G, H) 28-4
Figure 28-3. Signal Connections Within the Enhanced GPIO Port Control Logic (Ports A, B, F) 28-5

Tables

Table P-1. Frequency, Package, Applicable EP93xx Processor P-1
Table P-2. Chapter Number and Function, Applicable EP93xx Processor P-1
Table 1-1. EP93xx Maximum Clock Rates, Package Type and Number of Balls 1-1
Table 1-2. EP93xx Features Summary 1-2
Table 2-1. AHB Arbiter Priority Scheme 2-10
Table 2-2. AHB Peripheral Address Range 2-11
Table 2-3. APB Peripheral Address Range 2-12
Table 2-4. ARM920T Core Operating Modes 2-13
Table 2-5. Register Organization Summary 2-14
Table 2-6. CP15 ARM920T Register Description 2-15
Table 2-7. Global Memory Map for the Two Boot Modes 2-16
Table 2-8. Internal Register Map 2-17
Table 3-1. Saturation for Non-accumulator Instructions 3-5
Table 3-2. Accumulator Bit Formats for Saturation 3-5
Table 3-3. Comparison Relationships and Their Results 3-7
Table 3-4. ARM® Condition Codes and Crunch Compare Results 3-7
Table 3-5. Condition Code Definitions 3-15

©Copyright 2007 Cirrus Logic, Inc.

Table 3-6. LDC/STC Opcode Map 3-16
Table 3-7. CDP Opcode Map 3-16
Table 3-8. MCR Opcode Map 3-17
Table 3-9. MRC Opcode Map 3-17
Table 3-10. MaverickCrunch Instruction Set 3-18
Table 3-11. Mnemonic Codes for Loading Floating Point Value from Memory 3-21
Table 3-12. Mnemonic Codes for Loading Integer Value from Memory 3-22
Table 3-13. Mnemonic Codes for Storing Floating Point Values to Memory 3-23
Table 3-14. Mnemonic Codes for Storing Integer Values to Memory 3-23
Table 4-1. Boot Configuration Options 4-5
Table 5-1. Hardware Configuration Control Latched Pins 5-2
Table 5-2. Boot Configuration Options 5-3
Table 5-3. Clock Speeds and Sources 5-8
Table 5-4. Peripherals with PCLK Gating 5-10
Table 5-5. Syscon Register List 5-13
Table 5-6. Priority Order for AHB Arbiter 5-23
Table 5-7. Audio Interfaces Pin Assignment 5-26
Table 6-1. Interrupt Configuration 6-3
Table 6-2. VICx Register Summary 6-8
Table 7-1. Raster Engine Video Mode Output Examples 7-2
Table 7-2. Byte Oriented Frame Buffer Organization 7-5
Table 7-3. Output Pixel Transfer Modes 7-13
Table 7-4. Grayscale Lookup Table (GrySclLUT) 7-17
Table 7-5. Grayscale Timing Diagram 7-18
Table 7-6. Programming Format 7-19
Table 7-7. Programming 50% Duty Cycle Into Lookup Table 7-22
Table 7-8. Programming 33% Duty Cycle into the Lookup Table 7-23
Table 7-9. Programming 33% Duty Cycle into the Lookup Table 7-24
Table 7-10. Cursor Memory Organization 7-25
Table 7-11. Bits P[2:0] in the PixelMode Register 7-32
Table 7-12. Raster Engine Register List 7-36
Table 7-13. Color Mode Definition Table 7-58
Table 7-14. Blink Mode Definition Table 7-58
Table 7-15. Output Shift Mode Table 7-59
Table 7-16. Bits per Pixel Scanned Out 7-59
Table 7-17. Grayscale Look-Up-Table (LUT) 7-75
Table 8-1. Screen Pixels 8-4

©Copyright 2007 Cirrus Logic, Inc.

Table 8-2. bpp Memory Organization 8-5
Table 8-3. 4 bpp Memory Organization 8-5
Table 8-4. 8 bpp Memory Organization 8-6
Table 8-5. 16 bpp Memory Organization 8-6
Table 8-6. 24 bpp Packed Memory Organization (4 pixel/ 3 words) 8-7
Table 8-7. 24 bpp Unpacked Memory Organization (1 pixel/ 1 word) 8-7
Table 8-8. Transfer Example 1 8-8
Table 8-9. Transfer Example 2 8-9
Table 8-10. Transfer Example 3 8-9
Table 8-11. Transfer Example 4 8-9
Table 8-12. Transfer Example 5 8-9
Table 8-13. 4 BPP Memory Layout for Source Image 8-10
Table 8-14. 4 BPP Memory Layout for Destination Image 8-10
Table 8-15. 8 BPP Memory Layout for Source Image 8-11
Table 8-16. 8 BPP Memory Layout for Destination Image 8-11
Table 8-17. 16 BPP Memory Layout for Source Image 8-11
Table 8-18. 16 BPP Memory Layout for Destination Image 8-12
Table 8-19. 24 BPP Memory Layout for Source Image 8-12
Table 8-20. 24 BPP Memory Layout for Destination Image 8-13
Table 8-21. Words Needed for Six 24-Bit Pixels 8-19
Table 8-22. Graphics Accelerator Registers 8-22
Table 8-23. Pixel Mode Encoding 8-30
Table 9-1. FIFO RAM Address Map 9-3
Table 9-2. RXCtl.MA and RXCtl.IAHA[0] Relationships 9-10
Table 9-3. Ethernet Register List 9-40
Table 9-4. Individual Accept, RxFlow Control Enable and Pause Accept Bits 9-42
Table 9-5. Address Filter Pointer 9-52
Table 10-1. Data Transfer Size 10-18
Table 10-2. M2P DMA Bus Arbitration 10-19
Table 10-3. DMA Memory Map 10-20
Table 10-4. Internal M2P/P2M Channel Register Map 10-21
Table 10-5. PPALLOC Register Bits Decode for a Transmit Channel 10-24
Table 10-6. PPALLOC Register Bits Decode for a Receive Channel 10-24
Table 10-7. PPALLOC Register Reset Values 10-24
Table 10-8. PPALLOC Register Reset Values 10-30
Table 10-9. BWC Decode Values 10-33
Table 10-10. DMA Global Interrupt (DMAGIInt) Register 10-45

©Copyright 2007 Cirrus Logic, Inc.

Table 11-1. Frame Bandwidth Allocation 11-7
Table 11-2. OpenHCI Register Addresses 11-11
Table 12-1. PCMCIA Address Memory Ranges 12-5
Table 12-2. PCMCIA Pin Usage 12-5
Table 12-3. Supported 8-Bit Accesses 12-8
Table 12-4. Supported 16-Bit Accesses 12-8
Table 12-5. PCMCIA Legacy Usage 12-8
Table 12-6. Accesses to 8-Bit Attribute / Common / IO Memory 12-9
Table 12-7. Accesses to 16-Bit Attribute / Common / IO Memory 12-9
Table 12-8. Static Memory Controller (SMC) Register Map 12-10
Table 13-1. Boot Device Selection 13-2
Table 13-2. Address Decoding for Synchronous Memory Domains 13-3
Table 13-3. Synchronous Memory Address Decoding 13-4
Table 13-4. General SDRAM Initialization Sequence 13-4
Table 13-5. Mode Register Command Decoding for 32-bit Wide Memory Bus 13-6
Table 13-6. Sync Memory CAS 13-7
Table 13-7. Sync Memory RAS, Burst Type, and Write Burst Length 13-7
Table 13-8. Burst Length 13-7
Table 13-9. Chip Select Decoding 13-9
Table 13-10. Memory Addressing Example 13-11
Table 13-11. EP93xx SDRAM Address Ranges (16-Bit Wide Data Systems) 13-12
Table 13-12. Address Bits Used for Chip Select 13-17
Table 13-13. Synchronous Memory Controller Registers 13-17
Table 13-14. Synchronous Memory Command Encoding 13-20
Table 14-1. Receive FIFO Bit Functions 14-6
Table 14-2. Legal HDLC Mode Configurations 14-10
Table 14-3. HDLC Receive Address Matching Modes 14-13
Table 14-4. UART1 Pin Functionality 14-15
Table 14-5. DeviceCfg Register Bit Functions 14-15
Table 15-1. UART2 / IrDA Modes 15-5
Table 15-2. IonU2 Pin Function 15-5
Table 16-1. UART3 Pin Functionality 16-1
Table 16-2. DeviceCfg Register Bit Functions 16-2
Table 17-1. Bit Values to Select Ir Module 17-3
Table 17-2. Address Offsets for End-of-Frame Data 17-5
Table 17-3. MIR Frame Format 17-9
Table 17-4. DeviceCfg.IonU2 Pin Function 17-20

©Copyright 2007 Cirrus Logic, Inc.

Table 17-5. UART2 / IrDA Modes 17-21
Table 17-6. IrDA Service Memory Accesses / Second 17-22
Table 18-1. Timers Register Map 18-2
Table 19-1. Watchdog Timer Register Memory Map 19-3
Table 20-1. Real Time Clock Register Memory Map 20-4
Table 21-1. I^2S Controller Input and Output Signals 21-2
Table 21-2. Audio Interfaces Pin Assignment 21-2
Table 21-3. Transmitter FIFO's 21-3
Table 21-4. I2SClkDiv SYSCON Register Effect on I^2S Clock Generation 21-8
Table 21-5. Bit Clock Rate Generation 21-9
Table 21-6. FIFO Flags 21-12
Table 21-7. I^2S TX Registers 21-12
Table 21-8. I^2S RX Registers 21-19
Table 21-9. I^2S Configuration and Status Registers 21-25
Table 22-1. AC'97 Input and Output Signals 22-1
Table 22-2. AC'97 Register Memory Map 22-5
Table 22-3. Interaction Between RSIZE and CM 22-9
Table 22-4. Interaction Between RSIZE and CM Bits 22-11
Table 23-1. SSP Register Memory Map Description 23-13
Table 24-1. Static Programming Steps 24-2
Table 24-2. Dynamic Programming Steps 24-3
Table 24-3. PWM Registers Map 24-3
Table 25-1. Switch Definitions and Logical Safeguards to Prevent Physical Damage 25-3
Table 25-2. Touch Screen Switch Register Configurations 25-7
Table 25-3. External Signal Functions 25-16
Table 25-4. Analog Touch Screen Register Memory Map 25-17
Table 26-1. Keypad Interface Register Memory Map 26-6
Table 27-1. IDE Host to IDE Interface Definition 27-2
Table 27-2. IDE Cycle Times and Data Transfer Rates 27-7
Table 27-3. Wait State Value for the DMA M2M Register Control.PWSC 27-8
Table 27-4. HCLK Cycles to De-assert DMA Request 27-8
Table 27-5. Maximum Theoretical Bandwidths for Various Operating Modes 27-9
Table 27-6. IDE Interface Register Map 27-10
Table 28-1. EP9301 and EP9302 GPIO Port to Pin Map 28-6
Table 28-2. EP9307 GPIO Port to Pin Map 28-6
Table 28-3. EP9312 GPIO Port to Pin Map 28-7
Table 28-4. EP9315 GPIO Port to Pin Map 28-8

©Copyright 2007 Cirrus Logic, Inc.

Table 28-5. GPIO Register Address Map....28-9
Table 29-1. Security Register List....29-2
Table 30-1. Glossary....30-1
Table 31-1. EP93xx Register List....31-1

Revision History

Revision	Date	Changes
UM1	September 14, 2007	This is the Initial Release of the *EP93xx User's Guide.* This manual covers all products in the EP93xx product family. This manual is based on the content of previous User's Guides for each of the individual products in the EP93xx family. New content has been added, formatting improved, and all known documentation errors fixed. Please discard previous User's Guides and rely on this manual for your future reference needs.

©Copyright 2007 Cirrus Logic, Inc.

P.1 About the EP93xx User's Guide

This EP93xx User's Guide describes the architecture, hardware, and operation of the Cirrus Logic EP9301, EP9302, EP9307, EP9312, and EP9315 processors. It is intended to be used in conjunction with the respective EP93xx Data Sheets, which contain the full electrical specifications for the EP93xx processors.

The EP9301, EP9302, EP9307, EP9312 processors are functional subsets of the EP9315 processor. All chapters in this Guide apply to the EP9315 processor. Most, but not all, chapters apply to the EP9301, EP9302, EP9307, EP9312 processors. Table P-1 shows the maximum core frequency and the maximum high-speed bus frequency as well as number of package balls and package type for the EP93xx processors. Table P-2 shows chapter numbers and function, and which EP93xx processors include the function (or not).

Table P-1. Frequency, Package, Applicable EP93xx Processor

	EP9301	EP9302	EP9307	EP9312	EP9315
Maximum Core Frequency - MHz	166	200	200	200	200
Maximum High-Speed Bus Frequency - MHz	66	100	100	100	100
Package Type	208 LQFP	208 LQFP	272 TFBGA	352 PBGA	352 PBGA

Table P-2. Chapter Number and Function, Applicable EP93xx Processor

Chapter Number and Function	Applicable EP93xx Processor				
	EP9301	EP9302	EP9307	EP9312	EP9315
0: Preface	X	X	X	X	X
1: Introduction	X	X	X	X	X
2: ARM920T Core and Advanced High-Speed Bus	X	X	X	X	X
3: MaverickCrunch Co-processor	-	X	X	X	X
4: Boot ROM	X	X	X	X	X
5: System Controller	**X**	**X**	**X**	**X**	**X**

Copyright 2007 Cirrus Logic

Table P-2. Chapter Number and Function, Applicable EP93xx Processor (Continued)

Chapter Number and Function	Applicable EP93xx Processor				
	EP9301	EP9302	EP9307	EP9312	EP9315
6: Vectored Interrupt Controller	X	X	X	X	X
7: Raster Engine with Analog and LCD Integrated Timing and Interface	-	-	X	X	X
8: Graphics Accelerator	-	-	X	-	X
9: 1/10/100 Mbps Ethernet LAN Controller	X	X	X	X	X
10: DMA Controller	X	X	X	X	X
11: Universal Serial Bus Host Controllers	2	2	3	3	3
12: Static Memory Controller Static Memory Controller with PCMCIA	X -	X -	X -	X -	- X
13: SDRAM, SyncROM, SyncFLASH Controllers	X	X	X	X	X
14: UART1 with Modem Control Signals and HDLC	X	X	X	X	X
15: UART2 with IrDA	X	X	X	X	X
16: UART3 with HDLC	-	-	X	X	X
17: IrDA	X	X	X	X	X
18: Timers	4	4	4	4	4
19: Watchdog Timer	X	X	X	X	X
20: Real Time Clock with Software Trim	X	X	X	X	X
21: I^2S Controller	3	3	3	3	3
22: AC'97 Controller	1	1	1	1	1
23: Synchronous Serial Port	1	1	1	1	1
24: Pulse Width Modulators	2	2	1	2	2
25: Analog Touch Screen Interface/ADC	5-ADC	5-ADC	8-Wire TS	8-Wire TS	8-Wire TS
26: Keypad Interface	-	-	X	X	X
27: IDE Interface	-	-	-	2 Devices	2 Devices
28: GPIO Interface	X	X	X	X	X
29: Security	X	X	X	X	X
30: Glossary	X	X	X	X	X

Copyright 2007 Cirrus Logic

Note: "X" means Function is included; "-" means Function is not included

P.2 Related Documents from Cirrus Logic

P

1. *EP9301 Data Sheet*, Document Number - DS636PP5
2. *EP9302 Data Sheet*, Document Number - DS653PP3
3. *EP9307 Data Sheet*, Document Number - DS667PP4
4. *EP9312 Data Sheet,* Document Number - DS515PP7
5. *EP9315 Data Sheet*, Document Number - DS638PP1

P.3 Reference Documents

1. *ARM®920T Technical Reference Manual,* ARM Limited
2. *AMBA Specification* (Rev. 2.0), ARM IHI 0011A, ARM Limited
3. *AHB Example AMBA System (Addendum 01),* ARM DDI 0170A, ARM Limited
4. The co-processor instruction assembler notation can be referenced from ARM programming manuals or the *Quick Reference Card*, document number ARM QRC 0001D, ARM Limited
5. The MAC engine is compliant with the requirements of ISO/IEC 8802-3 (1993), Sections 3 and 4
6. *OpenHCI - Open Host Controller interface Specification for USB*, Release 1.0a; Compaq®, Microsoft®, National Semiconductor®
7. *ARM Co-processor Quick Reference Card*, document number ARM QRC 0001D, ARM Limited
8. *Information Technology, AT Attachment with Packet Interface - 5 (ATA/ATAPI-5)* ANSI NCITS document T13 1321D, Revision 3, 29 February 2000
9. *ARM PrimeCell PL190-Rel1v1* Revision 1.7 Technical Reference Manual DDI0181C, ARM Limited
10. Audio Codec '97, Revision 2.3, April 2002, Intel® Corporation

P.4 Notational Conventions

This document uses the following conventions:

- Internal and external Signal Names, and Pin Names use mixed upper and lower case alphanumeric, and are shown in bold font, for example, **RDLED**
- Register Bit Fields are named using upper and lower case alphanumeric: for example, SBOOT, LCSn1

Copyright 2007 Cirrus Logic

P

- Registers are named using mixed upper and lower case alphanumeric, for example, SysCfg or PxDDR. Where there are multiple registers with the same names, a lower case "x" is used as a place holder. For example, in the PxDDR registers, x represents a letter from A to H, indicating the specific port being discussed

CAUTION: In the Internal Register Map in "Internal Register Map" on page 2-17 some memory locations are listed as **Reserved**. These memory locations should not be used. Reading from these memory locations will yield invalid data. Writing to these memory locations may cause unpredictable results.

(An example register description is shown below. This description is used for the following examples.)

A specific bit may be specified in one of three ways:

1. *Register name[bit number]*, for example, *SysCfg[29]*
2. R*egister name.bit field[bit number]*, for example, *SysCfg.REV[1]*
3. R*egister name.bit field[bit name]*, for example, *SysCfg.SBOOT*

Hexidecimal numbers are referred to as *0x0000_0000*.

Binary numbers are referred to as *0000_0000b*.

P.5 Register Example

Note: This is only an example. For actual SysCfg register information, see "SysCfg" on page 5-34 .

SysCfg

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV				RSVD											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD							SBOOT	LCSn7	LCSn6	LASDO	LEEDA	LEECLK	RSVD	LCSn2	LCSn1

Address:
0x8093_009C - Read/Write, Software locked

Default:
0x0000_0000

Definition:
System Configuration Register. Provides various system configuration options.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

Copyright 2007 Cirrus Logic

P

REV:	Revision, reads chip Version number: 0 - Rev A, 1 - Rev B, 2 - Rev C, 3 - Rev D.
SBOOT:	Serial Boot Flag. This bit is read-only. 1 hardware detected Serial Boot selection 0 hardware detected Normal Boot
LCSn7, LCSn6:	Latched version of CSn7 and CSn6 respectively. These are used to define the external bus width for the boot code boot.
LASDO:	Latched version of ASDO pin. Used to select synchronous versus asynchronous boot device.
LEEDA:	Latched version of EEDAT pin.
LEECLK:	Define Internal or external boot: 1 Internal 0 External
LCSn1, LCSn2:	Define Watchdog startup action: 0 0 Watchdog disabled, Reset duration disabled 0 1 Watchdog disabled, Reset duration active 1 0 Watchdog active, Reset duration disabled 1 1 Watchdog active, Reset duration active

Copyright 2007 Cirrus Logic

P

Copyright 2007 Cirrus Logic

Chapter 1
Introduction 1

1.1 Introduction

The EP93xx processors are highly integrated systems-on-a-chip that pave the way for a multitude of next-generation consumer and industrial electronic products. Designers of digital media servers and jukeboxes, telematic control systems, thin clients, set-top boxes, point-of-sale terminals, industrial controls, biometric security systems, and GPS devices will benefit from the EP93x processors' integrated architecture and advanced features. In fact, with amazingly agile performance provided by a 166 or 200 MHz ARM920T Core, and featuring an incredibly wide breadth of peripheral interfaces, the EP93xx processors are well suited to an even broader range of high volume applications. Furthermore, by enabling or disabling the EP93xx processor's peripherals and their interfaces, designers can throttle power consumption and reduce development costs and accelerate time-to-market by creating a single platform that can be easily modified to deliver a variety of differentiated end products.

1.2 EP93xx Features

Maximum clock rates plus package types and number of balls for EP93xx processors are shown in Table 1-1.

Table 1-1. EP93xx Maximum Clock Rates, Package Type and Number of Balls

Processor	Max Core Clock Rate	Max High-Speed Bus Clock Rate	Package
EP9301	166 MHz	66 MHz	208 LQFP
EP9302	200 MHz	100 MHz	208 LQFP
EP9307	200 MHz	100 MHz	272 TFBGA
EP9312	200 MHz	100 MHz	352 PBGA
EP9315	200 MHz	100 MHz	352 PBGA

Features of the EP93xx processors are summarized in Table 1-2. Block diagrams are shown in Figure 1-1 EP9301, Figure 1-2 EP9302, Figure 1-3 EP9307, Figure 1-4 EP9312, and Figure 1-5 EP9315.

Copyright 2007 Cirrus Logic

1

Table 1-2. EP93xx Features Summary

Processor	16-Bit External Bus	32-Bit External Bus	Math Co-Processor	Raster Analog / LCD	2-D Graphics Accelerator	Ethernet MAC	IDE	USB 2.0 Host	UART	Touch Screen / ADC	GPIO	PC Card
EP9301	X	-	-	-	-	X	-	2	2	5-ADC	37	-
EP9302	X	-	X	-	-	X	-	2	2	5-ADC	37	-
EP9307	-	X	X	X	X	X	-	3	3	8-Wire/ 12-ADC	48	-
EP9312	-	X	X	X	-	X	1	3	3	8-Wire/ 12-ADC	47	-
EP9315	-	X	X	X	X	X	1	3	3	8-Wire/ 12-ADC	55	X

Note: "X" means that the function is included; "-" means that the function is not included.

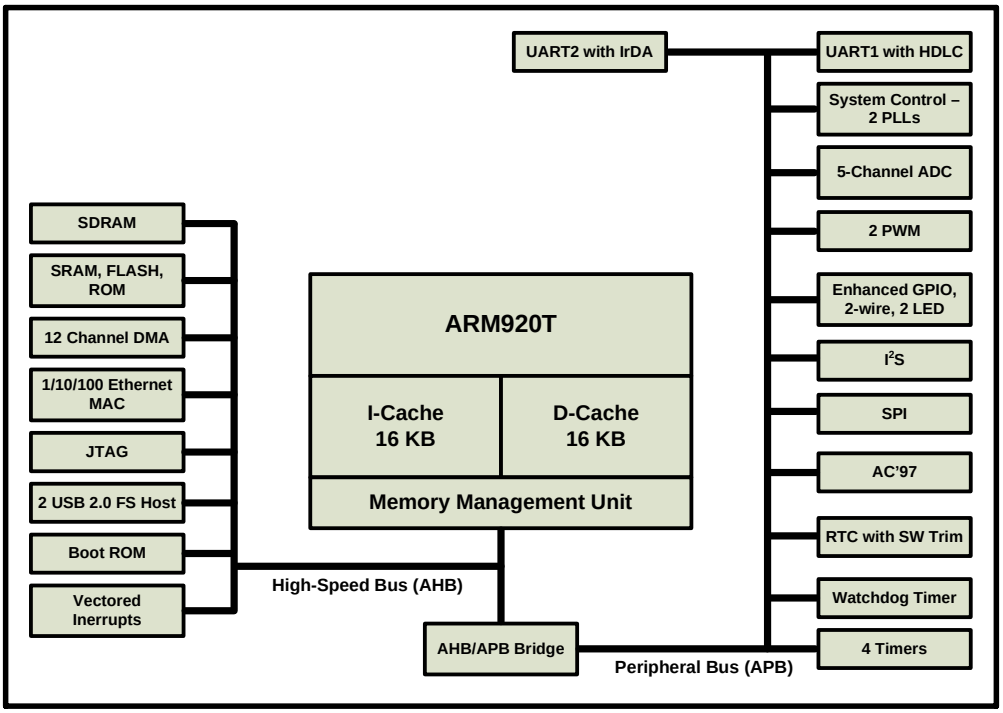


Figure 1-1. EP9301 Block Diagram

Copyright 2007 Cirrus Logic

1

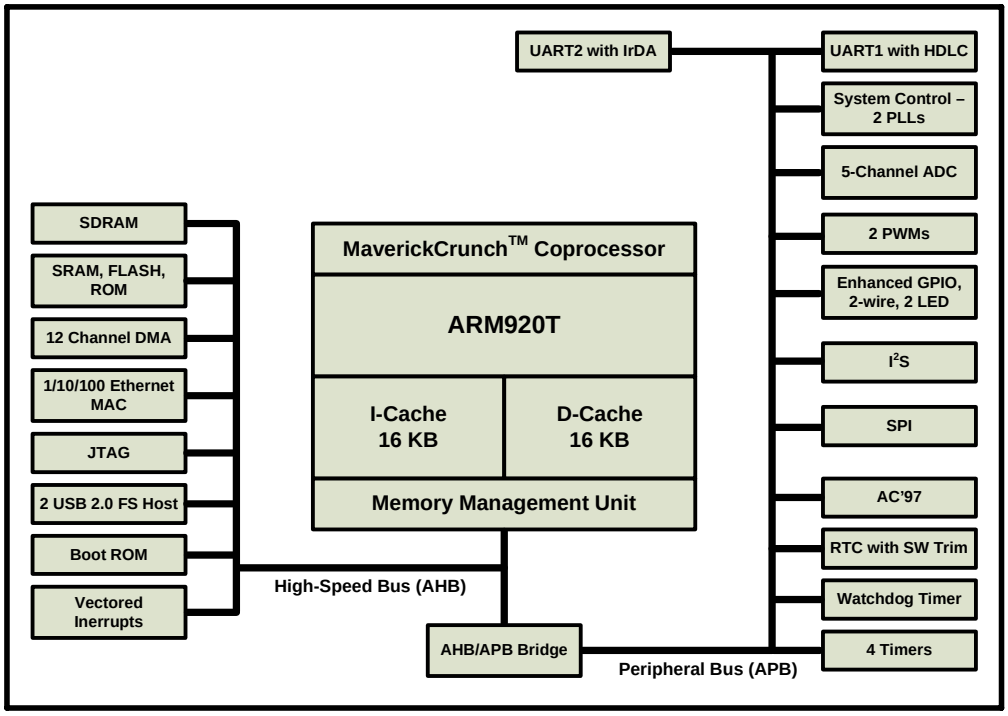


Figure 1-2. EP9302 Block Diagram

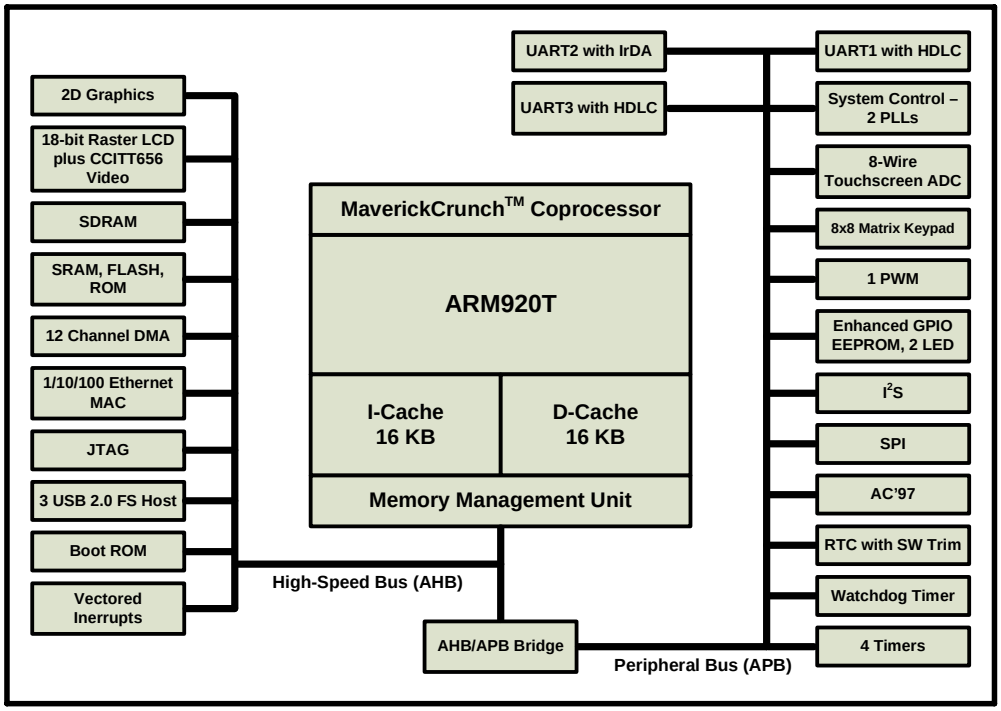


Figure 1-3. EP9307 Block Diagram

Copyright 2007 Cirrus Logic

1

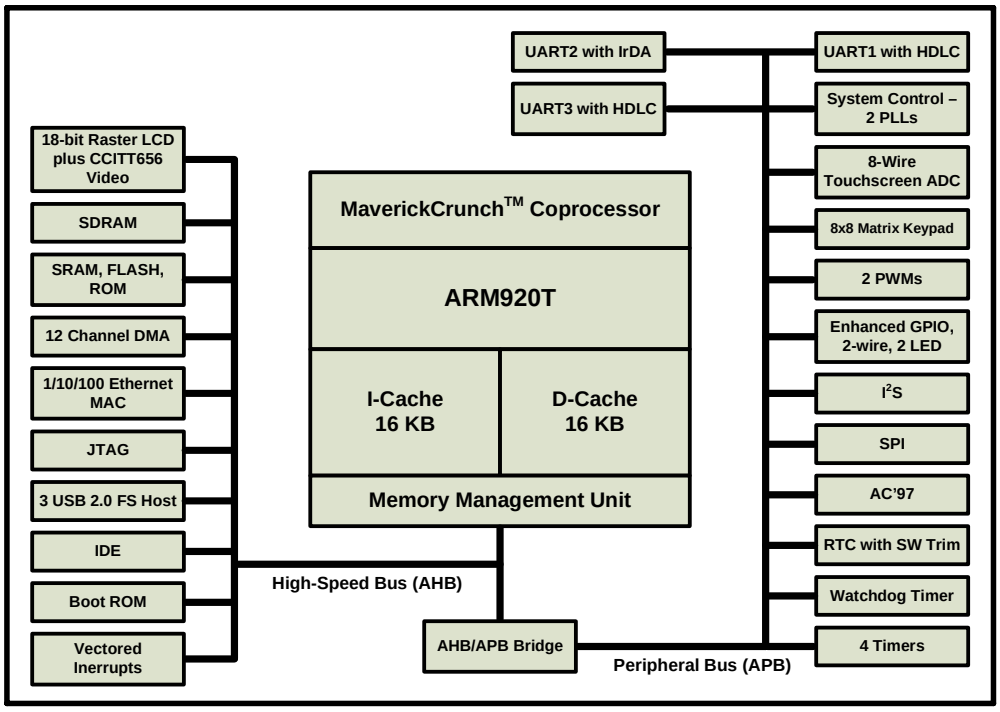


Figure 1-4. EP9312 Block Diagram

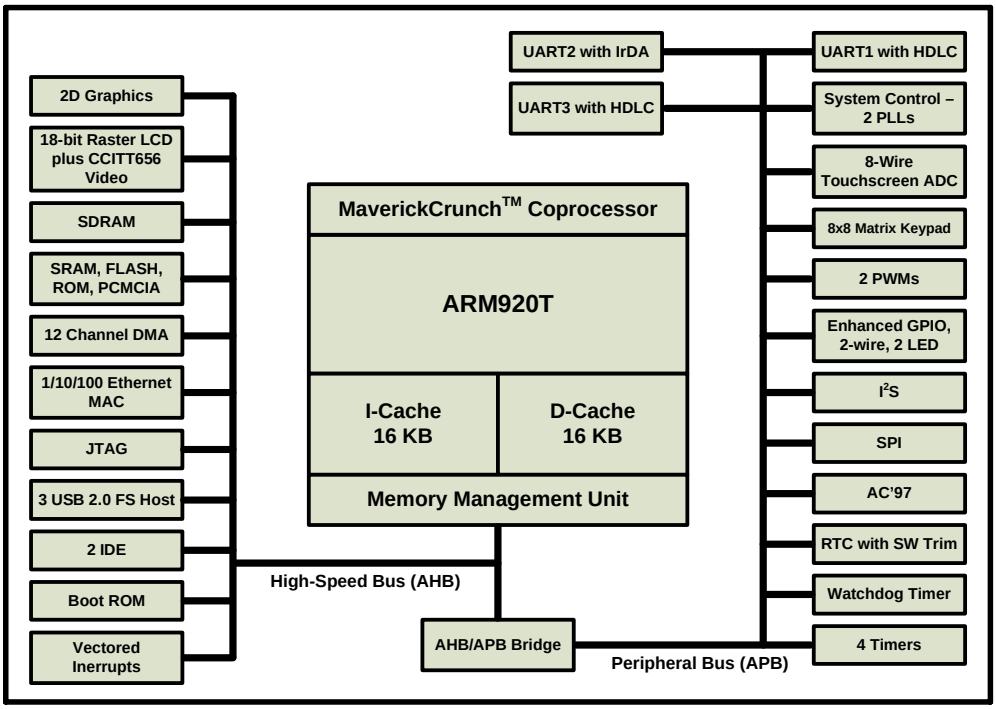


Figure 1-5. EP9315 Block Diagram

Copyright 2007 Cirrus Logic

1

Features of the EP93xx processors are:

- **ARM920T Core:**
  - 200 MHz maximum run frequency and 100 MHz maximum high-speed bus frequency for EP9302, 9307, 9312, and 9315 only
  - 166 MHz maximum run frequency and 66 MHz maximum high-speed bus frequency for EP9301 only
  - 16 KByte instruction cache and 16 KByte data cache
  - Memory Management Unit (MMU) with 64-entry Translation-Lookaside-Buffers (TLBs) enable Linux® and Windows® CE®
- **MaverickCrunch™ Co-processor** in EP9302, 9307, 9312, and 9315 only:
  - Floating point, integer and signal processing instructions
  - Optimized for digital music compression algorithms
  - Hardware interlocks allow in-line coding
- **MaverickKey™ IDs** for Digital Rights Management or Design IP Security:
  - 32-bit unique ID
  - 128-bit random ID
- **Integrated Peripherals and Interfaces:**
  - EIDE, up to 2 devices in EP9312 and 9315 only
  - 1/10/100 Mbps Ethernet MAC
  - Two-port USB 2.0 Full Speed host (OHCI) in EP9301 and 9302 only
  - Three-port USB 2.0 Full Speed host (OHCI) in EP9307, 9312, and 9315 only
  - IrDA controller, slow and fast mode
  - Two UARTs (16550 Type) in EP9301 and 9302 only:
  - - UART1 (optionally supports on-chip handling of HDLC)
  - - UART2 (optionally provides interface for IrDA controller)
  - Three UARTs (16550 Type) in EP9307, 9312, and 9315 only:
    - UART1 and UART3 (optionally support on-chip handling of HDLC)
    - UART2 (optionally provides interface for IrDA controller)
    - UART3 implements both a UART and an HDLC interface identical to that of UART1;
  - LCD and Analog Raster Interface in EP9307, 9312, and 9315 only
  - 2D Graphics Accelerator in EP9307and 9315 only
    - Line Draw

Copyright 2007 Cirrus Logic

- Block Copy
- Block Fill

- Touch Screen interface
  - 5-ADC in EP9301 and 9302 only
  - 8-Wire Touch Screen/ADC in EP9307, 9312, and 9315 only
- SPI port
- AC '97 interface
- I^2S interface with up to 6 channels
- 8x8 Matrix keypad scanner (in EP9307, EP9312, and EP9315 only)
- PCMCIA Interface supporting 8-bit or 16-bit PCMCIA (PC Card) devices in EP9315 only

- **External Memory Options**
  - 16-bit SDRAM interface (up to 4 banks) in EP9301 and 9302 only
  - 32-bit SDRAM interface (up to 4 banks) in EP9307, 9312, and 9315 only
  - 16/8-bit SRAM/Flash/ROM interface in EP9301 and 9302 only
  - 32/16/8-bit SRAM/Flash/ROM interface in EP9307, 9312, and 9315 only
  - Serial Flash interface
- **Internal Peripherals**
  - Real-Time clock with software trim
  - 12 DMA channels for data transfer to maximize system performance
  - Boot ROM
  - Dual PLLs
  - Watchdog timer
  - Two general purpose 16-bit timers
  - General purpose 32-bit timer
  - 40-bit debug timer
- **Standard General-Purpose I/Os** (GPIOs), no interrupts:
  - 18 in EP9301 and 9302 only
  - 30 in EP9307 only
  - 31 in EP9312 and 9315 only
- **Enhanced General-Purpose I/Os** (EGPIOs) plus Port F GPIOs can generate interrupts:
  - 19 in EP9301, 9302 only
  - 18 in EP9307 only

Copyright 2007 Cirrus Logic

- 16 in EP9312 only
- 24 in EP9315 only

1

1.3 EP93xx Processor Applications

The EP93xx processors can be used in a variety of applications, such as:

- Digital media servers
- Integrated home media gateways
- Digital audio jukeboxes
- Streaming audio/video players
- Telematic control systems
- Set-top boxes
- Point-of-sale terminals
- Thin clients
- Internet TVs
- Biometric security systems
- Industrial controls
- GPS & fleet management systems
- Educational toys
- Voting machines
- Medical equipment

1.4 EP93xx Processor Highlights

1.4.1 High-Performance ARM920T Core

The EP93xx Processors feature an advanced ARM920T Core design with an MMU that supports Linux®, Windows® CE®, and many other embedded operating systems. The ARM920T's 32-bit microcontroller architecture, with a five-stage pipeline, delivers impressive performance at very low power. The included 16 KByte instruction cache and 16 KByte data cache provide zero-cycle latency to the current program and data, or can be locked to provide guaranteed no-latency access to critical instructions and data. For applications with instruction memory size restrictions, the ARM920T's compressed Thumb® instruction set provides a space-efficient design that maximizes external instruction memory usage.

1.4.2 MaverickCrunch™ Co-processor for Ultra-Fast Math Processing

The EP9302, EP9307, EP9312, and EP9315 processors include an advanced MaverickCrunch co-processor that provides mixed-mode math functions to greatly accelerate the floating-point processing capabilities of the ARM920T Core. The MaverickCrunch co-

Copyright 2007 Cirrus Logic

processor simplifies the end-user's programming task by using predefined co-processor instructions, utilizing standard ARM compiler tools, and by requiring just one debugger session for the entire system. Furthermore, the integrated design provides a single instruction stream and the advantage of zero latency for cached instructions. To emulate this capability, competitors' solutions add a DSP to the system, which requires separate compiler/linker/debugger tool sets. This additional DSP requires programmers to write two separate programs and debug them simultaneously, which can result in frustration and costly delays.

1.4.3 MaverickKey™ Unique ID Secures Digital Content in OEM Designs

The EP93xx processors include MaverickKey unique hardware programmed IDs that provide an excellent solution to the growing concern over secure Web content and commerce. With Internet security playing an important role in the delivery of digital media such as books or music, traditional software methods are quickly becoming unreliable. The MaverickKey unique IDs provide OEMs with a method of utilizing specific hardware IDs for DRM (Digital Rights Management) and other authentication mechanisms.

MaverickKey uses a specific 32-bit ID and a 128-bit random ID that are programmed into the EP93xx processors through the use of laser probing technology. These IDs can then be used to match secure copyrighted content with the ID of the target device that the EP93xx processor is powering, and then deliver the copyrighted information over a secure connection. In addition, secure transactions can benefit by matching device IDs to server IDs.

MaverickKey IDs can also be used by OEMs and design houses to protect against design piracy by presetting ranges for unique IDs. For more information on securing your design using MaverickKey, please contact your Cirrus Logic sales representative.

1.4.4 Integrated Multi-Port USB 2.0 Full Speed Hosts with Transceivers

The EP9307, EP9312, and EP9315 processors integrate three USB 2.0 Full Speed Host ports while the EP9301 and EP9302 integrate two of the ports. Fully compliant to the OHCI USB 2.0 Full Speed specification (12 Mbps), the host ports can be used to provide connections to a number of external devices including mass storage devices, external portable devices such as audio players or cameras, printers, or USB hubs. Naturally, the USB host ports support the USB 2.0 Low Speed standard as well. This provides the opportunity to create a wide array of flexible system configurations.

Copyright 2007 Cirrus Logic

1

1.4.5 Integrated Ethernet MAC Reduces BOM Costs

The EP93xx processors integrate a 1/10/100 Mbps Ethernet Media Access Controller (MAC). With a simple connection to MII-based external PHYs (such as the Cirrus Logic CS8952 PHY Transceiver), an EP93xx processor-based system has easy, high-performance, cost-effective Internet capability.

1.4.6 8x8 Keypad Interface Reduces BOM Costs

The EP9307, 9312, and 9315 processors include a matrix keypad controller that scans an 8x8 array of 64 normally open, single pole switches. Any one or two keys depressed will be de-bounced and decoded. An interrupt is generated whenever a stable set of depressed keys is detected. If the keypad is not utilized, the 16 column/row pins may be used as general-purpose I/Os.

1.4.7 Multiple Booting Mechanisms Increase Flexibility

The EP93xx processors include a 16 KByte Boot ROM to set up standard configurations. The Boot ROM controls booting from either FLASH memory, the SPI serial interface, or a UART. This boot flexibility makes it easy to design user-controlled, field-upgradable systems. See Chapter 4 on page 4-1, for additional details. The EP93xx processors can also boot directly from CSn0, bypassing the Boot ROM.

1.4.8 Abundant General Purpose I/Os Build Flexible Systems

The EP93xx processors include both enhanced and standard general-purpose I/O pins (GPIOs). The enhanced GPIOs may individually be configured as inputs, outputs, or interrupt-enabled inputs. Nineteen enhanced GPIOs are in EP9301 and 9302 processors, 18 are in the EP9307 processor, and 16 are in EP9312 processor, and 24 are in the EP9315 processor.

The standard GPIOs may individually be used as inputs, outputs, or (in some cases) open-drain pins. The standard GPIOs are multiplexed with peripheral function pins, so the number available depends on the utilization of peripherals. Eighteen standard GPIOs are in EP9301 and 9302 processors, 30 are in the EP9307 processor, 31 are in the EP9312 and EP9315 processors.

Together, the enhanced and standard GPIOs facilitate easy system design with external peripherals not integrated on the EP93xx processors.

1.4.9 General-Purpose Memory Interface (SDRAM, SRAM, ROM, FLASH)

The EP93xx processors feature a unified memory address model in which all memory devices are accessed over a common address/data bus. In the EP9301 and 9302 processors, the common address/data bus is 16-bits wide, the Static Memory Controller (SMC) supports 8-bit and 16-bit devices and the SDRAM, SyncROM, and SyncFLASH synchronous memory controller supports 16-bit devices. In the EP9307, EP9312, and EP9315 processors, the common address/data bus is programmable to either 16-bits or 32-

Copyright 2007 Cirrus Logic

bits wide, the SMC supports 8-bit, 16-bit, and 32-bit devices, and the SDRAM, SyncROM, and SyncFLASH synchronous memory controller supports 16-bit and 32-bit devices. In the EP9307, EP9312, and EP9315 processors, a separate internal bus to the dynamic memory controller is dedicated to the read-only Raster/Display refresh engine.

1.4.10 12-Bit Analog-to-Digital Converter (ADC) Provides an Integrated Touch-Screen Interface or General ADC Functionality

The EP9301 and EP9302 processors include a 5-channel ADC. The EP9307, EP9212, and EP9315 processors include a 12-bit ADC, which can be utilized either as an 8-wire touch-screen interface or for general ADC functionality. The touch-screen interface performs all sampling, averaging, ADC range checking, and control for a wide variety of analog-resistive touch screens. To improve system performance, the controller only interrupts the ARM Core when a meaningful change occurs. The touch screen hardware may be disabled, and the switch matrix and ADC controlled directly for general ADC usage if desired.

1.4.11 Raster Analog / LCD Controller

The EP9307, EP9312, and EP9315 processors include a raster/LCD controller that features fully programmable video interface timing for either non-interlaced or dual scan color and grayscale flat panel displays. Resolutions up to 1024x768 pixels are supported from a unified SDRAM-based frame buffer with pixel depths of 4, 8, 16, or 18 bits. A 256x18 color lookup table, a hardware blinking cursor with up to 64x64 pixels, and an interface to smart panel displays is also included.

1.4.12 Graphics Accelerator

The EP9307 and EP9315 processors include a hardware graphics acceleration engine that improves graphic performance by handling block copy, block fill and hardware line draw operations. The graphics accelerator is used to off load graphics operations from the ARM Core.

1.4.13 PCMCIA Interface

The EP9315 processor (only) provides a PCMCIA interface that supports 8-bit or 16-bit PCMCIA PC Cards. These PCMCIA cards are credit card sized peripherals that add memory, mass storage and I/O capabilities to computer systems, and can be used to further broaden the options of a designer's platform.

Copyright 2007 Cirrus Logic

Chapter 2

ARM920T Core and Advanced High-Speed Bus (AHB) 2

2.1 Introduction

This chapter describes the ARM920T Core and the Advanced High-Speed Bus (AHB).

2.2 Overview: ARM920T Core

The ARM920T is a Harvard architecture core with separate 16 kbyte instruction and data caches with an 8-word line length. The ARM Core utilizes a five-stage pipeline consisting of fetch, decode, execute, data memory access, and write stages.

2.2.1 Features

Key features include:

- ARM V4T (32-bit) and Thumb (16-bit compressed) instruction sets
- 32-bit Advanced Micro-Controller Bus Architecture (AMBA)
- 16 kbyte Instruction Cache with lockdown
- 16 kbyte Data Cache (programmable write-through or write-back) with lockdown
- Write Buffer
- MMU for Microsoft Windows CE and Linux operating systems
- Translation Look-aside Buffers (TLB) with 64 Data and 64 Instruction Entries
- Programmable Page Sizes of 64 kbyte, 4 kbyte, and 1 kbyte
- Independent lockdown of TLB Entries
- JTAG Interface for Debug Control
- Co-processor Interface

Copyright 2007 Cirrus Logic

2

2.2.2 Block Diagram

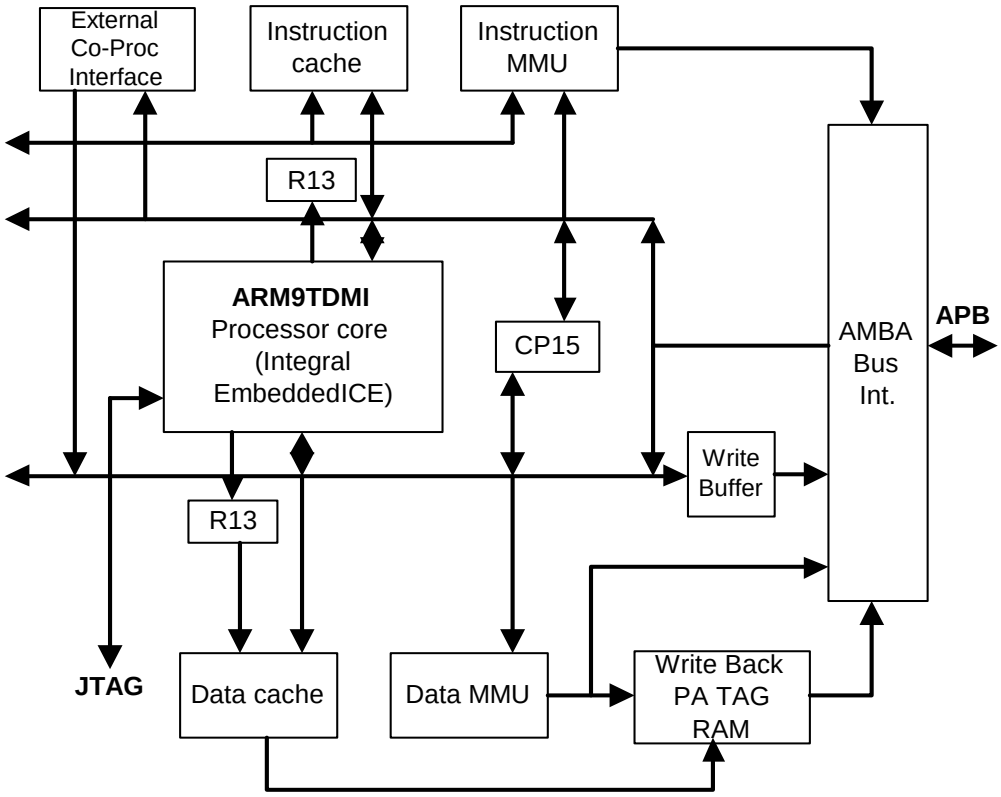


Figure 2-1. ARM920T Block Diagram

2.2.3 Operations

The ARM920T core follows a Harvard architecture and consists of an ARM9TDMI core, MMU, instruction and data cache. The core supports both the 32-bit ARM and 16-bit Thumb instruction sets.

The internal bus structure (AMBA) includes both a high speed and low speed bus. The high speed bus AHB (Advanced High-performance Bus) contains a high speed internal bus clock to synchronize co-processor, MMU, cache, DMA controller, and memory modules. AMBA includes a AHB/APB bridge to the lower speed APB (Advanced Peripheral Bus). The APB bus connects to lower speed peripheral devices such as UARTs and GPIOs.

The MMU provides memory address translation for all memory and peripherals designed to remap memory devices and peripheral address locations. Sections, large, small and tiny pages are programmable to map memory in 1 Mbyte, 64 kbyte, 4 kbyte, 1 kbyte size blocks. To increase system performance, a 64-entry translation look-aside buffer will cache 64 address locations before a TLB miss occurs.

Copyright 2007 Cirrus Logic

A 16 kbyte instruction and a 16 kbyte data cache are included to increase performance for cache-enabled memory regions. The 64-way associative cache also has lock-down capability. A 16-word Write Buffer allows cached instructions to be fetched and decoded while the Write Buffer sends data to external memory.

The ARM920T Core supports a number of co-processors, including the MaverickCrunch co-processor by means of a specific pipeline architecture interface.

2.2.3.1 ARM9TDMI Core

ARM9TDMI core is responsible for executing both 32-bit ARM and 16-bit Thumb instructions. Each provides a unique advantage to a system design. Internally, the instructions enter a 5-stage pipeline. These stages are:

- Instruction Fetch
- Instruction Decode
- Execute
- Data Memory Access
- Register Write

All instructions are fully interlocked. This mechanism will delay the execution stage of a instruction if data in that instruction comes from a previous instruction that is not available yet. This simply insures that software will function identically across different implementations.

For memory access instructions, the base register used for the access will be restored by the ARM Core in the event of an Abort exception. The base register will be restored to the value contained in it immediately before execution of the instruction.

The ARM9TDMI core memory interface includes a separate instruction and data interface to allow concurrent access of instructions and data to reduce the number of CPI (cycles per instruction). Both interfaces use pipeline addressing. The core can operate in big and little endian mode. Endianess affects both the address and the data interfaces.

The memory interface executes four types of memory transfers: sequential, non-sequential, internal, and co-processor. It will also support uni- and bi-directional transfer modes.

The core provides a debug interface called JTAG (Joint Testing Action Group). This interface provides debug capability with five external control signals:

- **TDO** - Test Data Out
- **TDI** - Test Data In
- **TMS** - Test Mode Select
- **TCK** - Test Clock
- **nTRST** - Test Reset

There are six scan chains (0 through 5) in the ARM9TDMI controlled by the JTAG Test Access Port (TAP) controller. Details on the individual scan chain function and bit order can be found in the ARM920T Technical Reference Manual.

Copyright 2007 Cirrus Logic

2.2.3.2 Memory Management Unit

The MMU provides the translation and access permissions for the address and data ports for the ARM9TDMI core. The MMU is controlled by page tables stored in system memory and accessed using the CP15 register 1. The main features of the MMU are as follows:

- Address Translation
- Access Permissions and Domains
- MMU Cache and Write Buffer Access

2.2.3.2.1 Address Translation

The virtual address from the ARM920T core is modified by R13 internally to create a modified virtual address. The MMU then translates the modified virtual address from R13 by the CP15 register 3 into a physical address to access external memory or a device. The MMU looks for the physical address from the Translation Table Base (TTB) in system memory. It will also update the TLB cache.

The TLB is two 64-entry caches, one for data and one for instruction. If the physical address for the current virtual address is not found in the TLB (miss), the ARM Core will go to external memory and look for the TTB in system memory. The internal translation table walks hardware steps through the page table setup in external memory for the appropriate physical address.

When the physical address is acquired, the TLB is updated. When the address is found in the TLB, system performance will increase since additional cycles to access memory and update the TLB are avoided.

Translation of system memory is done by breaking up the memory into different size blocks called sections, large pages, small pages, and tiny pages. System memory and registers can be remapped by the MMU. The block sizes are as follows:

- Section - 1 Mbyte
- Large Page - 64 kbyte
- Small Page - 16 kbyte
- Tiny Page - 1 kbyte

2.2.3.2.2 Access Permission and Domains

Access to any section or page of memory is dependent on its domain. The page table in external memory also contains access permissions for all sub-divisions of external memory. Access to specific instructions or data has three possible states:

- *Client*: Access permissions based on the section or page table descriptor
- *Manager*: Ignore access permissions in the section or page table descriptor
- *No access*: any attempted access generates a domain fault

Copyright 2007 Cirrus Logic

2.2.3.2.3 MMU Enable

Enabling the MMU allows system memory control, but is also required if the Data Cache and the Write Buffer are to be used. Features are enabled for specific memory regions, as defined in the system page table. MMU enablement is done via CP15 register 1. The procedure is as follows:

1. Program the Translation Table Base (TTB) and domain access control registers
2. Create level 1 and level 2 pages for the system, and enable the Data Cache and the Write Buffer
3. Enable the MMU via bit 0 of CP15 register 1.

2.2.3.3 Cache and Write Buffer

Cache configuration is 64-way set associative. There is a 16 kbyte instruction cache and a 16 kbyte data cache. The caches have the following characteristics:

- 8 words per line, with 1 valid bit and 2 dirty bits per line to allow half-line write-backs
- Write-through or write-back capability, selectable per memory region defined by the MMU
- Pseudo random or round robin replacement algorithms for cache misses. This is determined by the RR bit (bit 14) in CP15 register 1. On a cache miss (instruction or data not in the respective cache), an 8-word line is fetched from memory and loaded into the cache
- Independent cache lock-down with granularity of 1/64th of total cache size or 256 bytes for both instructions and data. Lock-down of the cache will prevent an eight-word cache line fill into that region of the cache
- For compatibility with Windows CE and to reduce latency, physical addresses for data cache entries are stored in the PA TAG RAM, which is used for cache line write-back operations without need of the MMU. This prevents a possible TLB miss that would degrade performance
- The Write Buffer has a depth of 16 data words. If enabled, writes are sent to the Write Buffer directly from the Data Cache or from the CPU (in the event of a cache miss or if the cache is not enabled).

2.2.3.3.1 Instruction Cache Enable

- At reset, the Instruction Cache is disabled
- A write to bit 12 of CP15 register 1 will enable or disable the Instruction Cache. If the Instruction Cache (I-Cache) is enabled without the MMU enabled, all accesses are treated as cacheable
- If the I-Cache is disabled, current contents are ignored. If re-enabled before a reset, contents will be unchanged, but may not be coherent with eternal memory. If so, contents must be flushed before re-enabling.

Copyright 2007 Cirrus Logic

2.2.3.3.2 Data Cache Enable

- A write to bit 2 of CP15 register 1 will enable or disable the Data Cache (D-Cache)/Write Buffer
- The D-Cache may only be enabled when the MMU is enabled. All data accesses are subject to MMU and permission checks
- If disabled, current contents are ignored. If re-enabled before a reset, contents will be unchanged, but may not be coherent with external memory. Depending on system software, a clean and invalidate action may be required before re-enabling.

2.2.3.3.3 Write Buffer Enable

- The Write Buffer is enabled via the page table entries in the MMU. The Write buffer cannot be enabled unless the MMU is enabled.

2.2.4 Co-processor Interface

The MaverickCrunch co-processor is explained in detail in Chapter 3 on page 3-1. The relationship between the ARM co-processor instructions and MaverickCrunch co-processor is also explained in Chapter 3.

The ARM co-processor instruction set includes:

- LDC - Load co-processor from memory
- STC - Store co-processor register from memory
- MRC - Move to ARM register from co-processor register
- MCR - Move to co-processor register from ARM register

The ARM co-processor has sixteen (C0 through C15) 64-bit registers for data transfer and data manipulation. See Chapter 3, Section 3.2 on page 3-8 for a code example.

2.2.5 AMBA AHB Bus Interface Overview

The AHB (Advanced High-Performance Bus) is the high-performance system backbone bus. Figure 2-2 on page 2-7 shows a typical AMBA AHB System.

The AHB connects devices that require high bandwidth, such as DMA controllers, external memory, and co-processors. The AHB supports:

- Burst Transactions
- Split Transactions
- Bus Master hand-over to devices such as the MaverickCrunch co-processor or DMA controller
- Single clock edge operations

The APB (Advanced Peripheral Bus) is a lower bandwidth, but lower power, bus that provides:

Copyright 2007 Cirrus Logic

- Latched address and control
- A simple Interface to on-chip peripherals such as UARTs and AC'97.

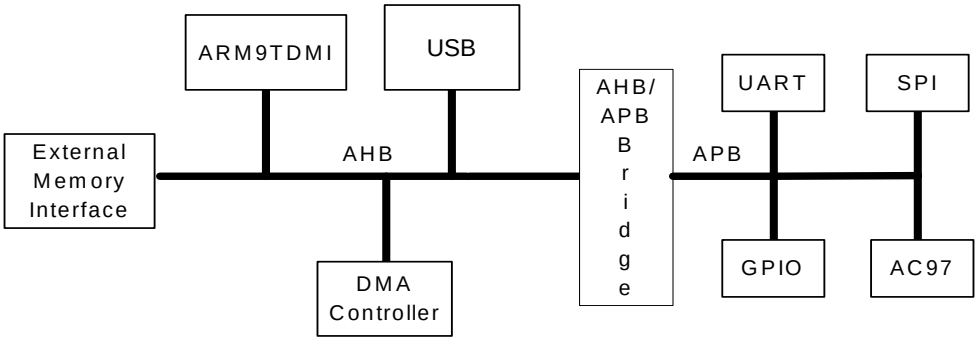


Figure 2-2. Typical AMBA AHB System

2.2.6 AHB Implementation Details

Peripherals or the external memory interface that have high bandwidth and low latency requirements are connected to the CPU using the AHB bus. The peripherals include the Vectored Interrupt Controllers (VIC1, VIC2), DMA, LCD/Raster registers, USB host, IDE, Ethernet MAC and the bridge to the APB interface. The AHB/APB Bridge transparently converts the AHB accesses into the slower speed APB accesses. All of the control registers for the APB peripherals are programmed using the AHB/APB bridge interface. The main AHB data and address lines are configured using a multiplexed bus. This removes the need for three state buffers and bus holders, and simplifies bus arbitration. Figure 2-3 on page 2-8 shows the main data paths in the processor's AHB implementation.

Copyright 2007 Cirrus Logic

2

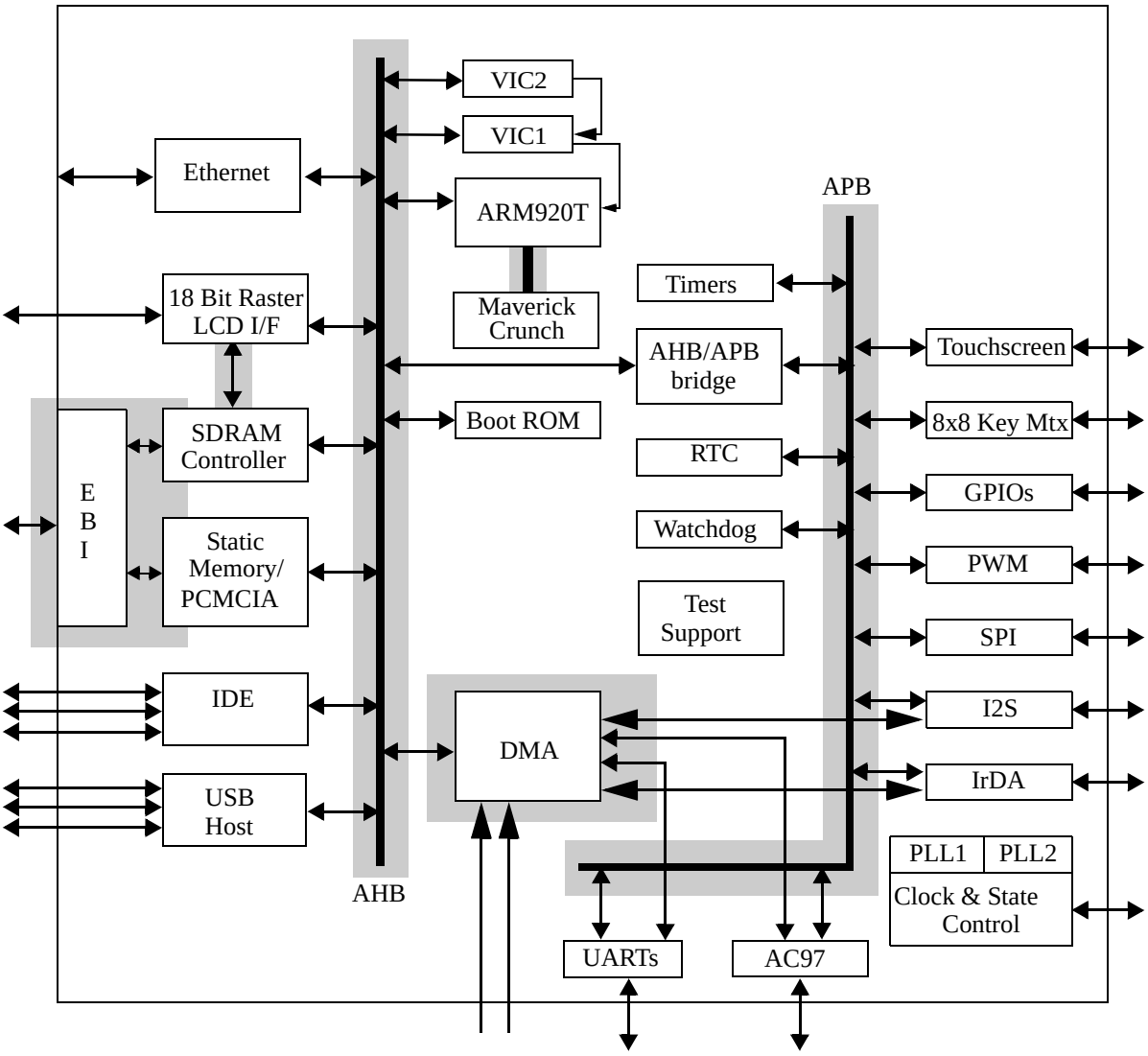


Figure 2-3. Main Data Paths

Before an AMBA-to-AHB transfer can commence, the bus master must be granted access to the bus. This process is started by the master asserting a request signal to the Arbiter. The Arbiter then indicates when the master will be granted use of the bus. A granted bus master starts an AMBA-to-AHB transfer by driving the address and control signals. These signals provide information on the address, direction and width of the transfer, as well as indicating whether the transfer is part of a burst.

Two different forms of burst transfers are allowed:

- Incrementing bursts, which do not wrap at address boundaries
- Wrapping bursts, which wrap at particular address boundaries.

Copyright 2007 Cirrus Logic

A write data bus is used to move data from the master to a slave, while a read data bus is used to move data from a slave to the master. Every transfer consists of:

- An address and control cycle
- One or more cycles for the data.

In normal operation a master is allowed to complete all the transfers in a particular burst before the arbiter grants another master access to the bus. However, in order to avoid excessive arbitration latencies, it is possible for the arbiter to break up a burst, and, in such cases, the master must re-arbitrate for the bus in order to complete the remaining transfers in the burst.

2.2.7 Memory and Bus Access Errors

There are several possible sources of access errors:

- Reads to reserved or undefined register memory addresses will return indeterminate data. Writes to reserved or undefined memory addresses are generally ignored, but this behavior is not guaranteed. Many register addresses are not fully decoded, so aliasing may occur. Addresses and memory ranges listed as **Reserved** should not be accessed; access behavior to these regions is not defined
- Access to non-existent registers or memory may result in a bus error
- Any access to the APB control register space will complete normally, as these devices have no means of signaling an error
- Access to non-existent AHB or APB registers may result in a bus error, depending on the device and nature of the error. Device specific access rules are defined in the device descriptions
- External memory access is controlled by the Static Memory Controller (SMC) or the Synchronous Dynamic RAM (SDRAM) controller. In general, access to non-existent external memory will complete normally, with reads returning random false data.

2.2.8 Bus Arbitration

The arbitration mechanism is used to ensure that only one master has access to the bus that it controls at any one time. The Arbiter performs this function by observing a number of different requests to use the bus, and then deciding which is currently the highest priority master requesting the bus.

The arbitration scheme can be broken down into three main areas:

- The main AHB system bus Arbiter
- The SDRAM slave interface Arbiter
- The EBI bus Arbiter

Copyright 2007 Cirrus Logic

2.2.8.1 Main AHB Bus Arbiter

This Main AHB Bus Arbiter controls bus master arbitration for the AHB bus. The AHB bus has eight master interfaces:

- ARM920T
- DMA controller
- USB hosts (USB1, 2, 3)
- Ethernet MAC
- LCD/Raster
- Raster Hardware Cursor.

These interfaces have an order of priority that is linked closely with the power saving modes Halt and Standby. These power saving modes force the Arbiter to grant the default bus master, in this case, the ARM920T.

The order of priority of the bus masters, from highest to lowest, is shown in Table 2-1.

Table 2-1. AHB Arbiter Priority Scheme

Priority Number	PRIORITY 00 (Reset value)	PRIORITY 01	PRIORITY 10	PRIORITY 11
1	Raster Cursor	Raster	Raster	Raster
2	MAC	Raster Cursor	Raster Cursor	DMA
3	USB	MAC	DMA	MAC
4	DMA	USB	USB	USB
5	ARM920T	ARM920T	MAC	Raster Cursor
6	Raster	DMA	ARM920T	ARM920T

The priority of the arbiter may be programmed via the BusMstrArb register in the Clock and State Controller. The arbiter can also be programmed to degrant one of these masters: DMA, USB Host or Ethernet MAC if an interrupt (IRQ or FIQ) is pending or being serviced. This prevents one of these masters from blocking important interrupt service routines. These masters are thereby prevented from accessing the bus, that is, their bus requests are masked until the IRQ/FIQ is removed (by the Interrupt Service Routine). After the IRQ/FIQ is removed, their bus requests will again be recognized. The default is to program the arbiter so that it does *not* degrant any of these masters.

In normal operation, when the ARM920T is granted the bus and a request to enter Halt mode is received, the ARM920T is de-granted from the AHB bus. Any other master requesting the bus during Halt mode (according to it's priority) will be granted the bus. In the case of entry into Standby mode, the dummy master will be granted the bus, which simply performs IDLE transfers. In this way, all the masters except the ARM920T can be used during Halt mode, but are shutdown upon entry into Standby mode.

Copyright 2007 Cirrus Logic

2.2.8.2 SDRAM Slave Arbiter

The SDRAM Slave Arbiter prioritizes between accesses from the AHB bus and the Raster DMA bus. If an access request from the AHB arrives at the same time as an access request from the Raster DMA, the Raster DMA will be given access while the AHB request is queued.

2.2.8.3 EBI Bus Arbiter

The EBI Bus Arbiter is used to arbitrate between accesses from the SDRAM controller and the Static Memory controller, where priority is given to accesses from the SDRAM controller.

2.3 AHB Decoder

The AHB Decoder contains the device memory map for all of the AHB masters/slaves and for the APB bridge. When a particular address range is selected, the appropriate signal is generated as defined in Table 2-2.

(For additional information, see 17, "Reference Documents" on page P-3.

Table 2-2. AHB Peripheral Address Range

Address Range	Register Width	Peripheral Type	Peripheral
0x800D_0000 - 0x800F_FFFF	-	-	Reserved
0x800C_0000 - 0x800C_FFFF	32	AHB	VIC2
0x800B_0000 - 0x800B_FFFF	32	AHB	VIC1
0x800A_0000 - 0x800A_FFFF	32	AHB	IDE
0x8009_0000 - 0x8009_FFFF	32	AHB	Boot ROM physical address
0x8008_0000 - 0x8008_FFFF	32	AHB	SRAM Controller/ PCMCIA
0x8007_0000 - 0x8007_FFFF	-	-	Reserved
0x8006_0000 - 0x8006_FFFF	32	AHB	SDRAM Controller
0x8005_0000 - 0x8005_FFFF	-	-	Reserved
0x8004_0000 - 0x8004_FFFF	-	-	Reserved
0x8003_0000 - 0x8003_FFFF	32	AHB	Raster
0x8002_0000 - 0x8002_FFFF	32	AHB	USB Host
0x8001_0000 - 0x8001_FFFF	32	AHB	Ethernet MAC
0x8000_0000 - 0x8000_FFFF	32	AHB	DMA

Note: Due to decoding optimization, the AHB peripheral registers are aliased throughout each peripherals register bank. Do not attempt to access an unspecified register within the bank.

2.3.1 AHB Slave

An AHB Slave responds to transfers initiated by bus masters. The slave uses signals from the decoder to determine when it should respond to a bus transfer. All other signals required for the transfer, such as the address and control information, are generated by the bus master.

Copyright 2007 Cirrus Logic

2.3.2 AHB-to-APB Bridge

The AHB-to-APB Bridge is an AHB slave that provides an interface between the high-speed AHB and the low-power APB. Read and write transfers on the AHB are converted into equivalent transfers on the APB. As the APB is not pipelined. Wait states are added during transfers to and from the APB when the AHB is required to wait for the APB.

The main sections of this bridge are:

- AHB slave bus interface
- APB transfer state machine, which is independent of the device memory map
- APB output signal generation.

2.3.2.1 Function and Operation of the AHB-to-APB Bridge

The AHB-to-APB Bridge responds to access requests from the currently granted AHB master. The AHB accesses are then converted into APB accesses.

If an undefined location is accessed, operation of the system continues as normal, but no peripherals are selected. The APB bridge acts as the only master on the APB.

The APB memory map is shown in Table 2-3.

Table 2-3. APB Peripheral Address Range

Address Range	Register Width	Peripheral Type	Peripheral
0x8095_0000 - 0x9000_FFFF	-	-	Reserved
0x8094_0000 - 0x8094_FFFF	16	APB	Watchdog Timer
0x8093_0000 - 0x8093_FFFF	32	APB	Syscon
0x8092_0000 - 0x8092_FFFF	32	APB	Real time clock
0x8091_0000 - 0x8091_FFFF	16	APB	Pulse Width Modulation
0x8090_0000 - 0x8090_FFFF	32	APB	Touchscreen
0x808F_0000 - 0x808F_FFFF	16	APB	Key Matrix
0x808E_0000 - 0x808E_FFFF	32	APB	UART3
0x808D_0000 - 0x808D_FFFF	8	APB	UART2
0x808C_0000 - 0x808C_FFFF	32	APB	UART1
0x808B_0000 - 0x808B_FFFF	32	APB	IrDA
0x808A_0000 - 0x808A_FFFF	16	APB	SPI
0x8089_0000 - 0x8089_FFFF	-	-	Reserved
0x8088_0000 - 0x8088_FFFF	32	APB	AAC
0x8087_0000 - 0x8087_FFFF	-	-	Reserved
0x8086_0000 - 0x8086_FFFF	-	-	Reserved
0x8085_0000 - 0x8085_FFFF	-	-	Reserved
0x8084_0000 - 0x8084_FFFF	16	APB	GPIO
0x8083_0000 - 0x8083_FFFF	32	APB	Security
0x8082_0000 - 0x8082_FFFF	32	APB	I2S
0x8081_0000 - 0x8081_FFFF	32	APB	Timers
0x8080_0000 - 0x8080_FFFF	-	-	Reserved
0x8010_0000 - 0x807F_FFFF	-	-	Reserved

Copyright 2007 Cirrus Logic

Note: Due to decoding optimization, the APB peripheral registers are aliased throughout each peripherals register bank. Do not attemp to access an unspecified register within the bank.

2.3.3 APB Slave

An APB Slave responds to accesses initiated by bus masters. The slave uses signals from the decoder to determine when it should respond to a bus access. All other signals required for the access, such as the address and control information, are generated by the AHB-to-APB Bridge.

2.3.4 Register Definitions

The ARM920T Core has thirty seven 32-bit internal registers, where some are modal and some are banked. If operating in Thumb instructions state, the ARM Core must switch to ARM instructions state before taking an exception. The return instruction will restore the ARM Core to the Thumb state. Most tasks are executed out of User mode. The ARM920T Core's operating modes are shown in Table 2-4.

Table 2-4. ARM920T Core Operating Modes

Mode	Description
User	Unprivileged normal operating mode
FIQ	Fast interrupt (high priority) mode when FIQ is asserted
IRQ	Interrupt request (normal) mode when IRQ is asserted
Supervisor	Software interrupt instruction (SWI) or reset will cause entry into this mode.
Abort:	Memory access violation will cause entry into this mode.
Undef	Undefined instructions mode
System	Privileged mode. Uses same registers as User mode

Table 2-5 illustrates the use of all registers for the ARM920T Core's operating modes. Each will bank or store a specific number of registers. Banked register information is not shared between modes. FIQs bank the largest number of registers, and increase performance by reducing the need to push/pop registers from the stack.

Copyright 2007 Cirrus Logic

2

Table 2-5. Register Organization Summary

	Privileged Modes						
		Exception Modes					
User	**System**	**Supervisor**	**Abort**	**Undefined**	**IRQ**	**FIQ**	
r0	r0	r0	r0	r0	r0	r0	Thumb state low registers
r1	r1	r1	r1	r1	r1	r1	
r2	r2	r2	r2	r2	r2	r2	
r3	r3	r3	r3	r3	r3	r3	
r4	r4	r4	r4	r4	r4	r4	
r5	r5	r5	r5	r5	r5	r5	
r6	r6	r6	r6	r6	r6	r6	
r7	r7	r7	r7	r7	r7	r7	
r8	r8	r8	r8	r8	r8	r8_fiq	Thumb state high registers
r9	r9	r9	r9	r9	r9	r9_fiq	
r10	r10	r10	r10	r10	r10	r10_fiq	
r11	r11	r11	r11	r11	r11	r11_fiq	
r12	r12	r12	r12	r12	r12	r12_fiq	
r13(sp)	r13	r13_svc	r13_abt	r13_und	r13_irq	r13_fiq	
r14(lr)	r14	r14_svc	r14_abt	r14_und	r14_irq	r14_fiq	
r15(pc)	pc	pc	pc	pc	pc	pc	
cpsr	cpsr	cpsr	cpsr	cpsr	cpsr	cpsr	
		spsr_svc	spsr_abt	spsr_und	spsr_irq	spsr_fiq	

Note: Colored areas represent banked registers.

User mode in Thumb state limits access to the low registers r0-r7. To access to the high registers, the ARM Core must first revert to the ARM state. The high registers are:

- r0-r12: General purpose read/write 32-bit registers
- r13 (sp): Stack Pointer
- r14 (lr): Link Register
- r15 (pc): Program Counter
- cpsr: Current Program Status Register containing condition codes and operating modes

Copyright 2007 Cirrus Logic

- spsr: Saved Program Status Register contains CPSR after occurrence of an exception

CP15 has 16 registers that control the core as described in Table 2-6.

Table 2-6. CP15 ARM920T Register Description

Register	Description
0	**ID Code:** (Read/Only) This register returns a 32-bit device ID code. ID Code data includes the core type, revision, part number etc. Access to this register is via the instruction MRC p15 0, Rd, c0, c0, 0. **Cache Code:** This register will return cache type, size and length of both I-Cache and D-Cache, and associativity. Access to this register is via the instruction MRC p15 0, Rd, c0, c0, 1.
1	**Control Register:** (Read/Write) This register is used to enable: MMU, instruction and data cache, round robin replacement 'RR'-bit, system protection, ROM protection, and clocking mode. Read/Write Instructions are: MRC p15, 0, Rd, c1, c0, 0 - Read control register - value stored in Rd MCR p15, 0, Rd, c1, c0, 0 - Write control register - value first loaded into Rd
2	**Translation Base Table:** (Read/Write) This register contains the start address of the first level translation table. The upper 18 bits represent the pointer to the table base. The lower 14 bits should be all zeroes for a write, unpredictable if read. MRC p15, 0, Rd, c2, c0, 0 - Read TTB MCR p15, 0, Rd, c2, c0, 0 - Write TTB
3	**Domain Access Control:** (Read/Write) This register specifies permissions for each of the 16 domains. Read/Write Instructions are: MRC p15, 0, Rd, c3, c0, 0 MCR p15, 0, Rd, c3, c0, 0
4	**Reserved:** Do not access. Unpredictable behavior may result.
5	**Fault Status:** (Read/Write) This register indicates the type of fault and the domain of the most recent data abort. Read/Write Instructions are: MRC p15, 0, Rd, c5, c0, 0 - read data FSR value MCR p15, 0, Rd, c5, c0, 0 - write data FSR value
6	**Fault Address:** (Read/Write) This register contains the address of the last data access abort. Read/Write Instructions are: MRC p15, 0, Rd, c6, c0, 0 - read FAR data MCR p15, 0, Rd, c6, c0, 0 - write FAR data
7	**Cache Operation:** (Write/Only) This register configures, or performs a clean (flush) of, the cache and write buffer when written to. Example: MRC p15, 0, Rd, c7, c7, 0 - Invalidate I/D-cache MRC p15, 0, Rd, c7, c5, 0 - Invalidate I-Cache
8	**TLB Operation:** (Write/Only) This register configures, or performs a clean (flush) of, the TLB when written to. Example: MRC p15, 0, Rd, c8, c7, 0 - Invalidate TLB
9	**Cache Lockdown:** (Read/Write) This register prevents certain existing cache-lines from being overwritten (locked) during a new cache-line fill. Examples: MRC p15, 0, Rd, c9, c0, 1- Write lockdown base pointer for D-Cache MRC p15, 0, Rd, c9, c0, 1 - Write lockdown base pointer for I-Cache
10	**TLB Lockdown:** (Read/Write) This register prevents existing TLB entries from being erased during a table walk. Examples: MRC p15, 0, Rd, c10, c0, 1- Write lockdown base pointer for data TLB entry MRC p15, 0, Rd, c10, c0, 1 - Write lockdown base pointer for instruction TLB entry

Copyright 2007 Cirrus Logic

Table 2-6. CP15 ARM920T Register Description (Continued)

Register	Description
11,12,14	**Reserved**
13	**FCSE PID Register:** (Read/Write) ARM9TDMI core addresses ranging from 0 to 32MB are translated by this register to A + FCSE*32MB and then sent to the MMU. If turned off, straight addresses are sent to the MMU.
15	**Test Register Only:** Reads or writes will cause unpredictable behavior.

2.3.5 Memory Map

The memory map for Synchronous Memory Boot and Asynchronous Memory Boot is shown in Table 2-7.

If internal Boot Mode is selected and the register BootModeClr has been written, the address range 0x0000_0000 -> 0x0000_FFFF is occupied by the internal Boot ROM until the internal Boot Code is completed. After boot completion, either Synchronous or Asynchronous memory is re-mapped to occupy this address space.

NOTE: Some memory locations are listed as **Reserved**. These memory locations should not be used. Reading from these memory locations will yield invalid data. Writing to these memory locations may cause unpredictable results.

Table 2-7. Global Memory Map for the Two Boot Modes

Address Range	Sync Memory Boot	Async Memory Boot
	ASD0 Pin = 1	**ASD0 Pin = 0**
0xF000_0000 - 0xFFFF_FFFF	Async memory (nCS0)	Sync memory (nSDCE3)
0xE000_0000 - 0xEFFF_FFFF	Sync memory (nSDCE2)	Sync memory (nSDCE2)
0xD000_0000 - 0xDFFF_FFFF	Sync memory (nSDCE1)	Sync memory (nSDCE1)
0xC000_0000 - 0xCFFF_FFFF	Sync memory (nSDCE0)	Sync memory (nSDCE0)
0x9000_0000 - 0xBFFF_FFFF	Not Used	Not Used
0x8080_0000 - 0x8FFF_FFFF	APB mapped registers	APB mapped registers
0x8010_0000 - 0x807F_FFFF	Reserved	Reserved
0x8000_0000 - 0x800F_FFFF	AHB mapped registers	AHB mapped registers
0x7000_0000 - 0x7FFF_FFFF	Async memory (nCS7)	Async memory (nCS7)
0x6000_0000 - 0x6FFF_FFFF	Async memory (nCS6)	Async memory (nCS6)
0x5000_0000 - 0x5FFF_FFFF	Reserved	Reserved
0x4000_0000 - 0x4FFF_FFFF	PCMCIA (Slot 0)	PCMCIA (Slot 0)
0x3000_0000 - 0x3FFF_FFFF	Async memory (nCS3)	Async memory (nCS3)
0x2000_0000 - 0x2FFF_FFFF	Async memory (nCS2)	Async memory (nCS2)
0x1000_0000 - 0x1FFF_FFFF	Async memory (nCS1)	Async memory (nCS1)
0x0001_0000 - 0x0FFF_FFFF	Sync memory (nSDCE3)	Async memory (nCS0)
0x0000_0000 - 0x0000_FFFF	Sync memory (nSDCE3) or Internal Boot ROM if INTBOOT is selected	Async memory (nCS0) or Internal Boot ROM if INTBOOT is selected

Copyright 2007 Cirrus Logic

Note: The shaded memory areas are dedicated to system registers. Details of these registers are in Table 2-8.

2.3.6 Internal Register Map

Table 2-8 on page 2-17 shows the memory map for internal registers. Registers are set to their default state by the **RSTOn** pin input or by the **PRSTn** pin input. Some state conserving registers are reset only by the **PRSTn** pin. All registers are read/write unless otherwise specified.

2.3.6.1 Memory Access Rules

Any memory address not specifically assigned to a register should be avoided. Reads to register memory addresses labelled Reserved, Unused or Undefined will return indeterminate data. Writes to register memory addresses labelled Reserved, Unused or Undefined are generally ignored, but this behavior is not guaranteed. Many register addresses are not fully decoded, so aliasing may occur. Addresses and memory ranges listed as Reserved (RSVD) should not be accessed; behavior resulting from accesses to these regions is not defined.

The SW Lock field identifies registers with a software lock. A software lock prevents the register from being written (unless an unlock operation is performed immediately prior to the write). Any register whose accidental alteration could cause system damage may be controlled with a software lock. Each peripheral with software lock capability has its own software lock register.

Within a register definition, a reserved bit indicated by the name RSVD, means the bit is not accessible. Software should mask the RSVD bits when doing bit reads. RSVD bits will ignore writes, that is writing a zero or a one has no affect.

Register bits identified as NC are functionally alive but have an undocumented or a "don't care" operating function. Bits identified as NC must be treated in a specific manner for reads and writes. The register descriptions will provide information on how to handle NC bits.

Unless specified otherwise, all registers can be accessed as a byte, half-word, or word.

CAUTION: Some memory locations are listed as **Reserved**. These memory locations should not be accessed. Reading from these memory locations will yield invalid data. Writing to these memory locations may cause unpredictable results.

Table 2-8. Internal Register Map

Address	Register Name	Register Description	SW Lock
0x8000_xxxx	**DMA**	**DMA Control Registers**	
0x8000_0000 - 0x8000_003C	M2P Channel 0 Registers (Tx)	Memory-to-Peripheral Channel 0 Registers (Tx)	N
0x8000_0040 - 0x8000_007C	M2P Channel 1 Registers (Rx)	Memory-to-Peripheral Channel 1 Registers (Rx)	N
0x8000_0080 - 0x8000_00BC	M2P Channel 2 Registers (Tx)	Memory-to-Peripheral Channel 2 Registers (Tx)	N
0x8000_00C0 - 0x8000_00FC	M2P Channel 3 Registers (Rx)	Memory-to-Peripheral Channel 3 Registers (Rx)	N
0x8000_0100 - 0x8000_013C	M2M Channel 0 Registers	Memory-to-Memory Channel 0 Registers	N

Copyright 2007 Cirrus Logic

Table 2-8. Internal Register Map (Continued)

Address	Register Name	Register Description	SW Lock
0x8000_0140 - 0x8000_017C	M2M Channel 1 Registers	Memory-to-Memory Channel 1 Registers	N
0x8000_0180 - 0x8000_01FC		Reserved	
0x8000_0200 - 0x8000_023C	M2P Channel 5 Registers (Rx)	Memory-to-Peripheral Channel 5 Registers (Rx)	N
0x8000_0240 - 0x8000_027C	M2P Channel 4 Registers (Tx)	Memory-to-Peripheral Channel 4 Registers (Tx)	N
0x8000_0280 - 0x8000_02BC	M2P Channel 7 Registers (Rx)	Memory-to-Peripheral Channel 7 Registers (Rx)	N
0x8000_02C0 - 0x8000_02FC	M2P Channel 6 Registers (Tx)	Memory-to-Peripheral Channel 6 Registers (Tx)	N
0x8000_0300 - 0x8000_033C	M2P Channel 9 Registers (Rx)	Memory-to-Peripheral Channel 9 Registers (Rx)	N
0x8000_0340 - 0x8000_037C	M2P Channel 8 Registers (Tx)	Memory-to-Peripheral Channel 8 Registers (Tx)	N
0x8000_0380	DMAChArb	DMA Channel Arbitration Register	N
0x8000_03C0	DMAGlInt	DMA Global Interrupt Register	N
0x8000_03C4 - 0x8000_FFFC		Reserved	
0x8001_xxxx	**Ethernet MAC**	**Ethernet MAC Control Registers**	
0x8001_0000	RXCtl	MAC Receiver Control Register	N
0x8001_0004	TXCtl	MAC Transmitter Control Register	N
0x8001_0008	TestCtl	MAC Test Control Register	N
0x8001_0010	MIICmd	MAC MII Command Register	N
0x8001_0014	MIIData	MAC MII Data Register	N
0x8001_0018	MIISts	MAC MII Status Register	N
0x8001_0020	SelfCtl	MAC Self Control Register	N
0x8001_0024	IntEn	MAC Interrupt Enable Register	N
0x8001_0028	IntStsP	MAC Interrupt Status Preserve Register	N
0x8001_002C	IntStsC	MAC Interrupt Status Clear Register	N
0x8001_0030 - 0x8001_0034		Reserved	
0x8001_0038	DiagAd	MAC Diagnostic Address Register	N
0x8001_003C	DiagDa	MAC Diagnostic Data Register	N
0x8001_0040	GT	MAC General Timer Register	N
0x8001_0044	FCT	MAC Flow Control Timer Register	N
0x8001_0048	FCF	MAC Flow Control Format Register	N
0x8001_004C	AFP	MAC Address Filter Pointer Register	N
0x8001_0050 - 0x8001_0055	IndAd	MAC Individual Address Register, (shares address space with HashTbl)	N
0x8001_0050 - 0x8001_0057	HashTbl	MAC Hash Table Register, (shares address space with IndAd)	N
0x8001_0060	GlIntSts	MAC Global Interrupt Status Register	N
0x8001_0064	GlIntMsk	MAC Global Interrupt Mask Register	N
0x8001_0068	GlIntROSts	MAC Global Interrupt Read Only Status Register	N
0x8001_006C	GlIntFrc	MAC Global Interrupt Force Register	N
0x8001_0070	TXCollCnt	MAC Transmit Collision Count Register	N
0x8001_0074	RXMissCnt	MAC Receive Miss Count Register	N
0x8001_0078	RXRuntCnt	MAC Receive Runt Count Register	N
0x8001_0080	BMCtl	MAC Bus Master Control Register	N
0x8001_0084	BMSts	MAC Bus Master Status Register	N
0x8001_0088	RXBCA	MAC Receive Buffer Current Address Register	N

Copyright 2007 Cirrus Logic

Table 2-8. Internal Register Map (Continued)

Address	Register Name	Register Description	SW Lock
0x8001_0090	RXDQBAdd	MAC Receive Descriptor Queue Base Address Register	N
0x8001_0094	RXDQBLen	MAC Receive Descriptor Queue Base Length Register	N
0x8001_0096	RXDQCurLen	MAC Receive Descriptor Queue Current Length Register	N
0x8001_0098	RXDCurAdd	MAC Receive Descriptor Current Address Register	N
0x8001_009C	RXDEnq	MAC Receive Descriptor Enqueue Register	N
0x8001_00A0	RXStsQBAdd	MAC Receive Status Queue Base Address Register	N
0x8001_00A4	RXStsQBLen	MAC Receive Status Queue Base Length Register	N
0x8001_00A6	RXStsQCurLen	MAC Receive Status Queue Current Length Register	N
0x8001_00A8	RXStsQCurAdd	MAC Receive Status Queue Current Address Register	N
0x8001_00AC	RXStsEnq	MAC Receive Status Enqueue Register	N
0x8001_00B0	TXDQBAdd	MAC Transmit Descriptor Queue Base Address Register	N
0x8001_00B4	TXDQBLen	MAC Transmit Descriptor Queue Base Length Register	N
0x8001_00B6	TXDQCurLen	MAC Transmit Descriptor Queue Current Length Register	N
0x8001_00B8	TXDQCurAdd	MAC Transmit Descriptor Current Address Register	N
0x8001_00BC	TXDEnq	MAC Transmit Descriptor Enqueue Register	N
0x8001_00C0	TXStsQBAdd	MAC Transmit Status Queue Base Address Register	N
0x8001_00C4	TXStsQBLen	MAC Transmit Status Queue Base Length Register	N
0x8001_00C6	TXStsQCurLen	MAC Transmit Status Queue Current Length Register	N
0x8001_00C8	TXStsQCurAdd	MAC Transmit Status Queue Current Address Register	N
0x8001_00D0	RXBufThrshld	MAC Receive Buffer Threshold Register	N
0x8001_00D4	TXBufThrshld	MAC Transmit Buffer Threshold Register	N
0x8001_00D8	RXStsThrshld	MAC Receive Status Threshold Register	N
0x8001_00DC	TXStsThrshld	MAC Transmit Status Threshold Register	N
0x8001_00E0	RXDThrshld	MAC Receive Descriptor Threshold Register	N
0x8001_00E4	TXDThrshld	MAC Transmit Descriptor Threshold Register	N
0x8001_00E8	MaxFrmLen	MAC Maximum Frame Length Register	N
0x8001_00EC	RXHdrLen	MAC Receive Header Length Register	N
0x8001_0100 - 0x8001_010C		Reserved	
0x8001_4000 - 0x8001_50FF	MACFIFO	MAC FIFO RAM	N
0x8002_xxxx	**USB**	**USB Registers**	N
0x8002_0000	HcRevision	USB Host Controller Revision	N
0x8002_0004	HcControl	USB Host Controller Control	N
0x8002_0008	HcCommandStatus	USB Host Controller Command Status	N
0x8002_000C	HcInterruptStatus	USB Host Controller Interrupt Status	N
0x8002_0010	HcInterruptEnable	USB Host Controller Interrupt Enable	N
0x8002_0014	HcInterruptDisable	USB Host Controller Interrupt Disable	N
0x8002_0018	HcHCCA	USB Host Controller HCCA	N
0x8002_001C	HcPeriodCurrentED	USB Host Controller Period CurrentED	N
0x8002_0020	HcControlHeadED	USB Host Controller Control HeadED	N
0x8002_0024	HcControlCurrentED	USB Host Controller Control CurrentED	N
0x8002_0028	HcBulkHeadED	USB Host Controller Bulk HeadED	N
0x8002_002C	HcBulkCurrentED	USB Host Controller Bulk CurrentED	N

Copyright 2007 Cirrus Logic

Table 2-8. Internal Register Map (Continued)

Address	Register Name	Register Description	SW Lock
0x8002_0030	HcDoneHead	USB Host Controller Done Head	N
0x8002_0034	HcFmInterval	USB Host Controller Fm Interval	N
0x8002_0038	HcFmRemaining	USB Host Controller Fm Remaining	N
0x8002_003C	HcFmNumber	USB Host Controller Fm Number	N
0x8002_0040	HcPeriodicStart	USB Host Controller Periodic Start	N
0x8002_0044	HcLSThreshold	USB Host Controller LS Threshold	N
0x8002_0048	HcRhDescriptorA	USB Host Controller Root Hub Descriptor A	N
0x8002_004C	HcRhDescriptorB	USB Host Controller Root Hub Descriptor B	N
0x8002_0050	HcRhStatus	USB Host Controller Root Hub Status	N
0x8002_0054	HcRhPortStatus[1]	USB Host Controller Root Hub Port Status 1	N
0x8002_0058	HcRhPortStatus[2]	USB Host Controller Root Hub Port Status 2	N
0x8002_005C	HcRhPortStatus[3]	USB Host Controller Root Hub Port Status 3	N
0x8002_0080	USBCtrl	USB Configuration Control	N
0x8002_0084	USBHCI	USB Host Controller Interface Status	N
0x8003_xxxx	**RASTER**	**Raster Control Registers**	
0x8003_0000	VLinesTotal	Total Number of vertical frame lines	Y
0x8003_0004	VSyncStrtStop	Vertical sync pulse setup	Y
0x8003_0008	VActiveStrtStop	Vertical blanking setup	Y
0x8003_000C	VClkStrtStop	Vertical clock active frame	Y
0x8003_0010	HClkTotal	Total Number of horizontal line clocks	Y
0x8003_0014	HSyncStrtStop	Horizontal sync pulse setup	Y
0x8003_0018	HActiveStrtStop	Horizontal blanking setup	Y
0x8003_001C	HClkStrtStop	Horizontal clock active frame	Y
0x8003_0020	Brightness	PWM brightness control	N
0x8003_0024	VideoAttribs	Video state machine parameters	Y
0x8003_0028	VidScrnPage	Starting address of video screen	N
0x8003_002C	VidScrnHPage	Starting address of video screen half page	N
0x8003_0030	ScrnLines	Number of active lines scanned to the screen	N
0x8003_0034	LineLength	Length in words of data for lines	N
0x8003_0038	VLineStep	Memory step for each line	N
0x8003_003C	LineCarry	Horizontal/vertical offset parameter	Y
0x8003_0040	BlinkRate	Blink counter setup	N
0x8003_0044	BlinkMask	Logic mask applied to pixel to perform blink operation	N
0x8003_0048	BlinkPattrn	Compare value for determining blinking pixels	N
0x8003_004C	PattrnMask	Mask to limit pattern	N
0x8003_0050	BkgrndOffset	Background color or blink offset value	N
0x8003_0054	PixelMode	Pixel mode definition setup Register	N
0x8003_0058	ParlIfOut	Parallel interface write/control Register	N
0x8003_005C	ParlIfIn	Parallel interface read/setup Register	N
0x8003_0060	CursorAdrStart	Word location of the top left corner of cursor to be displayed	N
0x8003_0064	CursorAdrReset	Location of first word of cursor to be scanned after last line	N
0x8003_0068	CursorSize	Cursor height, width, and step size Register	N

Copyright 2007 Cirrus Logic

Table 2-8. Internal Register Map (Continued)

Address	Register Name	Register Description	SW Lock
0x8003_006C	CursorColor1	Cursor color overlaid when cursor value is 10	N
0x8003_0070	CursorColor2	Cursor color overlaid when cursor value is 11	N
0x8003_0074	CursorXYLoc	Cursor X and Y location Register	N
0x8003_0078	CursorDScanLHYLoc	Cursor dual scan lower half Y location Register	N
0x8003_007C	RasterSWLock	Software Lock Register. Register used to unlock registers that have SWLOCK	N
0x8003_0080 - 0x8003_00FC	GrySclLUTR	Grayscale Look Up Table	N
0x8003_0200	VidSigRsltVal	Video signature result value	N
0x8003_0204	VidSigCtrl	Video signature Control Register	N
0x8003_0208	VSigStrtStop	Vertical signature bounds setup	N
0x8003_020C	HSigStrtStop	Horizontal signature bounds setup	N
0x8003_0210	SigClrStr	Signature clear and store location	N
0x8003_0214	ACRate	LCD AC voltage bias control counter setup	N
0x8003_0218	LUTSwCtrl	LUT switching control Register	N
0x8003_021C	CursorBlinkColor1	Cursor Blink color 1	N
0x8003_0220	CursorBlinkColor2	Cursor Blink color 2	N
0x8003_0224	CursorBlinkRateCtrl	Cursor Blink rate control Register	N
0x8003_0228	VBlankStrtStop	Vertical Blank signal Start/Stop Register	N
0x8003_022C	HBlankStrtStop	Horizontal Blank signal Start/Stop Register	N
0x8003_0230	EOLOffset	End Of Line Offset value	N
0x8003_0234	FIFOLevel	FIFO refill level Register	N
0x8003_0280 - 0x8003_02FC	GrySclLUTG	Grayscale Look Up Table	N
0x8003_0300 - 0x8003_037C	GrySclLUTB	Grayscale Look Up Table	N
0x8003_0400 - 0x8003_07FC	ColorLUT	Color Look Up Table	N
0x8004_xxxx - 0x8005_xxxx		Reserved	
0x8006_xxxx	**SDRAM**	**SDRAM Registers**	N
0x8006_0000		Reserved	
0x8006_0004	GlConfig	Control and status bits used in configuration	N
0x8006_0008	RefrshTimr	Set the period between refresh cycles	N
0x8006_000C	BootSts	Reflect the state of the boot mode option pins	N
0x8006_0010	SDRAMDevCfg0	Device configuration 0	N
0x8006_0014	SDRAMDevCfg1	Device configuration 1	N
0x8006_0018	SDRAMDevCfg2	Device configuration 2	N
0x8006_001C	SDRAMDevCfg3	Device configuration 3	N
0x8008_xxxx	**SMC**	**SMC and PCMCIA Control Registers**	
0x8008_0000	SMCBCR0	Bank config Register 0 (used to program characteristics of the SRAM/ROM memory)	N
0x8008_0004	SMCBCR1	Bank config Register 1 (used to program characteristics of the SRAM/ROM memory)	N

Copyright 2007 Cirrus Logic

2

Table 2-8. Internal Register Map (Continued)

Address	Register Name	Register Description	SW Lock
0x8008_0008	SMCBCR2	Bank config Register 2 (used to program characteristics of the SRAM/ROM memory)	N
0x8008_000C	SMCBCR3	Bank config Register 3 (used to program characteristics of the SRAM/ROM memory)	N
0x8008_0010 - 0x8008_0014		Reserved	
0x8008_0018	SMCBCR6	Bank config Register 6 (used to program characteristics of the SRAM/ROM memory)	N
0x8008_001C	SMCBCR7	Bank config Register 7 (used to program characteristics of the SRAM/ROM memory)	N
0x8008_0020	PC1Attribute	PC1 Attribute Register	
0x8008_0024	PC1Common	PC1 Common Register	
0x8008_0028	PC1IO	PC1 IO Register	
0x8008_002C		Reserved	
0x8008_0030	PC2Attribute	PC2 Attribute Register	
0x8008_0034	PC2Common	PC2 Common Register	
0x8008_0038	PC2IO	PC2 IO Register	
0x8008_003C		Reserved	
0x8008_0040	PCMCIACtrl	PCMCIA Control register	
0x8008_0044 - 0x8008_FFFC		Reserved	
0x8009_xxxx	**Boot ROM**	**Boot ROM Memory Locations**	
0x8009_0000		Boot ROM Start	N
0x8009_3FFF		Boot ROM End	N
0x800A_xxxx	**IDE**	**IDE Control Registers**	
0x800A_0000	IDECtrl	IDE Control Register	N
0x800A_0004	IDECfg	IDE Configuration Register	N
0x800A_0008	IDEMDMAOp	IDE MDMA Operation Register	N
0x800A_000C	IDEUDMAOp	IDE UDMA Operation Register	N
0x800A_0010	IDEDataOut	IDE PIO Data Output Register	N
0x800A_0014	IDEDataIn	IDE PIO Data Input Register	N
0x800A_0018	IDEMDMADataOut	IDE MDMA Data Output Register	N
0x800A_001C	IDEMDMADataIn	IDE MDMA Data Input Register	N
0x800A_0020	IDEUDMADataOut	IDE UDMA Data Output Register	N
0x800A_0024	IDEUDMADataIn	IDE UDMA Data Input Register	N
0x800A_0028	IDEUDMASts	IDE UDMA Status Register	N
0x800A_002C	IDEUDMADebug	IDE UDMA Debug Register	N
0x800A_0030	IDEUDMAWrBufSts	IDE UDMA Write Buffer Status Register	N
0x800A_0034	IDEUDMARdBufSts	IDE UDMA Read Buffer Status Register	N
0x800B_xxxx	**VIC1**	**Vectored Interrupt Controller 1 Registers**	
0x800B_0000	VIC1IRQStatus	IRQ status Register	N
0x800B_0004	VIC1FIQStatus	FIQ status Register	N

Copyright 2007 Cirrus Logic

Table 2-8. Internal Register Map (Continued)

Address	Register Name	Register Description	SW Lock
0x800B_0008	VIC1RawIntr	Raw interrupt status Register	N
0x800B_000C	VIC1IntSelect	Interrupt select Register	N
0x800B_0010	VIC1IntEnable	Interrupt enable Register	N
0x800B_0014	VIC1IntEnClear	Interrupt enable clear Register	N
0x800B_0018	VIC1SoftInt	Software interrupt Register	N
0x800B_001C	VIC1SoftIntClear	Software interrupt clear Register	N
0x800B_0020	VIC1Protection	Protection enable Register	N
0x800B_0030	VIC1VectAddr	Vector address Register	N
0x800B_0034	VIC1DefVectAddr	Default vector address Register	N
0x800B_0100	VIC1VectAddr0	Vector address 0 Register	N
0x800B_0104	VIC1VectAddr1	Vector address 1 Register	N
0x800B_0108	VIC1VectAddr2	Vector address 2 Register	N
0x800B_010C	VIC1VectAddr3	Vector address 3 Register	N
0x800B_0110	VIC1VectAddr4	Vector address 4 Register	N
0x800B_0114	VIC1VectAddr5	Vector address 5 Register	N
0x800B_0118	VIC1VectAddr6	Vector address 6 Register	N
0x800B_011C	VIC1VectAddr7	Vector address 7 Register	N
0x800B_0120	VIC1VectAddr8	Vector address 8 Register	N
0x800B_0124	VIC1VectAddr9	Vector address 9 Register	N
0x800B_0128	VIC1VectAddr10	Vector address 10 Register	N
0x800B_012C	VIC1VectAddr11	Vector address 11 Register	N
0x800B_0130	VIC1VectAddr12	Vector address 12 Register	N
0x800B_0134	VIC1VectAddr13	Vector address 13 Register	N
0x800B_0138	VIC1VectAddr14	Vector address 14 Register	N
0x800B_013C	VIC1VectAddr15	Vector address 15 Register	N
0x800B_0200	VIC1VectCntl0	Vector control 0 Register	N
0x800B_0204	VIC1VectCntl1	Vector control 1 Register	N
0x800B_0208	VIC1VectCntl2	Vector control 2 Register	N
0x800B_020C	VIC1VectCntl3	Vector control3 Register	N
0x800B_0210	VIC1VectCntl4	Vector control 4 Register	N
0x800B_0214	VIC1VectCntl5	Vector control 5 Register	N
0x800B_0218	VIC1VectCntl6	Vector control 6 Register	N
0x800B_021C	VIC1VectCntl7	Vector control 7 Register	N
0x800B_0220	VIC1VectCntl8	Vector control 8 Register	N
0x800B_0224	VIC1VectCntl9	Vector control 9 Register	N
0x800B_0228	VIC1VectCntl10	Vector control 10 Register	N
0x800B_022C	VIC1VectCntl11	Vector control 11 Register	N
0x800B_0230	VIC1VectCntl12	Vector control 12 Register	N
0x800B_0234	VIC1VectCntl13	Vector control 13 Register	N
0x800B_0238	VIC1VectCntl14	Vector control 14 Register	N
0x800B_023C	VIC1VectCntl15	Vector control 15 Register	N
0x800B_0FE0	VIC1PeriphID0	VIC Identification Register bits 7:0	N
0x800B_0FE4	VIC1PeriphID1	VIC Identification Register bits 15:8	N

Copyright 2007 Cirrus Logic

Table 2-8. Internal Register Map (Continued)

Address	Register Name	Register Description	SW Lock
0x800B_0FE8	VIC1PeriphID2	VIC Identification Register bits 23:16	N
0x800B_0FEC	VIC1PeriphID3	VIC Identification Register bits 31:24	N
0x800B_0FF0 - 0x800B_0FFC		Reserved	N
0x800C_xxxx	**VIC2**	**Vectored Interrupt Controller 2 Registers**	
0x800C_0000	VIC2IRQStatus	IRQ status Register	N
0x800C_0004	VIC2FIQStatus	FIQ status Register	N
0x800C_0008	VIC2RawIntr	Raw interrupt status Register	N
0x800C_000C	VIC2IntSelect	Interrupt select Register	N
0x800C_0010	VIC2IntEnable	Interrupt enable Register	N
0x800C_0014	VIC2IntEnClear	Interrupt enable clear Register	N
0x800C_0018	VIC2SoftInt	Software interrupt Register	N
0x800C_001C	VIC2SoftIntClear	Software interrupt clear Register	N
0x800C_0020	VIC2Protection	Protection enable Register	N
0x800C_0030	VIC2VectAddr	Vector address Register	N
0x800C_0034	VIC2DefVectAddr	Default vector address Register	N
0x800C_0100	VIC2VectAddr0	Vector address 0 Register	N
0x800C_0104	VIC2VectAddr1	Vector address 1 Register	N
0x800C_0108	VIC2VectAddr2	Vector address 2 Register	N
0x800C_010C	VIC2VectAddr3	Vector address 3 Register	N
0x800C_0110	VIC2VectAddr4	Vector address 4 Register	N
0x800C_0114	VIC2VectAddr5	Vector address 5 Register	N
0x800C_0118	VIC2VectAddr6	Vector address 6 Register	N
0x800C_011C	VIC2VectAddr7	Vector address 7 Register	N
0x800C_0120	VIC2VectAddr8	Vector address 8 Register	N
0x800C_0124	VIC2VectAddr9	Vector address 9 Register	N
0x800C_0128	VIC2VectAddr10	Vector address 10 Register	N
0x800C_012C	VIC2VectAddr11	Vector address 11 Register	N
0x800C_0130	VIC2VectAddr12	Vector address 12 Register	N
0x800C_0134	VIC2VectAddr13	Vector address 13 Register	N
0x800C_0138	VIC2VectAddr14	Vector address 14 Register	N
0x800C_013C	VIC2VectAddr15	Vector address 15 Register	N
0x800C_0200	VIC2VectCntl0	Vector control 0 Register	N
0x800C_0204	VIC2VectCntl1	Vector control 1 Register	N
0x800C_0208	VIC2VectCntl2	Vector control 2 Register	N
0x800C_020C	VIC2VectCntl3	Vector control3 Register	N
0x800C_0210	VIC2VectCntl4	Vector control 4 Register	N
0x800C_0214	VIC2VectCntl5	Vector control 5 Register	N
0x800C_0218	VIC2VectCntl6	Vector control 6 Register	N
0x800C_021C	VIC2VectCntl7	Vector control 7 Register	N
0x800C_0220	VIC2VectCntl8	Vector control 8 Register	N
0x800C_0224	VIC2VectCntl9	Vector control 9 Register	N
0x800C_0228	VIC2VectCntl10	Vector control 10 Register	N

Copyright 2007 Cirrus Logic

Table 2-8. Internal Register Map (Continued)

Address	Register Name	Register Description	SW Lock
0x800C_022C	VIC2VectCntl11	Vector control 11 Register	N
0x800C_0230	VIC2VectCntl12	Vector control 12 Register	N
0x800C_0234	VIC2VectCntl13	Vector control 13 Register	N
0x800C_0238	VIC2VectCntl14	Vector control 14 Register	N
0x800C_023C	VIC2VectCntl15	Vector control 15 Register	N
0x800C_0FE0	VIC2PeriphID0	VIC Identification Register bits 7:0	N
0x800C_0FE4	VIC2PeriphID1	VIC Identification Register bits 15:8	N
0x800C_0FE8	VIC2PeriphID2	VIC Identification Register bits 23:16	N
0x800C_0FEC	VIC2PeriphID3	VIC Identification Register bits 31:24	N
0x800C_0FF0 - 0x800C_0FFC		Reserved	N
0x8081_xxxx	**TIMER**	**Timer Registers**	
0x8081_0000	Timer1Load	Contains the initial value of the timer	N
0x8081_0004	Timer1Value	Gives the current value of the timer	N
0x8081_0008	Timer1Control	Provides enable/disable and mode configurations for the timer	N
0x8081_000C	Timer1Clear	Clears an interrupt generated by the timer	N
0x8081_0020	Timer2Load	Contains the initial value of the timer	N
0x8081_0024	Timer2Value	Gives the current value of the timer	N
0x8081_0028	Timer2Control	Provides enable/disable and mode configurations for the timer	N
0x8081_002C	Timer2Clear	Clears an interrupt generated by the timer	N
0x8081_0060 - 0x8081_0064		Reserved	
0x8081_0080	Timer3Load	Contains the initial value of the timer	N
0x8081_0084	Timer3Value	Gives the current value of the timer	N
0x8081_0088	Timer3Control	Provides enable/disable and mode configurations for the timer	N
0x8081_008C	Timer3Clear	Clears an interrupt generated by the timer	N
0x8082_xxxx	**I2S**	**I2S Registers**	N
0x8082_0000	I2STXClkCfg	Transmitter clock configuration Register	N
0x8082_0004	I2SRXClkCfg	Receiver clock configuration Register	N
0x8082_0008	I2SGISts	I2S Global Status Register. This reflects the status of the 3 RX FIFOs and the 3 TX FIFOs	N
0x8082_000C	I2SGICtrl	I2S Global Control Register	N
0x8082_0010	I2STX0Lft	Left Transmit data Register for channel 0	N
0x8082_0014	I2STX0Rt	Right Transmit data Register for channel 0	N
0x8082_0018	I2STX1Lft	Left Transmit data Register for channel 1	N
0x8082_001C	I2STX1Rt	Right Transmit data Register for channel 1	N
0x8082_0020	I2STX2Lft	Left Transmit data Register for channel 2	N
0x8082_0024	I2STX2Rt	Right Transmit data Register for channel 2	N
0x8082_0028	I2STXLinCtrlData	Transmit Line Control Register	N
0x8082_002C	I2STXCtrl	Transmit Control Register	N
0x8082_0030	I2STXWrdLen	Transmit Word Length	N
0x8082_0034	I2STX0En	TX0 Channel Enable	N
0x8082_0038	I2STX1En	TX1 Channel Enable	N

Copyright 2007 Cirrus Logic

Table 2-8. Internal Register Map (Continued)

Address	Register Name	Register Description	SW Lock
0x8082_003C	I2STX2En	TX2 Channel Enable	N
0x8082_0040	I2SRX0Lft	Left Receive data Register for channel 0	N
0x8082_0044	I2SRX0Rt	Right Receive data Register for channel 0	N
0x8082_0048	I2SRX1Lft	Left Receive data Register for channel 1	N
0x8082_004C	I2SRX1Rt	Right Receive data Register for channel 1	N
0x8082_0050	I2SRX2Lft	Left Receive data Register for channel 2	N
0x8082_0054	I2SRX2Rt	Right Receive data Register for channel 2	N
0x8082_0058	I2SRXLinCtrlData	Receive Line Control Register	N
0x8082_005C	I2SRXCtrl	Receive Control Register	N
0x8082_0060	I2SRXWrdLen	Receive Word Length	N
0x8082_0064	I2SRX0En	RX0 Channel Enable	N
0x8082_0068	I2SRX1En	RX1 Channel Enable	N
0x8082_006C	I2SRX2En	RX2 Channel Enable	N
0x8083_xxxx	**SECURITY**	**Security Registers**	
0x8083_2714	ExtensionID	Contains the Part ID for EP93XX devices	N
Contact Cirrus Logic for details regarding implementation of device Security measures.			
0x8084_xxxx	**GPIO**	**GPIO Control Registers**	
0x8084_0000	PADR	GPIO Port A Data Register	N
0x8084_0004	PBDR	GPIO Port B Data Register	N
0x8084_0008	PCDR	GPIO Port C Data Register	N
0x8084_000C	PDDR	GPIO Port D Data Register	N
0x8084_0010	PADDR	GPIO Port A Data Direction Register	N
0x8084_0014	PBDDR	GPIO Port B Data Direction Register	N
0x8084_0018	PCDDR	GPIO Port C Data Direction Register	N
0x8084_001C	PDDDR	GPIO Port D Data Direction Register	N
0x8084_0020	PEDR	GPIO Port E Data Register	N
0x8084_0024	PEDDR	GPIO Port E Data Direction Register	N
0x8084_0028 - 0x8084_002C		Reserved	
0x8084_0030	PFDR	GPIO Port F Data Register	N
0x8084_0034	PFDDR	GPIO Port F Data Direction Register	N
0x8084_0038	PGDR	GPIO Port G Data Register	N
0x8084_003C	PGDDR	GPIO Port G Data Direction Register	N
0x8084_0040	PHDR	GPIO Port H Data Register	N
0x8084_0044	PHDDR	GPIO Port H Data Direction Register	N
0x8084_0048		Reserved	
0x8084_004C	GPIOFIntType1	Register controlling type, level or edge, of interrupt generated by the pins of Port F	N
0x8084_0050	GPIOFIntType2	Register controlling polarity, high/low or rising/falling, of interrupt generated by Port F	N
0x8084_0054	GPIOFEOI	GPIO Port F End Of Interrupt Register	N
0x8084_0058	GPIOFIntEn	Interrupt Enable for Port F	N

Copyright 2007 Cirrus Logic

Table 2-8. Internal Register Map (Continued)

Address	Register Name	Register Description	SW Lock
0x8084_005C	IntStsF	GPIO Interrupt Status Register. Contains status of Port F interrupts after masking.	N
0x8084_0060	RawIntStsF	Raw Interrupt Status Register. Contains raw interrupt status of Port F before masking.	N
0x8084_0064	GPIOFDB	GPIO F Debounce Register	N
0x8084_0068 - 0x8084_008C		Reserved	
0x8084_0090	GPIOAIntType1	Register controlling type, level or edge, of interrupt generated by the pins of Port A	N
0x8084_0094	GPIOAIntType2	Register controlling polarity, high/low or rising/falling, of interrupt generated by Port A	N
0x8084_0098	GPIOAEOI	GPIO Port A End Of Interrupt Register	N
0x8084_009C	GPIOAIntEn	Controlling the generation of interrupts by the pins of Port A	N
0x8084_00A0	IntStsA	GPIO Interrupt Status Register. Contains status of Port A interrupts after masking.	N
0x8084_00A4	RawIntStsA	Raw Interrupt Status Register. Contains raw interrupt status of Port A before masking.	N
0x8084_00A8	GPIOADB	GPIO A Debounce Register	N
0x8084_00AC	GPIOBIntType1	Register controlling type, level or edge, of interrupt generated by the pins of Port B	N
0x8084_00B0	GPIOBIntType2	Register controlling polarity, high/low or rising/falling, of interrupt generated by Port B	N
0x8084_00B4	GPIOBEOI	GPIO Port B End Of Interrupt Register	N
0x8084_00B8	GPIOBIntEn	Controlling the generation of interrupts by the pins of Port B	N
0x8084_00BC	IntStsB	GPIO Interrupt Status Register. Contains status of Port B interrupts after masking.	N
0x8084_00C0	RawIntStsB	Raw Interrupt Status Register. Contains raw interrupt status of Port B before masking.	N
0x8084_00C4	GPIOBDB	GPIO B Debounce Register	N
0x8084_00C8	EEDrive	EEPROM pin drive type control. Defines the driver type for the EECLK and EEDAT pins	N
0x8088_xxxx	**AC'97**	**AC'97 Control Registers**	
0x8088_0000	AC97DR1	Data read or written from/to FIFO1	N
0x8088_0004	AC97RXCR1	Control Register for receive	N
0x8088_0008	AC97TXCR1	Control Register for transmit	N
0x8088_000C	AC97SR1	Status Register	N
0x8088_0010	AC97RISR1	Raw interrupt status Register	N
0x8088_0014	AC97ISR1	Interrupt Status	N
0x8088_0018	AC97IE1	Interrupt Enable	N
0x8088_001C		Reserved	
0x8088_0020	AC97DR2	Data read or written from/to FIFO2	N
0x8088_0024	AC97RXCR2	Control Register for receive	N
0x8088_0028	AC97TXCR2	Control Register for transmit	N
0x8088_002C	AC97SR2	Status Register	N
0x8088_0030	AC97RISR2	Raw interrupt status Register	N

Copyright 2007 Cirrus Logic

Table 2-8. Internal Register Map (Continued)

Address	Register Name	Register Description	SW Lock
0x8088_0034	AC97ISR2	Interrupt Status	N
0x8088_0038	AC97IE2	Interrupt Enable	N
0x8088_003C		Reserved	
0x8088_0040	AC97DR3	Data read or written from/to FIFO3	N
0x8088_0044	AC97RXCR3	Control Register for receive	N
0x8088_0048	AC97TXCR3	Control Register for transmit	N
0x8088_004C	AC97SR3	Status Register	N
0x8088_0050	AC97RISR3	Raw interrupt status Register	N
0x8088_0054	AC97ISR3	Interrupt Status	N
0x8088_0058	AC97IE3	Interrupt Enable	N
0x8088_005C		Reserved	
0x8088_0060	AC97DR4	Data read or written from/to FIFO4	N
0x8088_0064	AC97RXCR4	Control Register for receive	N
0x8088_0068	AC97TXCR4	Control Register for transmit	N
0x8088_006C	AC97SR4	Status Register	N
0x8088_0070	AC97RISR4	Raw interrupt status Register	N
0x8088_0074	AC97ISR4	Interrupt Status	N
0x8088_0078	AC97IE4	Interrupt Enable	N
0x8088_007C		Reserved	
0x8088_0080	AC97S1Data	Data received/transmitted on SLOT1	N
0x8088_0084	AC97S2Data	Data received/transmitted on SLOT2	N
0x8088_0088	AC97S12Data	Data received/transmitted on SLOT12	N
0x8088_008C	AC97RGIS	Raw Global interrupt status Register	N
0x8088_0090	AC97GIS	Global interrupt status Register	N
0x8088_0094	AC97IM	Interrupt mask Register	N
0x8088_0098	AC97EOI	End Of Interrupt Register	N
0x8088_009C	AC97GCR	Main Control Register	N
0x8088_00A0	AC97Reset	RESET control Register	N
0x8088_00A4	AC97SYNC	SYNC control Register	N
0x8088_00A8	AC97GCIS	Global channel FIFO interrupt status Register	N
0x808A_xxxx	**SPI**	**SPI Control Registers**	
0x808A_0000	SSP1CR0	SPI1 Control Register 0	N
0x808A_0004	SSP1CR1	SPI1 Control Register 1	N
0x808A_0008	SSP1DR	SPI1 Data Register	N
0x808A_000C	SSP1SR	SPI1 Status Register	N
0x808A_0010	SSP1CPSR	SPI1 Clock Prescale Register	N
0x808A_0014	SSP1IIR	SPI1 Interrupt/Interrupt Clear Register	N
0x808B_xxxx	**IrDA**	**IrDA Control Registers**	
0x808B_0000	IrEnable	IrDA Interface Enable	N
0x808B_0004	IrCtrl	IrDA Control Register	N

Copyright 2007 Cirrus Logic

Table 2-8. Internal Register Map (Continued)

Address	Register Name	Register Description	SW Lock
0x808B_0008	IrAdrMatchVal	IrDA Address Match Value Register	N
0x808B_000C	IrFlag	IrDA Flag Register	N
0x808B_0010	IrData	IrDA Transmit and Receive FIFOs	N
0x808B_0014	IrDataTail	IrDA Data Tail Register	N
0x808B_0018 - 0x808B_001C		Reserved	
0x808B_0020	IrRIB	IrDA Receive Information Buffer	N
0x808B_0024	IrTR0	IrDA Test Register, Received byte count	N
0x808B_0088	MIIR	IrDA MIR Interrupt Register	N
0x808B_008C - 0x808B_018C		Reserved	
0x808C_xxxx	**UART1**	**UART1 Control Registers**	
0x808C_0000	UART1Data	UART1 Data Register	N
0x808C_0004	UART1RXSts	UART1 Receive Status Register	N
0x808C_0008	UART1LinCtrlHigh	UART1 Line Control Register - High Byte	N
0x808C_000C	UART1LinCtrlMid	UART1 Line Control Register - Middle Byte	N
0x808C_0010	UART1LinCtrlLow	UART1 Line Control Register - Low Byte	N
0x808C_0014	UART1Ctrl	UART1 Control Register	N
0x808C_0018	UART1Flag	UART1 Flag Register	N
0x808C_001C	UART1IntIDIntClr	UART1 Interrupt ID and Interrupt Clear Register	N
0x808C_0020		Reserved	
0x808C_0028	UART1DMACtrl	UART1 DMA Control Register	N
0x808C_0100	UART1ModemCtrl	UART1 Modem Control Register	N
0x808C_0104	UART1ModemSts	UART1 Modem Status Register	N
0x808C_0114 - 0x808C_0208		Reserved	
0x808C_020C	UART1HDLCCtrl	UART1 HDLC Control Register	N
0x808C_0210	UART1HDLCAddMtchVal	UART1 HDLC Address Match Value	N
0x808C_0214	UART1HDLCAddMask	UART1 HDLC Address Mask	N
0x808C_0218	UART1HDLCRXInfoBuf	UART1 HDLC Receive Information Buffer	N
0x808C_021C	UART1HDLCSts	UART1 HDLC Status Register	N
0x808D_xxxx	**UART2**	**UART2 Control Registers**	
0x808D_0000	UART2Data	UART2 Data Register	N
0x808D_0004	UART2RXSts	UART2 Receive Status Register	N
0x808D_0008	UART2LinCtrlHigh	UART2 Line Control Register - High Byte	N
0x808D_000C	UART2LinCtrlMid	UART2 Line Control Register - Middle Byte	N
0x808D_0010	UART2LinCtrlLow	UART2 Line Control Register - Low Byte	N
0x808D_0014	UART2Ctrl	UART2 Control Register	N
0x808D_0018	UART2Flag	UART2 Flag Register	N
0x808D_001C	UART2IntIDIntClr	UART2 Interrupt ID and Interrupt Clear Register	N
0x808D_0020	UART2IrLowPwrCntr	UART2 IrDA Low-power Counter Register	N
0x808D_0028	UART2DMACtrl	UART2 DMA Control Register	N

Copyright 2007 Cirrus Logic

Table 2-8. Internal Register Map (Continued)

Address	Register Name	Register Description	SW Lock
0x808E_xxxx	**UART3**	**UART3 Control Registers**	
0x808E_0000	UART3Data	UART3 Data Register	N
0x808E_0004	UART3RXSts	UART3 Receive Status Register	N
0x808E_0008	UART3LinCtrlHigh	UART3 Line Control Register - High Byte	N
0x808E_000C	UART3LinCtrlMid	UART3 Line Control Register - Middle Byte	N
0x808E_0010	UART3LinCtrlLow	UART3 Line Control Register - Low Byte	N
0x808E_0014	UART3Ctrl	UART3 Control Register	N
0x808E_0018	UART3Flag	UART3 Flag Register	N
0x808E_001C	UART3IntIDIntClr	UART3 Interrupt ID and Interrupt Clear Register	N
0x808E_0020	UART3IrLowPwrCntr	UART3 IrDA Low-power Counter Register	N
0x808E_0028	UART3DMACtrl	UART3 DMA Control Register	N
0x808E_0100	UART3ModemCtrl	UART3 Modem Control Register	N
0x808E_0104	UART3ModemSts	UART3 Modem Status Register	N
0x808E_0108	UART3ModemTstCtrl	UART3 Modem Support Test Control Register	N
0x808E_0114 - 0x808E_0208		Reserved	
0x808E_020C	UART3HDLCCtrl	UART3 HDLC Control Register	N
0x808E_0210	UART3HDLCAddMtchVal	UART3 HDLC Address Match Value	N
0x808E_0214	UART3HDLCAddMask	UART3 HDLC Address Mask	N
0x808E_0218	UART3HDLCRXInfoBuf	UART3 HDLC Receive Information Buffer	N
0x808E_021C	UART3HDLCSts	UART3 HDLC Status Register	N
0x808F_xxxx	**KEY**	**Key Matrix Control Registers**	
0x808F_0000	KeyScanInit	Key Matrix Scan Initialize	N
0x808F_0004	KeyDiagnostic	Key Matrix Diagnostic	N
0x808F_0008	KeyRegister	Key Matrix Key Register	N
0x8090_xxxx	**TOUCH**	**Touchscreen Control Registers**	
0x8090_0000	TSSetup	Touchscreen Setup Register	N
0x8090_0004	TSXYMaxMin	Touchscreen X/Y Max Min Register	N
0x8090_0008	TSXYResult	Touchscreen X/Y Result Register	N
0x8090_000C	TSDischarge	Touchscreen Switch Matrix Discharge Control Register	Y
0x8090_0010	TSXSample	Touchscreen Switch Matrix X-Sample Control Register	Y
0x8090_0014	TSYSample	Touchscreen Switch Matrix Y-Sample Control Register	Y
0x8090_0018	TSDirect	Touchscreen Switch Matrix Direct Control Register	Y
0x8090_001C	TSDetect	Touchscreen Direct Control Touch Detect Register	N
0x8090_0020	TSSWLock	Touchscreen Software Lock Register	N
0x8090_0024	TSSetup2	Touchscreen Setup Register 2	N
0x8091_xxxx	**PWM**	**PWM Control Registers**	
0x8091_0000	PWM0TermCnt	PWM0 Terminal Count	N
0x8091_0004	PWM0DutyCycle	PWM0 Duty Cycle	N
0x8091_0008	PWM0En	PWM0 Enable	N

Copyright 2007 Cirrus Logic

Table 2-8. Internal Register Map (Continued)

Address	Register Name	Register Description	SW Lock
0x8091_000C	PWM0Invert	PWM0 Invert	N
0x8091_0010	PWM0Sync	PWM0 Synchronous	N
0x8091_0020	PWM1_TC	PWM1 Terminal Count	N
0x8091_0024	PWM1_DC	PWM1 Duty Cycle	N
0x8091_0028	PWM1_EN	PWM1 Enable	N
0x8091_002C	PWM1_INV	PWM1 Invert	N
0x8091_0030	PWM1_SYNC	PWM1 Synchronous	N
0x8092_xxxx	**RTC**	**RTC Control Registers**	
0x8092_0000	RTCData	RTC Data Register	N
0x8092_0004	RTCMatch	RTC Match Register	N
0x8092_0008	RTCSts	RTC Status/EOI Register	N
0x8092_000C	RTCLoad	RTC Load Register	N
0x8092_0010	RTCCtrl	RTC Control Register	N
0x8092_0098	RTCSWComp	RTC Software Compensation	N
0x8093_xxxx	**Syscon**	**System Control Registers**	
0x8093_0000	PwrSts	Power/state control state	N
0x8093_0004	PwrCnt	Clock/debug control status	N
0x8093_0008	Halt	Enter IDLE mode	N
0x8093_000C	Stby	Enter Standby mode	N
0x8093_0018	TEOI	Write to clear Watchdog interrupt	N
0x8093_001C	STFClr	Write to clear Nbflg, rstflg, pfflg and cldflg	N
0x8093_0020	ClkSet1	Clock speed control 1	N
0x8093_0024	ClkSet2	Clock speed control 2	N
0x8093_0040	ScratchReg0	Scratch Register 0	N
0x8093_0044	ScratchReg1	Scratch Register 1	N
0x8093_0050	APBWait	APB wait	N
0x8093_0054	BusMstrArb	Bus Master Arbitration	N
0x8093_0058	BootModeClr	Boot Mode Clear Register	N
0x8093_0080	DeviceCfg	Device configuration	Y
0x8093_0084	VidClkDiv	Video Clock Divider	Y
0x8093_0088	MIRClkDiv	MIR Clock Divider. Configures video clock for the raster engine.	Y
0x8093_008C	I2SClkDiv	I2S Audio Clock Divider	
0x8093_0090	KeyTchClkDiv	Keyscan/Touch Clock Divider	Y
0x8093_0094	ChipID	Chip ID Register	Y
0x8093_009C	SysCfg	System Configuration	Y
0x8093_00C0	SysSWLock	Syscon Software Lock Register	N
0x8094_xxxx	**WATCHDOG**	**Watchdog Control Register**	N
0x8094_0000	Watchdog	Watchdog Timer Register	N
0x8094_0004	WDStatus	Watchdog Status Register	N

Copyright 2007 Cirrus Logic

Table 2-8. Internal Register Map (Continued)

Address	Register Name	Register Description	SW Lock
0x8095_0000 - 0x8FFF_FFFF		Reserved	

Copyright 2007 Cirrus Logic

Chapter 3
MaverickCrunch Co-Processor 3

3.1 Introduction

Note:This chapter applies only to the EP9302, EP9307, EP9312, and EP9315 processors.

The MaverickCrunch co-processor accelerates IEEE-754 floating point arithmetic and 32-bit and 64-bit fixed point arithmetic operations. It provides an integer multiply-accumulate (MAC) that is considerably faster than the native MAC implementation in the ARM920T. The MaverickCrunch co-processor significantly accelerates the arithmetic processing required to encode/decode digital audio formats.

The MaverickCrunch co-processor uses the standard ARM920T co-processor interface, sharing its memory interface and instruction stream. All MaverickCrunch operations are simply ARM920T co-processor instructions. The co-processor handles all internal inter-instruction dependencies by using internal data forwarding and inserting wait states.

3.1.1 Features

Key features include:

- IEEE-754 single and double precision floating point
- 32/64-bit integer
- Add/multiply/compare
- Integer Multiply-Accumulate (MAC) 32-bit input with 72-bit accumulate
- Integer Shifts
- Floating point to/from integer conversion
- Sixteen 64-bit registers
- Four 72-bit accumulators

3.1.2 Operational Overview

The MaverickCrunch co-processor is a true ARM920T co-processor. It communicates with the ARM920T via the co-processor bus and shares the instruction stream and memory interface of the ARM920T. It runs at the ARM920T core clock frequency (either FCLK or BCLK).

The co-processor supports four primary data formats:

Copyright 2007 Cirrus Logic

- IEEE-754 single precision floating point (24-bit signed significand and 8-bit biased exponent)
- IEEE-754 double precision floating point (53-bit signed significand and 11-bit biased exponent)
- 32-bit integer
- 64-bit integer

The co-processor performs the following standard operations on all four supported data formats:

- addition
- subtraction
- multiplication
- absolute value
- negation
- logical left/right shift
- comparison

In addition, for 32-bit integers, the co-processor provides:

- multiply-accumulate (MAC)
- multiply-subtract (MSB)

Any of the four data formats may be converted to another of the formats. All four data types may be loaded directly from and stored directly to memory via the ARM920T co-processor interface. They may also be moved to or from ARM920T registers.

The MaverickCrunch co-processor also provides a 72-bit extended precision integer format that is used only in the accumulators. The accumulators may also be used in MAC and MSB operations.

IEEE-754 rounding and exceptions are also provided. Four rounding modes for floating point operations are:

- round to nearest
- round toward $+\infty$
- round toward $-\infty$
- round toward 0

Exceptions include:

- Invalid operator
- Overflow
- Underflow

Copyright 2007 Cirrus Logic

- Inexact

Note that the division by zero exception is not supported as the MaverickCrunch co-processor does not provide division or square root.

3.1.3 Pipelines and Latency

There are two primary pipelines within the MaverickCrunch co-processor. One handles all communication with the ARM920T, while the other, the "data path" pipeline, handles all arithmetic operations (this one actually operates at one half the MaverickCrunch co-processor clock frequency).

The data path pipeline may run synchronously or asynchronously with respect to the ARM instruction pipeline. If run asynchronously, data path computation is decoupled from the ARM, allowing high throughput, though arithmetic exceptions are not synchronous. If run synchronously, exceptions are synchronous, but throughput suffers.

Assuming no inter-instruction dependencies causing pipeline stalls, arithmetic instructions can produce a new result every two ARM920T clocks, which is a maximum throughput of one data path instruction per eight ARM920T clocks. The only exception is 64-bit multiplies (CFMULD or CFMUL64), which require six extra ARM920T clocks to produce their result, which is maximum throughput of eight ARM920T clocks per instruction.

The normal latency for an arithmetic instruction is approximately nine ARM920T clocks, from initial decode to the time the result is written to the register file. A 64-bit multiply requires 15 clocks.

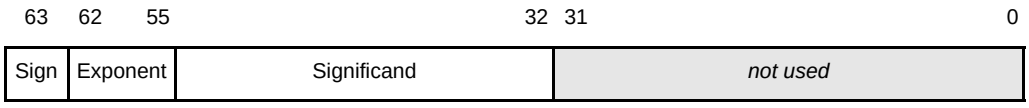
3.1.4 Data Registers

The MaverickCrunch co-processor contains these registers:

- Sixteen 64-bit general purpose registers, c0 through c15
- Four 72-bit accumulators, a0 through a3
- One status and control register, DSPSC

A single precision floating point value is stored in the upper 32 bits of a 64-bit register and must be explicitly promoted to double precision to be used in double precision calculations:

Opcode

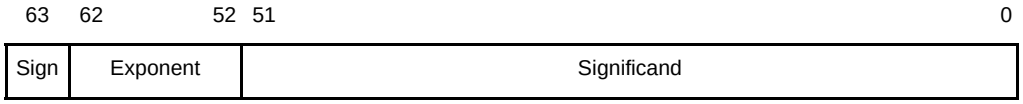


Copyright 2007 Cirrus Logic

3

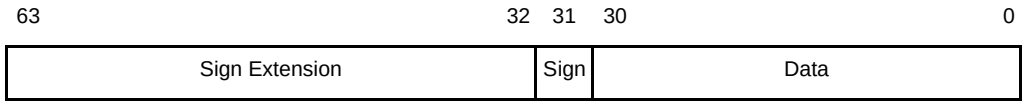
A double precision value requires all 64 bits:

Opcode



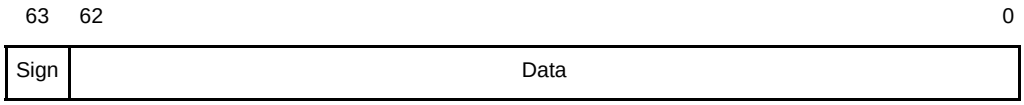
A 32-bit integer is stored in the lower 32 bits of a 64-bit register and sign-extended when written, provided the UI bit in the DSPSC is clear:

Opcode



Hence, 32-bit integers may be used directly in calculations with 64-bit integers, which are stored as:

Opcode



3.1.5 Integer Saturation Arithmetic

By default, the co-processor treats all 32-bit and 64-bit integers as signed values and automatically saturates the results of most integer operations and all conversions from floating-point to integer format. Instructions that may saturate their results are:

- CFADD32 and CFADD64
- CFSUB32 and CFSUB64
- CFMUL32 and CFMUL64
- CFMAC32 and CFMSC32
- CFCVTS32 and CFCVTD32
- CFTRUNCS32 and CFTRUNCD32

This behavior, however, can be altered by setting the UI bit and the ISAT bit in the DSPSC. With the UI bit clear (the default), 32-bit and 64-bit integer operations are treated as signed with respect to overflow and underflow detection and saturation as well as compare operations. Setting the UI bit causes the MaverickCrunch co-processor to treat all 32-bit and 64-bit integer operations as unsigned with respect to overflow, underflow, saturation, and comparison.

Copyright 2007 Cirrus Logic

With saturation enabled (the default), the maximum representable value is returned on overflow and the minimum representable value is returned on underflow. The maximum and minimum values depends on the operand size and whether the UI bit in the DSPSC is set, as shown in Table 3-1.

Table 3-1. Saturation for Non-accumulator Instructions

Overflow	**Signed**	32-bit	0x7FFF_FFFF
		64-bit	0x7FFF_FFFF_FFFF_FFFF
	Unsigned	32-bit	0xFFFF_FFFF
		64-bit	0xFFFF_FFFF_FFFF_FFFF
Underflow	**Signed**	32-bit	0x8000_0000
		64-bit	0x8000_0000_0000_0000
	Unsigned	32-bit	0x0000_0000
		64-bit	0x0000_0000_0000_0000

To disable saturation on overflow and underflow, set the ISAT bit in the DSPSC.

Normally, arithmetic instructions that write to an accumulator do not saturate their results on overflow or underflow. These instructions are:

- CFMADD32 and CFMSUB32
- CFMADDA32 and CFMSUBA32

However, the SAT[1:0] bits in the DSPSC may be set to select one of several kinds of saturation to occur on the results of these instructions before they are written to an accumulator.

Note: This action does not affect the operation of instructions that do not write their result to an accumulator.

Enabling saturation also modifies the representation of data stored in the accumulator. The three supported bit formats and their maximum and minimum saturation values are shown in Table 3-2.

Table 3-2. Accumulator Bit Formats for Saturation

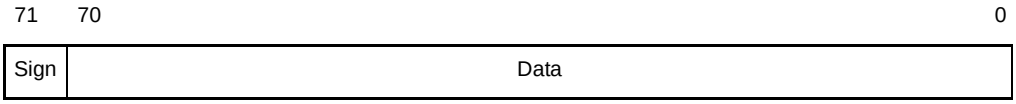
Bit Format	Maximum Value (hex)	Minimum Value (hex)
2.62	64 bits - 0x3FFF FFFF FFFF FFFF	64 bits - 0xC000 0000 0000 0000
1.63	64 bits - 0x7FFF FFFF FFFF FFFF	64 bits - 0x8000 0000 0000 0000
1.31	32 bits - 0x7FFF FFFF	32 bits - 0x8000 0000

The bit format x.yy represents x binary bits before the decimal point and yy fraction bits after the decimal point, as for example, when the bit format 2.62 has two binary bits and sixty-two fraction bits. Though these formats utilize either 32- or 64-bit integers, the accumulators are

Copyright 2007 Cirrus Logic

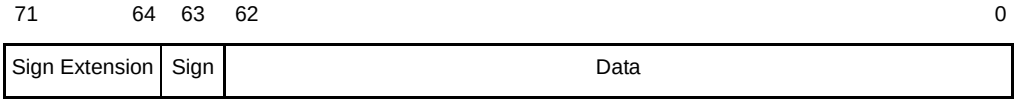
72 bits wide. If the accumulator saturation mode is disabled (the default), the accumulator bit fields are assigned as below for a 2's complement integer.

Opcode



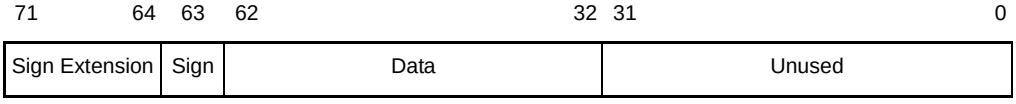
If the saturation mode 1.63 is selected, the bit field assignments are:

Opcode



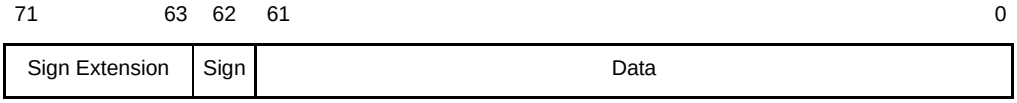
If the saturation mode 1.31 is selected, the bit field assignments are:

Opcode



If the saturation mode 2.62 is selected, the bit field assignments are:

Opcode



3.1.6 Comparisons

The Crunch co-processor provides four compare operations:

- CFCMP32 - 32-bit integer
- CFCMP64 - 64-bit integer
- CFCMPS - single floating point
- CFCMPD - double floating point

The DSPSC register bit UINT affects the operation of integer comparisons. If clear, integers are treated as signed values, and if set, they are treated as unsigned. DSPSC.UINT has no effect on floating point comparisons.

All compare operations update both the FCC[1:0] bits in the DSPSC register and an ARM register. Though any of the ARM general purpose registers r0 through r14 may be specified as the destination, specifying r15 actually updates the CPSR flag bits NZCV. This permits the

Copyright 2007 Cirrus Logic

condition code field of any subsequent ARM instruction to gate the execution of that instruction based on the result of a Crunch compare operation.

Table 3-3 illustrates the legal relationships and, for each one, the values written to the FCC bits and the NZCV flags. The FCC bits and the NZCV flags provide the same information, but in different ways and in different places. Their values depend only on the relationship between the operands, regardless of whether the operands are considered signed integer, unsigned integer, or floating point. The unordered relationship can only apply to floating point operands.

Table 3-3. Comparison Relationships and Their Results

Relationship	FCC[1:0]	NCZV
A = B	00	0100
A < B	01	1000
A > B	10	1001
Unordered	11	0000

The NZCV flags are not computed exactly as with integer comparisons using the ARM CMP instruction. Hence, when examining the result of Crunch comparisons, the condition codes field of ARM instructions should be interpreted differently, as shown in Table 3-4. The same six condition codes should be used whether the comparison operands were signed integers, unsigned integers, or floating point. No other condition codes are meaningful.

Table 3-4. ARM® Condition Codes and Crunch Compare Results

Condition Code		Relationship	ARM Meaning	Crunch Meaning
Opcode[31:28]	Mnemonic			
0000	EQ	A = B	Equal	Equal
0001	NE	A ≠ B	Not Equal	Not Equal
1010	GE	A ≥ B	Signed Greater Than or Equal	Greater Than or Equal
1011	LT	A < B	Signed Less Than	Less Than
1100	GT	A > B	Signed Greater Than	Greater Than
1101	LE	A ≤ B	Signed Less Than or Equal	Less Than or Equal
1110	AL	N/A	Always (unconditional)	Always (unconditional)
1111	NV	N/A	Never	Never

Copyright 2007 Cirrus Logic

3

3.2 Programming Examples

The examples below show two algorithms, each implemented using the standard programming languages and the MaverickCrunch instruction set.

3.2.1 Example 1

Section 3.2.1.2, Section 3.2.1.3, and Section 3.2.1.4 show three coding samples performing the same operation. Section 3.2.1.1 shows common setup code used by all three samples. Section 3.2.1.2 shows the program implemented in C code. Section 3.2.1.3 uses ARM assembly language, accessing the MaverickCrunch with ARM co-processor instructions. Section 3.2.1.4 uses MaverickCrunch assembly language instructions.

3.2.1.1 Setup Code

```
ldr     r0,    =80930000              ; Syscon base address
mov  r1,    #0xaa                     ; SW lock key
str     r1,    [r0, #0xc0]             ; unlock by writing key to SysSWLock
register
ldr     r1,    [r0, #0x80]            ; Turn on CPENA bit in DEVCFG register
to
orr     r1,    r1, #0x00800000     ; enable MaverickCrunch co-processor
str     r1,    [r0, #0x80]               ;
```

3.2.1.2 C Code

```
int num = 0;

for(num=0; num < 10; num++)
 num = num * 5;
```

3.2.1.3 Accessing MaverickCrunch with ARM Co-Processor Instructions

```
   ldc p5, c0, [r0, #0x0]       ; data section preloaded with 0x0 ("num")
   ldc p5, c1, [r0, #0x4]       ; data section preloaded with 0xa
   ldc p5, c2, [r0, #0x8]       ; data section preloaded with 0x1
   ldc p5, c3, [r0, #0xc]       ; data section preloaded with 0x5
 loop
   cdp p5, 1, c0, c0, c3, 0    ; c0 <= c0 * 5
   cdp p5, 3, c0, c0, c2, 6    ; c0 <= c0 - 1
   mrc p5, 0, r15 c0, c1, 4    ; c0 < 10 ?
   blt loop                            ;   yes
   stc p5, c0, [r0, #0x0]       ;   no, store result
```

3.2.1.4 MaverickCrunch Assembly Language Instructions

```
   cfldr32 c0, [r0, #0x0]       ; data section preloaded with 0x0 ("num")
   cfldr32 c1, [r0, #0x4]       ; data section preloaded with 0xa
   cfldr32 c2, [r0, #0x8]       ; data section preloaded with 0x1
   cfldr32 c3, [r0, #0xc]       ; data section preloaded with 0x5
```


Copyright 2007 Cirrus Logic

```
loop
  cfmul32 c0, c0, c3           ; c0 <= c0 * 5
  cfsub32 c0, c0, c2           ; c0 <= c0 - 1
  cfcmp32 r15, c0, c1        ; c0 < 10 ?
  blt loop                          ;   yes
  cfstr32 c0, [r0, #0x0]      ;   no, store result
```

3.2.2 Example 2

The following function performs an FIR filter on the given input stream. The variable "data" points to an array of floating point values to be filtered, "n" is the number of samples for which the filter should be applied, "filter" is the FIR filter to be applied, and "m" is the number of taps in the FIR filter. The "data" array must be "n + m - 1" samples in length, and "n" samples will be produced.

3.2.2.1 C Code

```
void
ComputeFIR(float *data, int n, float *filter, int m)
{
    int i, j;
    float sum;

    for(i = 0; i < n; i++)
    {
        sum = 0;

        for(j = 0; j < m; j++)
        {
            sum += data[i + j] * filter[j];
        }

        data[i] = sum;
    }
}
```

3.2.2.2 MaverickCrunch Assembly Language Instructions

```
ComputeFIR
    mov     r1, r1, lsl #2              ; n *= 4
    mov     r3, r3, lsl #2              ; m *= 4
outer_loop
        mov     r12, r3                   ; j = m * 4
        cfsub64 c0, c0, c0            ; int_sum = 0;
        cfcvt32s c0, c0                 ; sum = float(int_sum);
inner_loop
            cfldrs  c2, [r0], #4          ; c2 = *data++;
```


Copyright 2007 Cirrus Logic

```
        cfldrs  c3, [r2], #4          ; c3 = *filter++;
        cfmuls  c1, c2, c3           ; c1 = c2 * c3;
        cfadds  c0, c0, c1           ; sum += c1;
        subs    r12, r12, #4         ; j -= 4;
        bne     inner_loop          ; branch if j != 0
    sub     r0, r3                          ; data -= m * 4;
    cfstrs  c0, [r0], #4              ; *data++ = sum;
    sub     r2, r3                          ; filter -= m * 4;
    subs    r1, r1, #4                ; n -= 4;
    bne     outer_loop              ; branch if n != 0
mov     pc, lr                          ; return to caller
```

3.3 DSPSC Register

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
INST															

47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
INST															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DAID			HVID			RSVD		ISAT	UI	INT	AEXC	SAT[1:0]		FCC[1:0]	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
V	FWDEN	Invalid	Denorm	RM[1:0]		IXE	UFE	OFE	RSVD	IOE	IX	UF	OF	RSVD	IO

Default:

0x0000_0000_0000_0000

Definition:

MaverickCrunch Status and Control Register. Accessed only via the MaverickCrunch instruction set. All bits, including status bits, are both readable and writable. This register should generally be written only using a read-modify-write sequence.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

INST: Exception Instruction. Whenever an unmasked exception occurs, these 32 bits are loaded with the instruction that caused the exception. Hence, this contains the instruction that caused the most recent unmasked exception.

Copyright 2007 Cirrus Logic

DAID: MaverickCrunch Architecture ID. This read-only value is incremented for each revision of the overall MaverickCrunch co-processor architecture. These bits are "000" for this revision.

HVID: Hardware Version ID. This read-only value is incremented each time the hardware implementation of the architecture named by DAID[2:0] is changed, typically done in response to bugs. These bits are "000" for this version.

ISAT: Integer Saturate Enable. This bit controls whether non-accumulator integer operations, both signed and unsigned, will saturate on overflow or underflow:
0 = Saturation enabled
1 = Saturation disabled

UI: Unsigned Integer Enable. This bit controls whether non-accumulator integer operations treat their operands as signed or unsigned. It also determines the saturation value if the ISAT bit is clear:
0 = Signed integers
1 = Unsigned integers

INT: MaverickCrunch Interrupt. This bit indicates whether an interrupt has occurred. This bit is identical to the external interrupt signal:
0 = No interrupt signaled
1 = Interrupt signaled

AEXC: Asynchronous Exception Enable. This bit determines whether exceptions generated by the co-processor are signaled synchronously or asynchronously to the ARM920T. Synchronous exceptions force all data path instructions to be serialized and to stall the ARM920T. If exceptions are asynchronous, they are signalled by assertion of the DSPINT output of the co-processor, which may interrupt the ARM920T via the interrupt controller. Enabling asynchronous exceptions does provide a performance improvement, but makes it difficult for an interrupt handler to determine the co-processor instruction that caused the exception because the address of the instruction is not preserved. Exceptions may be individually enabled by other bits in this register (IXE, UFE, OFE, and IOE). This bit has no effect if no exceptions are enabled:
0 = Exceptions are synchronous
1 = Exceptions are asynchronous

Copyright 2007 Cirrus Logic

SAT[1:0]: Accumulator saturation mode select. These bits are set to select the saturation mode or to disable saturation for accumulator operations:
0X = Saturation disabled for accumulator operations
10 = Accumulator saturation enabled, bit formats 1.63 and 1.31
11 = Accumulator saturation enabled, bit format 2.62

FCC[1:0]: FCC flags out of comparator:
00 = Operand A equals operand B
01 = Operand A less than operand B
10 = Operand A greater than operand B
11 = Operands are unordered (at least one is NaN)

V: Overflow Flag. Indicates the overflow status of the previous integer operation:
0 = No overflow
1 = Overflow

FWDEN: Forwarding Enable. This bit determines whether data path writeback results are forwarded to the data path operand fetch stage and to the STC/MRC execute stage. When pipeline interlocks occur due to dependencies of data path, STC, and MRC instruction source operands on data path results, setting this bit will improve instruction throughput:
0 = Forwarding not enabled
1 = Forwarding enabled

Invalid:
0 = No invalid operations detected
1 = An invalid operation was performed

Denorm:
0 = No denormalized numbers have been supplied as instruction operands
1 = A denormalized number has been supplied as an instruction operand

RM[1:0]: Rounding Mode. Selects IEEE 754 rounding mode:
0 0 = Round to nearest
0 1 = Round toward 0
1 0 = Round to -∞
1 1 = Round to +∞

IXE: Inexact Trap Enable. Enables/disables software trapping for IEEE 754 inexact exceptions:
0 = Disable software trapping for inexact exceptions
1 = Enable software trapping for inexact exceptions

Copyright 2007 Cirrus Logic

UFE: Underflow Trap Enable. Enables/disables software trapping for IEEE 754 underflow exceptions:
0 = Disable software trapping for underflow exceptions
1 = Enable software trapping for underflow exceptions

OFE: Overflow Trap Enable. Enables/disables software trapping for IEEE 754 overflow exceptions:
0 = Disable software trapping for overflow exceptions
1 = Enable software trapping for overflow exceptions

IOE: Invalid Operator Trap Enable. Enables/disables software trapping for IEEE 754 invalid operator exceptions:
0 = Disable software trapping for invalid operator exceptions
1 = Enable software trapping for invalid operator exceptions

IX: Inexact. Set when an IEEE 754 inexact exception occurs, regardless of whether or not software trapping for inexact exceptions is enabled. Writing a "0" to this position clears the status bit.
0 = No inexact exception detected
1 = Inexact exception detected

UF: Underflow. Set when an IEEE 754 underflow exception occurs, regardless of whether or not software trapping for underflow exceptions is enabled. Writing a "0" to this position clears the status bit.
0 = No underflow exception detected
1 = Underflow exception detected

OF: Overflow. Set when an IEEE 754 overflow exception occurs, regardless of whether or not software trapping for overflow exceptions is enabled. Writing a "0" to this position clears the status bit.
0 = No overflow exception detected
1 = Overflow exception detected

IO: Invalid Operator. Set when an IEEE 754 invalid operator exception occurs, regardless of whether or not software trapping for invalid operator exceptions is enabled. Writing a "0" to this position clears the status bit.
0 = No invalid operator exception detected
1 = Invalid operator exception detected

Copyright 2007 Cirrus Logic

3

3.4 ARM Co-Processor Instruction Format

The ARM V4T architecture defines five ARM co-processor instructions:

- CDP - Co-processor Data Processing
- LDC - Load Co-processor
- STC - Store Co-processor
- MCR - Move to Co-processor Register from ARM Register
- MRC - Move to ARM Register from Co-processor Register

The co-processor instruction assembler notation is found in the ARM programming manuals or the Quick Reference Card. (For additional information, see Preface, "Reference Documents" on page P-3) Formats for the above instructions and variants of these instructions are detailed below.

CDP (Co-Processor Data Processing) Instruction Format

31–28	27–24	23–20	19–16	15–12	11–8	7–5	4	3–0
cond	1110	opcode1	CRn	CRd	cp num	opcode2	0	CRm

LDC (Load Co-Processor) Instruction Format

31–28	27–25	24	23	22	21	20	19–16	15–12	11–8	7–0
cond	110	P	U	N	W	1	Rn	CRd	cp num	offset

STC (Store Co-Processor) Instruction Format

31–28	27–25	24	23	22	21	20	19–16	15–12	11–8	7–0
cond	110	P	U	N	W	0	Rn	CRd	cp num	offset

MCR (Move to Co-Processor from ARM Register) Instruction Format

31–28	27–24	23–21	20	19–16	15–12	11–8	7–5	4	3–0
cond	1110	opcode1	0	CRn	Rd	cp num	opcode2	1	CRm

MRC (Move to ARM Register from Co-Processor) Instruction Format

31–28	27–24	23–21	20	19–16	15–12	11–8	7–5	4	3–0
cond	1110	opcode1	1	CRn	Rd	cp num	opcode2	1	CRm

Copyright 2007 Cirrus Logic

Table 3-5 shows the condition codes, which are bits [31:28] for each instruction format.

Table 3-5. Condition Code Definitions

Cond [31:28]	Mnemonic Extension	Meaning	Status Flag State
0000	EQ	Equal	Z set
0001	NE	Not Equal	Z clear
0010	CS/HS	Carry Set/Unsigned Higher or Same	C set
0011	CC/LO	Carry Clear/Unsigned Lower	C clear
0100	MI	Minus/Negative	N set
0101	PL	Plus/Positive or Zero	N clear
0110	VS	Overflow	V set
0111	VC	No Overflow	V clear
1000	HI	Unsigned Higher	C set and Z clear
1001	LS	Unsigned Lower or Same	C clear or Z set
1010	GE	Signed Greater Than or Equal	N set and V set, or N clear and V clear (N = V)
1011	LT	Signed Less Than	N set and V clear, or N clear and V set (N ! = V)
1100	GT	Signed Greater Than	Z clear, and either N set and V set, or N clear and V clear (Z = 0, N = V)
1101	LE	Signed Less Than or Equal	Z set, or N set and V clear, or N clear and V set (Z = 1, N ! = V)
1110	AL	Always (unconditional)	-
1111	NV	Never	-

The remaining bits in the instruction formats are interpreted as follows:

- `opcode1`: MaverickCrunch co-processor-defined opcode
- `opcode2`: MaverickCrunch co-processor defined opcode
- `CRn`: MaverickCrunch co-processor-defined register
- `CRd`: MaverickCrunch co-processor-defined register
- `CRm`: MaverickCrunch co-processor-defined register
- `Rn`: Specifies an ARM base address register. These bits are ignored by the MaverickCrunch co-processor.
- `Rd`: Specifies a source or destination ARM register
- `cp_num`: Co-processor number
- `P`: Pre-indexing (P=1) or post-indexing (P=0) addressing. This bit is ignored by the MaverickCrunch co-processor.
- `U`: Specifies whether the supplied 8-bit offset is added to a base register (U=1) or subtracted from a base register (U=0). This bit is ignored by the MaverickCrunch co-processor.
- `N`: Specifies the width of a data type involved in a move operation. The MaverickCrunch

Copyright 2007 Cirrus Logic

co-processor uses this bit to distinguish between single precision floating point/32-bit integer numbers (N=0) and double precision floating point/64-bit integer numbers (N=1).

- `W`: Specifies whether or not a calculated address is written back to a base register (W=1) or not (W=0). This bit is ignored by the MaverickCrunch co-processor.
- `offset`: An 8-bit word offset used in address calculations. These bits are ignored by the MaverickCrunch co-processor.

Table 3-6, Table 3-7, Table 3-8, and Table 3-9, define the bit values for opcode2, opcode1, and cp_num for all of the MaverickCrunch instructions.

Table 3-6. LDC/STC Opcode Map

cp num [3:0]	Opcode Bits 22 and 20			
	00	01	10	11
0100 0101	cfstrs cfstr32	cfldrs cfldr32	cfstrd cfstr64	cfldrd cfldr64

Table 3-7. CDP Opcode Map

op code 1 [1:0]	cp num [3:0]	opcode2[2:0]							
		000	001	010	011	100	101	110	111
00	0100	cfcpys	cfcpyd	cfcvtds	cfcvtsd	cfcvt32s	cfcvt32d	cfcvt64s	cfcvt64d
	0101	cfsh32							
	0110	cfmadd32							
01	0100	cfmuls	cfmuld	cfmv32al	cfmv32am	cfmv32ah	cfmv32a	cfmv64a	cfmv32sc
	0101	cfmul32	cfmul64	cfmac32	cfmsc32	cfcvts32	cfcvtd32	cftruncs32	cftruncd32
	0110	cfmsub32							
10	0100			cfmval32	cfmvam32	cfmvah32	cfmva32	cfmva64	cfmvsc32
	0101	cfsh64							
	0110	cfmadda32							
11	0100	cfabss	cfabsd	cfnegs	cfnegd	cfadds	cfaddd	cfsubs	cfsubd
	0101	cfabs32	cfabs64	cfneg32	cfneg64	cfadd32	cfadd64	cfsub32	cfsub64
	0110	cfmsuba32							

Copyright 2007 Cirrus Logic

Table 3-8. MCR Opcode Map

op code1	cp num [3:0]	opcode2[2:0]							
		000	001	010	011	100	101	110	111
0	0100 0101 0110	cfmvdlr cfmv64lr	cfmvdhr cfmv64hr	cfmvsr cfrshl32	cfrshl64				

Table 3-9. MRC Opcode Map

op code1	cp num [3:0]	opcode2[2:0]							
		000	001	010	011	100	101	110	111
0	0100 0101 0110	cfmvrdl cfmvr64l	cfmvrdh cfmvr64h	cfmvrs		cfcmps cfcmp32	cfcmpd cfcmp64		

3.5 Instruction Set for the MaverickCrunch Co-Processor

Table 3-10 summarizes the MaverickCrunch co-processor instruction set. Please note that:

- CRd, CRn, and CRm each refer to any of the 16 general purpose MaverickCrunch registers unless otherwise specified
- CRa refers to any of the MaverickCrunch accumulators
- Rd and Rn refer to any of the 16 general purpose ARM920T registers
- <imm> refers to a seven-bit immediate value

The remainder of this section describes in detail each of the individual MaverickCrunch instructions. The fields in the opcode for each MaverickCrunch instruction are shown. When specific bit values are required for the instruction, they are shown as either '1' or '0'. Any field whose value may vary, such as a register index, is named as in the ARM programming manuals, and its function described below.

Copyright 2007 Cirrus Logic

Fields that are ignored by the co-processor are shaded. Dark shading implies that a field is processed by the ARM itself and can have any value, while light shading indicates that the field, though ignored by both the ARM and the co-processor, should have the value shown.

Table 3-10. MaverickCrunch Instruction Set

Maverick Crunch Co-Processor Instruction Type	ARM Co-Processor Instruction Type	Instruction	Description
Loads	LDC	cfldrs CRd, [Rn]	Load CRd with single stored at address in Rn
		cfldrd CRd, [Rn]	Load CRd with double stored at address in Rn
		cfldr32 CRd, [Rn]	Load CRd with 32-bit integer stored at address in Rn, sign extend through bit 63
		cfldr64 CRd, [Rn]	Load CRd with 64-bit integer stored at address in Rn
Stores	STC	cfstrs CRd, [Rn]	Store single in CRd at address in Rn
		cfstrd CRd, [Rn]	Store double in CRd at address in Rn
		cflstr32 CRd, [Rn]	Store 32-bit integer in CRd at address in Rn
		cfstr64 CRd, [Rn]	Store 64-bit integer in CRd at address in Rn
Moves to co-processor	MCR	cfmvsr CRn, Rd	Move single from Rd to CRn[63:32]
		cfmvdlr CRn, Rd	Move lower half of double from Rd to CRn[31:0]
		cfmvdhr CRn, Rd	Move upper half of double from Rd to CRn[63:32]
		cfmv64lr CRn, Rd	Move lower half of 64-bit integer from Rd to CRn[31:0], sign extend bit 31 through bits [63:31]
		cfmv64hr CRn, Rd	Move upper half of 64-bit integer from Rd to CRn[63:32]
Moves from co-processor	MRC	cfmvsr Rd, CRn	Move single from CRn[63:32] to Rd
		cfmvrdl Rd, CRn	Move lower half of double from CRn[31:0] to Rd
		cfmvrdh Rd, CRn	Move upper half of double from CRn[63:32] to Rd
		cfmvr64l Rd, CRn	Move lower half of 64-bit integer from CRn[31:0] to Rd
		cfmvr64h Rd, CRn	Move upper half of 64-bit integer from CRn[63:32] to Rd
Moves to accumulator	CDP	cfmval32 CRd, CRn	Move 32-bit integer from CRn [31:0] to accumulator CRd[31:0]
		cfmvam32 CRd, CRn	Move 32-bit integer from CRn [31:0] to accumulator CRd[63:32]
		cfmvah32 CRd, CRn	Move lower 8 bits of 32-bit integer from CRn [7:0] to accumulator CRd[71:64]
		cfmva32 CRd, CRn	Move 32-bit integer from CRn[31:0] to accumulator CRd[31:0] and sign extend through bit 71
		cfmva64 CRd, CRn	Move 64-bit integer from CRn to accumulator CRd[63:0] and sign extend through bit 71

Copyright 2007 Cirrus Logic

Table 3-10. MaverickCrunch Instruction Set (Continued)

Maverick Crunch Co-Processor Instruction Type	ARM Co-Processor Instruction Type	Instruction	Description
Moves from accumulator	CDP	cfmv32al CRd, CRn	Move accumulator CRn[31:0] to 32-bit integer CRd[31:0]
		cfmv32am CRd, CRn	Move accumulator CRn[63:32] to 32-bit integer CRd[31:0]
		cfmv32ah CRd, CRn	Move accumulator CRn[71:64] to lower 8 bits of 32-bit integer CRd[31:0]
		cfmv32a CRd, CRn	Saturate to 32-bit integer and move accumulator CRn[31:0] to 32-bit integer CRd[31:0]
		cfmv64a CRd, CRn	Saturate to 64-bit integer and move accumulator CRn[63:0] to 64-bit integer CRd
Move to DSPSC	CDP	cfmvsc32 CRd, CRn	Move CRd to DSPSC; CRn is ignored
Move from DSPSC		cfmv32sc CRd, CRn	Moves DSPSC to CRd; CRn is ignored
Conversions and copies	CDP	cfcpys CRd, CRn	Copy a single from CRn to CRd
		cfcpyd CRd, CRn	Copy a double from CRn to CRd
		cfcvtsd CRd, CRn	Convert a single in CRn to a double in CRd
		cfcvtds CRd, CRn	Convert a double in CRn to a single in CRd
		cfcvt32s CRd, CRn	Convert a 32-bit integer in CRn to a single in CRd
		cfcvt32d CRd, CRn	Convert a 32-bit integer in CRn to a double in CRd
		cfcvt64s CRd, CRn	Convert a 64-bit integer in CRn to a single in CRd
		cfcvt64d CRd, CRn	Convert a 64-bit integer in CRn to a double in CRd
		cfcvts32 CRd, CRn	Convert a single in CRn to a 32-bit integer in CRd
		cfcvtd32 CRd, CRn	Convert a double in CRn to a 32-bit integer in CRd
		cftruncs32 CRd, CRn	Truncate a single in CRn to a 32-bit integer in CRd
		cftruncd32 CRd, CRn	Truncate a double in CRn to a 32-bit integer in CRd
Shifts	MCR	cfrshl32 CRm, CRn, Rd	Shift 32-bit integer in CRn by two's complement value in Rd and store in CRm
		cfrshl64 CRm, CRn, Rd	Shift 64-bit integer in CRn by two's complement value in Rd and store in CRm
	CDP	cfsh32 CRd, CRn, <imm>	Shift 32-bit integer in CRn by <imm> bits and store in CRd, where <imm> is between -32 and 31, inclusive
		cfsh64 CRd, CRn, <imm>	Shift 64-bit integer in CRn by <imm> bits and store in CRd, where <imm> is between -32 and 31, inclusive

Copyright 2007 Cirrus Logic

3

Table 3-10. MaverickCrunch Instruction Set (Continued)

Maverick Crunch Co-Processor Instruction Type	ARM Co-Processor Instruction Type	Instruction	Description
Comparisons	MRC	cfcmps Rd, CRn, CRm	Compare singles in CRn to CRm, result in Rd, or CPSR if Rd == R15
		cfcmpd Rd, CRn, CRm	Compare doubles in CRn to CRm, result in Rd, or CPSR if Rd == R15
		cfcmp32 Rd, CRn, CRm	Compare 32-bit integers in CRn to CRm, result in Rd, or CPSR if Rd == R15
		cfcmp64 Rd, CRn, CRm	Compare 64-bit integers in CRn to CRm, result in Rd, or CPSR if Rd == R15
Floating point arithmetic, single precision	CDP	cfabss CRd, CRn	CRd gets absolute value of CRn
		cfnegs CRd, CRn	CRd gets negation of CRn
		cfadds CRd, CRn, CRm	CRd gets sum of CRn and CRm
		cfsubs CRd, CRn, CRm	CRd gets CRn minus CRm
		cfmuls CRd, CRn, CRm	CRd gets the product of CRn and CRm
Floating point arithmetic, double precision	CDP	cfabsd CRd, CRn	CRd gets absolute value of CRn
		cfnegd CRd, CRn	CRd gets negation of CRn
		cfaddd CRd, CRn, CRm	CRd gets sum of CRn and CRm
		cfsubd CRd, CRn, CRm	CRd gets CRn minus CRm
		cfmuld CRd, CRn, CRm	CRd gets the product of CRn and CRm
32-bit integer arithmetic	CDP	cfabs32 CRd, CRn	CRd gets absolute value of CRn
		cfneg32 CRd, CRn	CRd gets negation of CRn
		cfadd32 CRd, CRn, CRm	CRd gets sum of CRn and CRm
		cfsub32 CRd, CRn, CRm	CRd gets CRn minus CRm
		cfmul32 CRd, CRn, CRm	CRd gets the product of CRn and CRm
		cfmac32 CRd, CRn, CRm	CRd gets sum of CRd and the product of CRn and CRm
		cfmsc32 CRD, CRn, CRm	CRd gets CRd minus the product of CRn and CRm

Copyright 2007 Cirrus Logic

Table 3-10. MaverickCrunch Instruction Set (Continued)

Maverick Crunch Co-Processor Instruction Type	ARM Co-Processor Instruction Type	Instruction	Description
64-bit integer arithmetic	CDP	cfabs64 CRd, CRn	CRd gets absolute value of CRn
		cfneg64 CRd, CRn	CRd gets negation of CRn
		cfadd64 CRd, CRn, CRm	CRd gets sum of CRn and CRm
		cfsub64 CRd, CRn, CRm	CRd gets CRn minus CRm
		cfmul64 CRd, CRn, CRm	CRd gets the product of CRn and CRm
Accumulator arithmetic	CDP	cfmadd32 CRa, CRd, CRn, CRm	Accumulator CRa gets sum of CRd and the product of CRn and CRm
		cfmsub32 CRa, CRd, CRn, CRm	Accumulator CRa gets CRd minus the product of CRn and CRm
		cfmadda32 CRa, CRd, CRn, CRm	Accumulator CRa gets sum of accumulator CRd and the product of CRn and CRm
		cfmsuba32 CRa, CRd, CRn, CRm	Accumulator CRa gets accumulator CRd minus the product of CRn and CRm

3.5.1 Load and Store Instructions

Loading Floating Point Value from Memory

31:28	27:25	24	23	22	21	20	19:16	15:12	11:8	7:0
cond	1 1 0	P	U	N	W	1	Rn	CRd	0 1 0 0	8_bit_word_offset

Description:

Loads a single or double precision floating point value from memory into MaverickCrunch register.

Table 3-11. Mnemonic Codes for Loading Floating Point Value from Memory

Mnemonic	Addressing Mode	N
CFLDRS<cond> CRd, [Rn, <offset>]{!}	Immediate pre-indexed	0
CFLDRS<cond> CRd, [Rn], <offset>	Immediate post-indexed	0
CFLDRD<cond> CRd, [Rn, <offset>]{!}	Immediate pre-indexed	1
CFLDRD<cond> CRd, [Rn], <offset>	Immediate post-indexed	1

Copyright 2007 Cirrus Logic

3

Bit Definitions:

N: Floating point precision - 0 for single, 1 for double.

Rn: Base register in ARM

CRd: Destination register.

Loading Integer Value from Memory

31:28	27:25	24	23	22	21	20	19:16	15:12	11:8	7:0
cond	1 1 0	P	U	N	W	1	Rn	CRd	0 1 0 1	8_bit_word_offset

Description:

Loads a 32- or 64-bit integer from memory into a MaverickCrunch register.

Table 3-12. Mnemonic Codes for Loading Integer Value from Memory

Mnemonic	Addressing Mode	N
CFLDR32<cond> CRd, [Rn, <offset>]{!}	Immediate pre-indexed	0
CFLDR32<cond> CRd, [Rn], <offset>	Immediate post-indexed	0
CFLDR64<cond> CRd, [Rn, <offset>]{!}	Immediate pre-indexed	1
CFLDR64<cond> CRd, [Rn], <offset>	Immediate post-indexed	1

Bit Definitions:

N: Integer width - 0 for 32-bit integer, 1 for 64-bit integer

Rn: Base register in ARM

CRd: Destination register.

Store Floating Point Values to Memory

31:28	27:25	24	23	22	21	20	19:16	15:12	11:8	7:0
cond	1 1 0	P	U	N	W	0	Rn	CRd	0 1 0 0	8_bit_word_offset

Description:

Stores a single or double precision floating point value from a MaverickCrunch register into memory.

Copyright 2007 Cirrus Logic

Mnemonic:

Table 3-13. Mnemonic Codes for Storing Floating Point Values to Memory

Mnemonic	Addressing Mode	N
CFSTRS<cond> CRd, [Rn, <offset>]{!}	Immediate pre-indexed	0
CFSTRS<cond> CRd, [Rn], <offset>	Immediate post-indexed	0
CFSTRD<cond> CRd, [Rn, <offset>]{!}	Immediate pre-indexed	1
CFSTRD<cond> CRd, [Rn], <offset>	Immediate post-indexed	1

Bit Definitions:

N: Floating point precision - 0 for single, 1 for double.

Rn: Base register in ARM

CRd: Source register.

Store Integer Values to Memory

31:28	27:25	24	23	22	21	20	19:16	15:12	11:8	7:0
cond	1 1 0	P	U	N	W	0	Rn	CRd	0 1 0 1	8_bit_word_offset

Description:

Stores a 32- or 64-bit integer value from a MaverickCrunch register into memory.

Mnemonic:

Table 3-14. Mnemonic Codes for Storing Integer Values to Memory

Mnemonic	Addressing Mode	N
CFSTR32<cond> CRd, [Rn, <offset>]{!}	Immediate pre-indexed	0
CFSTR32<cond> CRd, [Rn], <offset>	Immediate post-indexed	0
CFSTR64<cond> CRd, [Rn, <offset>]{!}	Immediate pre-indexed	1
CFSTR64<cond> CRd, [Rn], <offset>	Immediate post-indexed	1

Bit Definitions:

N: Integer width - 0 for 32-bit integer, 1 for 64-bit integer

Rn: Base register in ARM

CRd: Source register.

Copyright 2007 Cirrus Logic

3.5.2 Move Instructions

Move Single Precision Floating Point from ARM to MaverickCrunch

31:28	27:24	23:22	21	20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0	0	CRn	Rd	0 1 0 0	0 1 0	1	CRm

Description:

Moves a single precision floating point number from an ARM register into the upper half of a MaverickCrunch register.

Mnemonic:

CFMVSR<cond> CRn, Rd

Bit Definitions:

Rd: Source ARM register

CRn: Destination register

Move Single Precision Floating Point from MaverickCrunch to ARM

31:28	27:24	23:22	21	20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0	1	CRn	Rd	0 1 0 0	0 1 0	1	CRm

Description:

Moves a single precision floating point number from the upper half of a MaverickCrunch register to an ARM register.

Mnemonic:

CFMVRS<cond> Rd, CRn

Bit Definitions:

Rd: Destination ARM register

CRn: Source register

Move Lower Half Double Precision Float from ARM to MaverickCrunch

31:28	27:24	23:22	21	20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0	0	CRn	Rd	0 1 0 0	0 0 0	1	CRm

Description:

Moves the lower half of a double precision floating point value from an ARM register into the lower half of a MaverickCrunch register.

Mnemonic:

CFMVDLR<cond> CRn, Rd

Bit Definitions:

CRn: Destination register

Rd: Source ARM register

Copyright 2007 Cirrus Logic

Move Lower Half Double Precision Float from MaverickCrunch to ARM

31:28	27:24	23:22	21	20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0	1	CRn	Rd	0 1 0 0	0 0 0	1	CRm

Description:

Moves the lower half of a double precision floating point value stored in a MaverickCrunch register into an ARM register.

Mnemonic:

CFMVRDL<cond> Rd, CRn

Bit Definitions:

Rd: Destination ARM register

CRn: Source register

Move Upper Half Double Precision Float from ARM to MaverickCrunch

31:28	27:24	23:22	21	20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0	0	CRn	Rd	0 1 0 0	0 0 1	1	CRm

Description:

Moves the upper half of a double precision floating point value from an ARM register into the upper half of a MaverickCrunch register.

Mnemonic:

CFMVDHR<cond> CRn, Rd

Bit Definitions:

CRn: Destination register

Rd: Source ARM register

Move Upper Half Double Precision Float from MaverickCrunch to ARM

31:28	27:24	23:22	21	20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0	1	CRn	Rd	0 1 0 0	0 0 1	1	CRm

Description:

Moves the upper half of a double precision floating point value stored in a MaverickCrunch register into an ARM register.

Mnemonic:

CFMVRDH<cond> Rd, CRn

Bit Definitions:

Rd: Destination ARM register

CRn: Source register

Copyright 2007 Cirrus Logic

Move Lower Half 64-bit Integer from ARM to MaverickCrunch

31:28	27:24	23:22	21	20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0	0	CRn	Rd	0 1 0 1	0 0 0	1	CRm

Description:

Moves the lower half of a 64-bit integer from an ARM register into the lower half of a MaverickCrunch register and sign extend it.

Mnemonic:

CFMV64LR<cond> CRn, Rd

Bit Definitions:

CRn: Destination register

Rd: Source ARM register

Move Lower Half 64-bit Integer from MaverickCrunch to ARM

31:28	27:24	23:22	21	20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0	1	CRn	Rd	0 1 0 1	0 0 0	1	CRm

Description:

Moves the lower half of a 64-bit integer stored in a MaverickCrunch register into an ARM register.

Mnemonic:

CFMVR64L<cond> Rd, CRn

Bit Definitions:

Rd: Destination ARM register

CRn: Source register

Move Upper Half 64-bit Integer from ARM to MaverickCrunch

31:28	27:24	23:22	21	20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0	0	CRn	Rd	0 1 0 1	0 0 1	1	CRm

Description:

Moves the upper half of a 64-bit integer from an ARM register into the upper half of a MaverickCrunch register.

Mnemonic:

CFMV64HR<cond> CRn, Rd

Bit Definitions:

CRn: Destination register

Rd: Source ARM register

Copyright 2007 Cirrus Logic

Move Upper Half 64-bit Integer from MaverickCrunch to ARM

31:28	27:24	23:22	21	20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0	1	CRn	Rd	0 1 0 1	0 0 1	1	CRm

Description:

Moves the upper half of a 64-bit integer stored in a MaverickCrunch register into an ARM register.

Mnemonic:

CFMVR64H<cond> Rd, CRn

Bit Definitions:

Rd: Destination ARM register

CRn: Source register

3.5.3 Accumulator and DSPSC Move Instructions

Move MaverickCrunch Register to Lower Accumulator

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 0	CRn	CRd	0 1 0 0	0 1 0	0	CRm

Description:

Moves the low 32 bits of a MaverickCrunch register to the lowest 32 bits of an accumulator (31:0).

Mnemonic:

CFMVAL32<cond> CRd, CRn

Bit Definitions:

CRd: Destination accumulator

CRn: Source register

Move Lower Accumulator to MaverickCrunch Register

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 1	CRn	CRd	0 1 0 0	0 1 0	0	CRm

Description:

Moves the lowest 32 bits of an accumulator (31:0) to the low 32 bits of a MaverickCrunch register.

Mnemonic:

CFMV32AL<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

CRn: Source accumulator

Copyright 2007 Cirrus Logic

3

Move MaverickCrunch Register to Middle Accumulator

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 0	CRn	CRd	0 1 0 0	0 1 1	0	CRm

Description:

Moves the low 32 bits of a MaverickCrunch register to the middle 32 bits of an accumulator (63:32).

Mnemonic:

CFMVAM32<cond> CRd, CRn

Bit Definitions:

CRd: Destination accumulator

CRn: Source register

Move Middle Accumulator to MaverickCrunch Register

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 1	CRn	CRd	0 1 0 0	0 1 1	0	CRm

Description:

Moves the middle 32 bits of an accumulator (63:32) to the low 32 bits of a MaverickCrunch register.

Mnemonic:

CFMV32AM<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

CRn: Source accumulator

Move MaverickCrunch Register to High Accumulator

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 0	CRn	CRd	0 1 0 0	1 0 0	0	CRm

Description:

Moves the lowest 8 bits (7:0) of a MaverickCrunch register to the highest 8 bits of an accumulator (71:64).

Mnemonic:

CFMVAH32<cond> CRd, CRn

Bit Definitions:

CRd: Destination accumulator

CRn: Source register

Copyright 2007 Cirrus Logic

Move High Accumulator to MaverickCrunch Register

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 1	CRn	CRd	0 1 0 0	1 0 0	0	CRm

Description:

Moves the highest 8 bits of an accumulator (71:64) to the lowest 8 bits of a MaverickCrunch register (7:0).

Mnemonic:

CFMV32AH<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

CRn: Source accumulator

Move 32-bit Integer from Accumulator

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 1	CRn	CRd	0 1 0 0	1 0 1	0	CRm

Description:

Saturates and rounds an accumulator value to 32 bits and moves the result to the low 32 bits of a MaverickCrunch register.

Mnemonic:

CFMV32A<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

CRn: Source accumulator

Move 32-bit Integer to Accumulator

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 0	CRn	CRd	0 1 0 0	1 0 1	0	CRm

Description:

Moves a 32-bit value from a MaverickCrunch register to an accumulator and sign extend to 72 bits.

Mnemonic:

CFMVA32<cond> CRd, CRn

Bit Definitions:

CRd: Destination accumulator

CRn: Source register

Copyright 2007 Cirrus Logic

Move 64-bit Integer from Accumulator

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 1	CRn	CRd	0 1 0 0	1 1 0	0	CRm

Description:

Saturates and rounds an accumulator value to 64 bits and moves the result to a MaverickCrunch register.

Mnemonic:

CFMV64A<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

CRn: Source accumulator

Move 64-bit Integer to Accumulator

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 0	CRn	CRd	0 1 0 0	1 1 0	0	CRm

Description:

Moves a 64-bit value from a MaverickCrunch register to an accumulator and sign extend to 72 bits.

Mnemonic:

CFMVA64<cond> CRd, CRn

Bit Definitions:

CRd: Destination accumulator

CRn: Source register

Move from MaverickCrunch Register to Control/Status Register

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 0	CRn	CRd	0 1 0 0	1 1 1	0	CRm

Description:

Moves a 64-bit value from a MaverickCrunch register to the MaverickCrunch Status/Control register, DSPSC. All DSPSC bits are writable. CRn is ignored.

Mnemonic:

CFMVSC32<cond> CRd, CRn

Bit Definitions:

CRd: Source register

Copyright 2007 Cirrus Logic

Move from Control/Status Register to MaverickCrunch Register

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 1	CRn	CRd	0 1 0 0	1 1 1	0	CRm

Description:

Moves a 64-bit value from the MaverickCrunch Status/Control register, DSPSC, to a MaverickCrunch register. CRn is ignored.

Mnemonic:

CFMV32SC<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

3.5.4 Copy and Conversion Instructions

Copy Single Precision Floating Point

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 0	CRn	CRd	0 1 0 0	0 0 0	0	CRm

Description:

Copies a single precision floating point value from one register to another.

Mnemonic:

CFCPYS<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

CRn: Source register

Copy Double Precision Floating Point

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 0	CRn	CRd	0 1 0 0	0 0 1	0	CRm

Description:

Copies a double precision floating point value from one register to another.

Mnemonic:

CFCPYD<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

CRn: Source register

Copyright 2007 Cirrus Logic

Convert Single Precision Floating Point to Double Precision Floating Point

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 0	CRn	CRd	0 1 0 0	0 1 1	0	CRm

Description:

Converts a single precision floating point value to a double precision floating point value.

Mnemonic:

CFCVTSD<cond> CRd, CRn

Bit Definitions

CRd: Destination register

CRn: Source register

Convert Double Precision Floating Point to Single Precision Floating Point

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 0	CRn	CRd	0 1 0 0	0 1 0	0	CRm

Description:

Converts a double precision floating point value to a single precision floating point value.

Mnemonic:

CFCVTDS<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

CRn: Source register

Convert 32-bit Integer to Single Precision Floating Point

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 0	CRn	CRd	0 1 0 0	1 0 0	0	CRm

Description:

Converts a 32-bit integer to a single precision floating point value.

Mnemonic:

CFCVT32S<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

CRn: Source register

Copyright 2007 Cirrus Logic

Convert 32-bit Integer to Double Precision Floating Point

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 0	CRn	CRd	0 1 0 0	1 0 1	0	CRm

Description:

Converts a 32-bit integer to a double precision floating point value.

Mnemonic:

CFCVT32D<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

CRn: Source register

Convert 64-bit Integer to Single Precision Floating Point

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 0	CRn	CRd	0 1 0 0	1 1 0	0	CRm

Description:

Converts a 64-bit integer to a single precision floating point value.

Mnemonic:

CFCVT64S<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

CRn: Source register

Convert 64-bit Integer to Double Precision Floating Point

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 0	CRn	CRd	0 1 0 0	1 1 1	0	CRm

Description:

Converts a 64-bit integer to a double precision floating point value.

Mnemonic:

CFCVT64D<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

CRn: Source register

Copyright 2007 Cirrus Logic

Convert Single Precision Floating Point to 32-bit Integer

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 1	CRn	CRd	0 1 0 1	1 0 0	0	CRm

Description:

Converts a single precision floating point number to a 32-bit integer.

Mnemonic:

CFCVTS32<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

CRn: Source register

Convert Double Precision Floating Point to 32-bit Integer

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 1	CRn	CRd	0 1 0 1	1 0 1	0	CRm

Description:

Converts a double precision floating point number to a 32-bit integer.

Mnemonic:

CFCVTD32<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

CRn: Source register

Truncate Single Precision Floating Point to 32-bit Integer

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 1	CRn	CRd	0 1 0 1	1 1 0	0	CRm

Description:

Truncates a single precision floating point number to a 32-bit integer.

Mnemonic:

CFTRUNCS32<cond> CRd, CRn

Bit Definitions:

CRd: Destination register.

CRn: Source register.

Truncate Double Precision Floating Point to 32-bit Integer

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 1	CRn	CRd	0 1 0 1	1 1 1	0	CRm

Description:

Truncates a double precision floating point number to a 32-bit integer.

Mnemonic:

CFTRUNCD32<cond> CRd, CRn

Copyright 2007 Cirrus Logic

Bit Definitions:

CRd: Destination register

CRn: Source register

3.5.5 Shift Instructions

Shift 32-bit Integer

31:28	27:24	23:22	21	20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0	0	CRn	Rd	0 1 0 1	0 1 0	1	CRm

Description:

Shifts a 32-bit integer left or right. The shift count is a two's complement integer stored in an ARM register; the count is positive for left shifts and negative for right shifts. This instruction may also be used to copy a 32-bit integer from one register to another by using a shift value of 0.

Mnemonic:

CFRSHL32<cond> CRm, CRn, Rd

Bit Definitions:

CRm: Destination register

CRn: Source register

Rd: Shift count register in ARM

Shift 64-bit Integer

31:28	27:24	23:22	21	20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0	0	CRn	Rd	0 1 0 1	0 1 1	1	CRm

Definition:

Shifts a 64-bit integer left or right. The shift count is a two's complement integer stored in an ARM register; the count is positive for left shifts and negative for right shifts. This instruction may also be used to copy a 64-bit integer from one register to another using a shift value of 0.

Mnemonic:

CFRSHL64<cond> CRm, CRn, Rd

Bit Definitions:

CRm: Destination register

CRn: Source register

Rd: Shift count register in ARM

Copyright 2007 Cirrus Logic

Shift 32-bit Integer Immediate

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 0	CRn	CRd	0 1 0 1	Shift[6:4]	0	Shift[3:0]

Definition:

Shift a 32-bit integer by the count specified in the seven bit, two's complement immediate value. A positive number indicates a left shift and a negative number indicates a right shift. This instruction may also be used to copy a 32-bit integer from one register to another using a shift value of 0.

Mnemonic:

CFSH32<cond> CRd, CRn, Shift[6:0]

Bit Definitions:

CRd: Destination register

CRn: Source register

Shift[6:0]: Shift count.

Shift 64-bit Integer Immediate

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 0	CRn	CRd	0 1 0 1	Shift[6:4]	0	Shift[3:0]

Definition:

Shifts a 64-bit integer by a count specifies in the seven bit, two's complement immediate value. A positive number indicates a left shift and a negative number indicates a right shift. This instruction may also be used to copy a 64-bit integer from one register to another by using a shift value of 0.

Mnemonic:

CFSH64<cond> CRd, CRn, Shift[6:0]

Bit Definitions:

CRd: Destination register

CRn: Source register

Shift[6:0]: Shift count.

3.5.6 Compare Instructions

Compare Single Precision Floating Point

31:28	27:24	23:22	21	20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0	1	CRn	Rd	0 1 0 0	1 0 0	1	CRm

Definition:

Compares two single precision floating point numbers and stores an integer representing the result in the ARM920T register; the highest four bits of the integer result match the N, Z, C, and V bits, respectively, in the ARM920T's program status register, while the bottom 28 bits are zeros. If Rd = 15, then the four status bits are stored in the ARM status register, CPSR.

Copyright 2007 Cirrus Logic

Mnemonic:

CFCMPS<cond> Rd, CRn, CRm

Bit Definitions:

CRn: First source register

CRm: Second source register

Rd: Destination ARM register. If Rd = 15, destination is ARM N, C, Z, and V flags.

Compare Double Precision Floating Point

31:28	27:24	23:22	21	20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0	1	CRn	Rd	0 1 0 0	1 0 1	1	CRm

Definition:

Compares two double precision floating point numbers and stores an integer representing the result in the ARM920T register; the highest four bits of the integer result match the N, Z, C, and V bits, respectively, in the ARM920T's program status register, while the bottom 28 bits are zeros. If Rd = 15, then the four status bits are stored in the ARM status register, CPSR.

Mnemonic:

CFCMPD<cond> Rd, CRn, CRm

Bit Definitions:

CRn: First source register

CRm: Second source register

Rd: Destination ARM register. If Rd = 15, destination is ARM N, C, Z, and V flags.

Compare 32-bit Integers

31:28	27:24	23:22	21	20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0	1	CRn	Rd	0 1 0 1	1 0 0	1	CRm

Definition:

Compares two 32-bit integers and stores an integer representing the result in the ARM920T register; the highest four bits of the integer result match the N, Z, C, and V bits, respectively, in the ARM920T's program status register, while the bottom 28 bits are zeros. If Rd = 15, then the four status bits are stored in the ARM status register, CPSR.

Mnemonic:

CFCMP32<cond> Rd, CRn, CRm

Bit Definitions:

CRn: First source register

CRm: Second source register

Copyright 2007 Cirrus Logic

Rd: Destination ARM register. If Rd = 15, destination is ARM N, C, Z, and V flags.

Compare 64-bit Integers

31:28	27:24	23:22	21	20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0	1	CRn	Rd	0 1 0 1	1 0 1	1	CRm

Description:

Compares two 64-bit integers and stores an integer representing the result in the ARM920T register; the highest four bits of the integer result match the N, Z, C, and V bits, respectively, in the ARM920T's program status register, while the bottom 28 bits are zeros. If Rd = 15, then the four status bits are stored in the ARM status register, CPSR.

Mnemonic:

CFCMP64<cond> Rd, CRn, CRm

Bit Definitions:

CRn: First source register

CRm: Second source register

Rd: Destination ARM register. If Rd = 15, destination is ARM N, C, Z, and V flags.

3.5.7 Floating Point Arithmetic Instructions

Single Precision Floating Point Absolute Value

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 1	CRn	CRd	0 1 0 0	0 0 0	0	CRm

Description:

Computes the absolute value of a single precision floating point number:
CRd = |CRn|

Mnemonic:

CFABSS<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

CRn: Source register

Double Precision Floating Point Absolute Value

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 1	CRn	CRd	0 1 0 0	0 0 1	0	CRm

Description:

Computes the absolute value of a double precision floating point number.

Mnemonic:

CFABSD<cond> CRd, CRn

Copyright 2007 Cirrus Logic

Bit Definitions:

CRd: Destination register

CRn: Source register

Single Precision Floating Point Negate

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 1	CRn	CRd	0 1 0 0	0 1 0	0	CRm

Description:

Takes the negative of a single precision floating point number:
CRd = -CRn

Mnemonic:

CFNEGS<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

CRn: Source register

Double Precision Floating Point Negate

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 1	CRn	CRd	0 1 0 0	0 1 1	0	CRm

Description:

Takes the negative of a double precision floating point number.

Mnemonic:

CFNEGD<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

CRn: Source register

Single Precision Floating Point Add

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 1	CRn	CRd	0 1 0 0	1 0 0	0	CRm

Description:

Adds two single precision floating point numbers:
CRd = CRn + CRm

Mnemonic:

CFADDS<cond> CRd, CRn, CRm

Bit Definitions:

CRd: Destination register

CRn: Addend register

CRm: Addend register

Copyright 2007 Cirrus Logic

Double Precision Floating Point Add

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 1	CRn	CRd	0 1 0 0	1 0 1	0	CRm

Description:

Adds two double precision floating point numbers.

Mnemonic:

CFADDD<cond> CRd, CRn, CRm

Bit Definitions:

CRd: Destination register

CRn: Addend register

CRm: Addend register

Single Precision Floating Point Subtract

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 1	CRn	CRd	0 1 0 0	1 1 0	0	CRm

Description:

Subtracts two single precision floating point numbers:
CRd = CRn - CRm

Mnemonic:

CFSUBS<cond> CRd, CRn, CRm

Bit Definitions:

CRd: Destination register

CRn: Minuend register

CRm: Subtrahend register

Double Precision Floating Point Subtract

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 1	CRn	CRd	0 1 0 0	1 1 1	0	CRm

Description:

Subtracts two double precision floating point numbers.

Mnemonic:

CFSUBD<cond> CRd, CRn, CRm

Bit Definitions:

CRd: Destination register

CRn: Minuend register

CRm: Subtrahend register

Copyright 2007 Cirrus Logic

Single Precision Floating Point Multiply

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 1	CRn	CRd	0 1 0 0	0 0 0	0	CRm

Description:

Multiplies two single precision floating point numbers:

CRd = CRn × CRm

Mnemonic:

CFMULS<cond> CRd, CRn, CRm

Bit Definitions:

CRd: Destination register

CRn: Multiplicand register

CRm: Multiplicand register

Double Precision Floating Point Multiply

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 1	CRn	CRd	0 1 0 0	0 0 1	0	CRm

Description:

Multiplies two double precision floating point numbers.

Mnemonic:

CFMULD<cond> CRd, CRn, CRm

Bit Definitions:

CRd: Destination register

CRn: Multiplicand register

CRm: Multiplicand register

3.5.8 Integer Arithmetic Instructions

32-bit Integer Absolute Value

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 1	CRn	CRd	0 1 0 1	0 0 0	0	CRm

Description:

Computes the absolute value of a 32-bit integer.

Mnemonic:

CFABS32<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

CRn: Source register

Copyright 2007 Cirrus Logic

64-bit Integer Absolute Value

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 1	CRn	CRd	0 1 0 1	0 0 1	0	CRm

Description:

Computes the absolute value of a 64-bit integer.

Mnemonic:

CFABS64<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

CRn: Source register

32-bit Integer Negate

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 1	CRn	CRd	0 1 0 1	0 1 0	0	CRm

Description:

Negate a 32-bit integer.

Mnemonic:

CFNEG32<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

CRn: Source register

64-bit Integer Negate

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 1	CRn	CRd	0 1 0 1	0 1 1	0	CRm

Description:

Negate a 64-bit integer.

Mnemonic:

CFNEG64<cond> CRd, CRn

Bit Definitions:

CRd: Destination register

CRn: Source register

32-bit Integer Add

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 1	CRn	CRd	0 1 0 1	1 0 0	0	CRm

Description:

Adds two 32-bit integers.

Copyright 2007 Cirrus Logic

Mnemonic:

CFADD32<cond> CRd, CRn, CRm

Bit Definitions:

CRd: Destination register

CRn: Addend register

CRm: Addend register

64-bit Integer Add

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 1	CRn	CRd	0 1 0 1	1 0 1	0	CRm

Description:

Adds two 64-bit integers.

Mnemonic:

CFADD64<cond> CRd, CRn, CRm

Bit Definitions:

CRd: Destination register

CRn: Addend register

CRm: Addend register

32-bit Integer Subtract

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 1	CRn	CRd	0 1 0 1	1 1 0	0	CRm

Description:

Subtracts two 32-bit integers.

Mnemonic:

CFSUB32<cond> CRd, CRn, CRm

Bit Definitions:

CRd: Destination register

CRn: Minuend register

CRm: Subtrahend register

64-bit Integer Subtract

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 1	CRn	CRd	0 1 0 1	1 1 1	0	CRm

Description:

Subtracts two 64-bit integers.

Mnemonic:

CFSUB64<cond> CRd, CRn, CRm

Copyright 2007 Cirrus Logic

Bit Definitions:

CRd: Destination register

CRn: Minuend register

CRm: Subtrahend register

32-bit Integer Multiply

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 1	CRn	CRd	0 1 0 1	0 0 0	0	CRm

Description:

Multiplies two 32-bit integers.

Mnemonic:

CFMUL32<cond> CRd, CRn, CRm

Bit Definitions:

CRd: Destination register

CRn: Multiplicand register

CRm: Multiplicand register

64-bit Integer Multiply

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 1	CRn	CRd	0 1 0 1	0 0 1	0	CRm

Description:

Multiplies two 64-bit integers.

Mnemonic:

CFMUL64<cond> CRd, CRn, CRm

Bit Definitions:

CRd: Destination register

CRn: Multiplicand register

CRm: Multiplicand register

32-bit Integer Multiply-Add

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 1	CRn	CRd	0 1 0 1	0 1 0	0	CRm

Description:

Multiplies two 32-bit integers and adds the result to another 32-bit integer:
CRd = CRd + (CRn × CRm)

Mnemonic:

CFMAC32<cond> CRd, CRn, CRm

Copyright 2007 Cirrus Logic

Bit Definitions:

CRd: Destination/addend register

CRn: Multiplicand register

CRm: Multiplicand register

32-bit Integer Multiply-Subtract

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 1	CRn	CRd	0 1 0 1	0 1 1	0	CRm

Description:

Multiplies two 32-bit integers and subtracts the result from another 32-bit integer:

CRd = CRd - (CRn × CRm)

Mnemonic:

CFMSC32<cond> CRd, CRn, CRm

Bit Definitions:

CRd: Destination/minuend register

CRn: Multiplicand register

CRm: Multiplicand register

3.5.9 Accumulator Arithmetic Instructions

32-bit Integer Multiply-Add, Result to Accumulator

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 0	CRn	CRd	0 1 1 0	CRa	0	CRm

Description:

Multiplies two 32-bit integers, adds the product to a third 32-bit integer, and stores the result in an accumulator:

CRa = CRd + (CRn × CRm)

Mnemonic:

CFMADD32<cond> CRa, CRd, CRn, CRm

Bit Definitions:

CRa: Destination accumulator

CRd: Addend register

CRn: Multiplicand register

CRm: Multiplicand register

Copyright 2007 Cirrus Logic

32-bit Integer Multiply-Subtract, Result to Accumulator

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	0 1	CRn	CRd	0 1 1 0	CRa	0	CRm

Description:

Multiplies two 32-bit integers, subtracts the product from a third 32-bit integer, and stores the result in an accumulator:

CRa = CRd - (CRn × CRm)

Mnemonic:

CFMSUB32<cond> CRa, CRd, CRn, CRm

Bit Definitions:

CRa: Destination accumulator

CRd: Minuend register

CRn: Multiplicand register

CRm: Multiplicand register

32-bit Integer Multiply-Add to Accumulator

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 0	CRn	CRd	0 1 1 0	CRa	0	CRm

Description:

Multiplies two 32-bit integers, adds the product to an accumulator, and stores the result in an accumulator:

CRa = CRd + (CRn × CRm)

Mnemonic:

CFMADDA32<cond> CRa, CRd, CRn, CRm

Bit Definitions:

CRa: Destination accumulator

CRd: Addend accumulator

CRn: Multiplicand register

CRm: Multiplicand register

Copyright 2007 Cirrus Logic

32-bit Integer Multiply-Subtract from Accumulator

31:28	27:24	23:22	21:20	19:16	15:12	11:8	7:5	4	3:0
cond	1 1 1 0	0 0	1 1	CRn	CRd	0 1 1 0	CRa	0	CRm

Description:

Multiplies two 32-bit integers, subtracts the product from an accumulator, and stores the result in an accumulator:

CRa = CRd - (CRn × CRm)

Mnemonic:

CFMSUBA32<cond> CRa, CRd, CRn, CRm

Bit Definitions:

CRa:	Destination accumulator
CRd:	Specifies minuend accumulator
CRn:	Multiplicand register
CRm:	Multiplicand register

Copyright 2007 Cirrus Logic

Copyright 2007 Cirrus Logic

Chapter 4
Boot ROM 4

4.1 Introduction

The Boot ROM allows a program or OS to boot from the following devices:

- SPI Flash
- FLASH, SyncFLASH or SyncROM
- UART1

4.1.1 Boot ROM Hardware Operational Overview

The Boot ROM is an AHB slave device containing a 16 kbyte mask-programmed ROM. The AHB slave always operates with one wait state, so all data reads from the ROM use 2 HCLK cycles.

On system reset, the ARM920T begins executing code at address zero. The system follows the Hardware Configuration controls to select the boot device that appears at address zero. If Internal Boot is selected, the Boot ROM is mapped to address zero and the ARM920T will execute the Boot ROM code.

4.1.1.1 Memory Map

The normal Boot ROM base address base is 0x8009_0000. It will alias on 16 kbyte intervals. When internal boot is active, the Boot ROM is double decoded and appears at its normal address base and at address 0x0000_0000. At address 0x0000_0000 plus the current offset, the Boot ROM can write the BootModeClr bit to remap itself back to 0x8009_0000 plus the current offset. Execution then continues with the instruction at the next Boot ROM address in 0x8009_0000 space.

4.1.2 Boot ROM Software Operational Overview

The Boot ROM is a 16 kbyte mask-programmed ROM that controls the source of the first off-chip code that is executed by the ARM Core. The code within the Boot ROM supports the following sources for the processor's initialization program:

- UART1: Code is downloaded through UART1 into an on chip buffer and executed
- SPI Serial Flash: Code is copied from an SPI Serial Flash into an on-chip buffer and executed
- FLASH: Code present in external FLASH memory is executed directly

Copyright 2007 Cirrus Logic

Note that the code retrieved via UART1 and the SPI Serial Flash is not intended to be a complete operating system image. It is intended to be a small (up to 2 kbyte) loader that will, in turn, retrieve a complete operating system image. This small loader can retrieve this complete image through UART1 or the SPI Serial Flash (just as the Boot ROM did) or it can be more sophisticated and retrieve it through the IrDA, USB, or Ethernet interfaces.

The Boot ROM code disables the ARM920T's MMU, so any loader program that is downloaded sees physical addresses. The loader is free to initialize the page tables and start the MMU and caches if needed.

The Boot ROM code also does not enable interrupts or timers, so that the system delivered to the user is in a known safe state and is ready for an operating system or for user code to be loaded.

4.1.2.1 Image Header

For images copied from the SPI Serial Flash or external FLASH, one of the ASCII strings, "CRUS" or "SURC", must be present as a HeaderID prefixed to an executable image.

4.1.2.2 Boot Algorithm

The steps in the software boot process are:

1. Remap memory
2. Turn the green LED off and the red LED on
3. Disable the Watchdog timer
4. Read the Boot State
5. Set up the Clocks to run from external clocks (PLLs are not configured)
6. Based on the Boot State memory width, follow steps A, B, and C.
   A. Initialize the SYNC Flash and SMC memory interfaces for slow (maximum compatibility) operation
   B. Initialize the SDRAM interfaces.
   C. Perform minimal memory tests
7. Based on the contents of the SysCfg register, start serial download (see Figure 4-1), and then follow Steps A, B, C, D, E, and F.
   A. Initialize UART1 to 9600 baud, 8 bits, no parity, 1 stop bit
   B. Output a "<" character
   C. Read 2048 (decimal count) characters from UART1 and store these in the internal Boot buffer (alias for the Ethernet Mac buffer)
   D. Output a ">" to signify 2048 characters have been read
   E. Turn on Green LED
   F. Jump to the start of the internal Boot Buffer

Copyright 2007 Cirrus Logic

8. If it is not a Serial Download, attempt to read from SPI Serial Flash (see Figure 4-1), and then follow Steps A, B, C, and D.
   A. Check if the first 4 bytes from the Serial Flash are equal to "CRUS" or to "SURC" in ASCII, verifying the HeaderID
   B. Read the next 2048 (decimal count) bytes into the Internal Boot Buffer
   C. Turn on Green LED
   D. Jump to the start of the Internal Boot Buffer
9. Attempt to read the "CRUS" or "SURC" HeaderID in ASCII in FLASH memory at FLASH Base + 0x0000, and verify the HeaderID. This is read in for each FLASH Chip select (see Figure 4-1), and then follow Steps A and B.
   A. Turn on Green LED
   B. Jump to the start of FLASH memory plus four bytes
10. Attempt to read the "CRUS" or "SURC" HeaderID in ASCII in FLASH memory at FLASH Base + 0x1000, and verify the HeaderID. This is read in for each FLASH Chip select (see Figure 4-1), and then follow Steps A and B.
   A. Turn on Green LED
   B. Jump to the start of FLASH memory
11. Attempt to read the "CRUS" or "SURC" HeaderID in ASCII in memory at 0xC000_0000 and 0xF000_0000, and verify the HeaderID. This is read in for SDRAM or SyncFLASH boot (see Figure 4-1), and then follow Steps A and B.
   A. Turn on Green LED
   B. Jump to memory location 0xC000_0004 or 0xF000_0004
12. Attempt to read the "CRUS" or "SURC" HeaderID in ASCII in memory at 0xC000_1000 and 0xF000_1000, and verify the HeaderID. This is read in for SDRAM or SyncFLASH boot (see Figure 4-1), and then follow Steps A and B.
   A. Turn on Green LED
   B. Jump to memory location 0xC000_0000 or 0xF000_0000
13. If "CRUS" or "SURC" HeaderID is not found, copy dummy vectors into low SDRAM, and then follow Step A.
   A. Flash Green LED

4.1.2.3 Flowchart

Figure 4-1 provides a flow chart for operation of the Boot ROM software.

Copyright 2007 Cirrus Logic

4

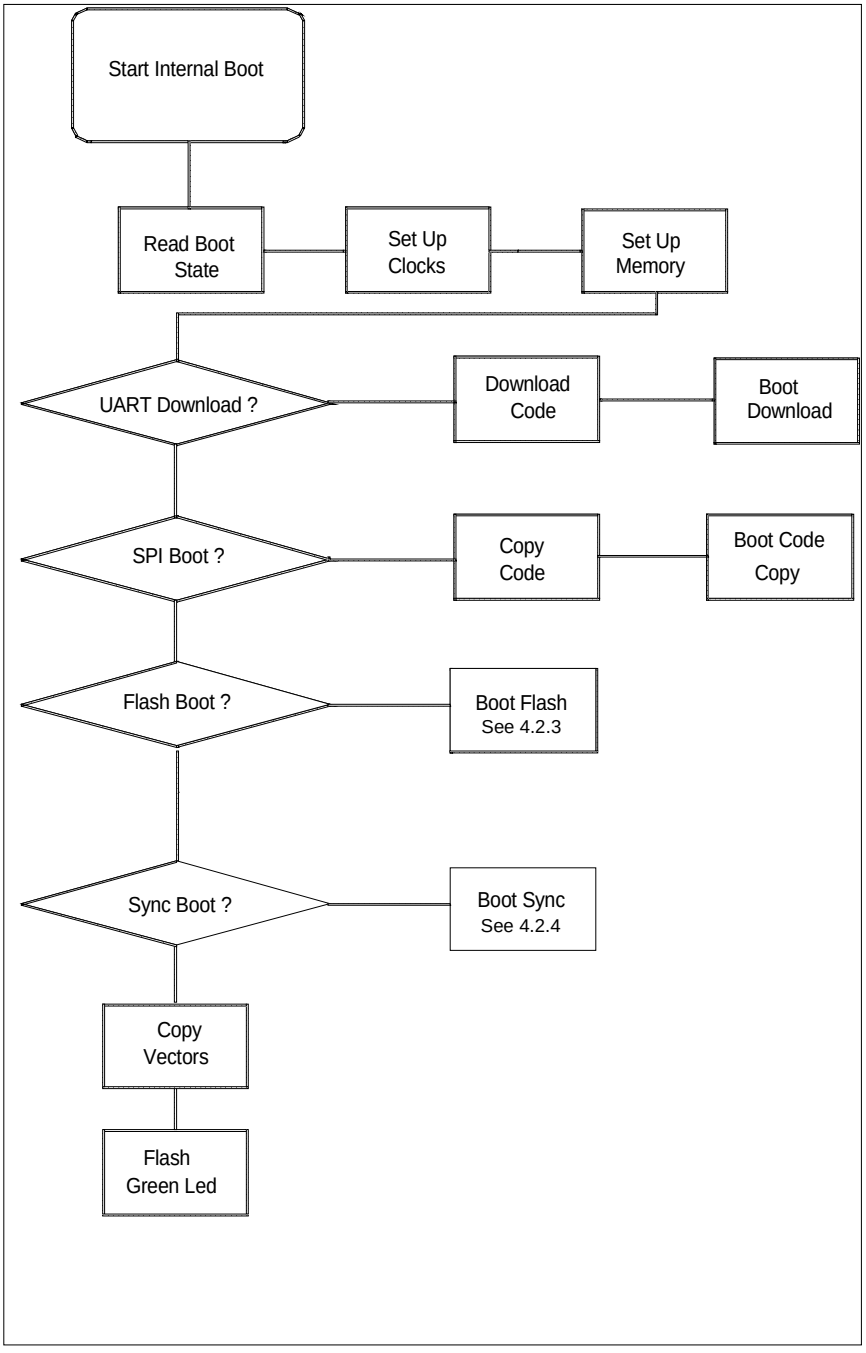


Figure 4-1. Flow Chart of Boot ROM Software

4.2 Boot Options

Table 4-1 shows configuration settings that are common to all boot modes.

Copyright 2007 Cirrus Logic

4

Table 4-1. Boot Configuration Options

EECLK	EEDAT	BOOT1	BOOT0	ASDO	CSn[7:6]	Boot Configuration
0	1	0	0	1	0 0 0 1 1 0 1 1	External boot using Sync boot mode and SDCSn3. The media type must be either SyncROM or SyncFLASH. The selection of the bus width is determined by latched **CSn[7:6]** value: 16-bit 16-bit 32-bit 32-bit
0	1	0	0	0	0 0 0 1 1 0 1 1	External boot using Async boot mode and CSn0. The selection of the bus width is determined by latched **CSn[7:6]** value: 8-bit 16-bit 32-bit 32-bit
1	1	0	1	x	xx	Internal boot from UART1.
1	1	0	0	x	xx	Internal SPI boot if HeaderID is found.
1	1	0	0	1	0 0 0 1 1 0 1 1	Internal boot using SYNC boot mode at the chip select where the HeaderID exists. The selection of the bus width is determined by latched **CSn[7:6]** value: 16-bit 16-bit 32-bit 32-bit See memory map in Table 2-7 on page 2-16 for SYNC boot mode.
1	1	0	0	0	0 0 0 1 1 0 1 1	Internal boot using ASYNC boot mode at the chip select where the HeaderID exists. The selection of the bus width is determined by latched **CSn[7:6]** value: 8-bit 16-bit 32-bit 32-bit See memory map in Table 2-7 on page 2-16 for ASYNC boot mode.

Note: ASYNC boot mode is the preferred boot mode type for new designs.

Copyright 2007 Cirrus Logic

4.2.1 UART Boot

Make sure that the boot configuration pins (see Table 5-1 on page 5-2) are configured for internal boot mode. **EEDAT** and **BOOT0** should be pulled high and **BOOT1** should be pulled low as shown in Table 5-2 on page 5-3. UART 1 is configured at 9600 bps, 8-bits, No Parity, No flow control. The code performs:

1. A single "<" is output by UART 1
2. The ASCII "CRUS" or "SURC" value in the HeaderID is read
3. 2048 characters are received by UART 1 and copied to the Ethernet buffer at address 0x8001_4000
4. The ARM Core will jump to 0x8001_4000. The ARM Core will be in SVC mode when the jump occurs.

4.2.2 SPI Boot

To boot from an SPI Serial Flash device, make sure that the boot configuration pins (see Table 5-1 on page 5-2) are configured for internal boot mode. **EEDAT** should be pulled high and **LBOOT1** and **LBOOT0** should be pulled low as shown in Table 5-2 on page 5-3.

To boot from the SPI ROM, place the ASCII "CRUS" or "SURC" value in the HeaderID at the first location in the ROM. The code will be copied from the SPI ROM to the Ethernet buffer at address 0x8001_4000 with a length of 2048 bytes. Code execution will start at 0x8001_4000 (MAC base + 0x4000). The ARM Core will be in SVC mode. At this point the user can use the code in the MAC buffer to load the rest of the image from the SPI ROM.

4.2.3 FLASH Boot

To enable FLASH boot, make sure that the boot configuration pins (see Table 5-1 on page 5-2) are configured for normal boot mode, as shown in Table 4-1. Also make sure that the FLASH word size is correct as shown in Table 4-1.

To boot from FLASH, put the ASCII "CRUS" or "SURC" value in the HeaderID at one of the following locations (this location will be referred to as FLASH base + 0x0):

0x1000_0000
0x2000_0000
0x3000_0000
0x6000_0000
0x7000_0000

Code execution will start at address FLASH base + 0x4. The ARM Core will be in SVC mode.

Alternatively, to boot from FLASH, put the ASCII "CRUS" or "SURC" value in the HeaderID at one of the following locations (this location will be referred to as FLASH base +0x1000):

0x1000_1000
0x2000_1000

Copyright 2007 Cirrus Logic

4

0x3000_1000
0x6000_0000
0x7000_0000

Code execution will start at address FLASH base + 0x0. The ARM Core will be in SVC mode.

Note: CSn6 is the recommended chip select for Flash when performing an Internal boot. CSn0 must be connected to Flash when performing an External boot.

4.2.4 SDRAM or SyncFLASH Boot

To enable SDRAM or SyncFLASH boot, make sure that the pins are configured for normal boot mode, as shown in Figure 4-2. If booting with SyncFLASH or a 32-bit SDRAM device, make sure the SDRAM or SyncFLASH word size is correct, as shown in Figure 4-2. If booting with a 16-bit SDRAM device, follow the suggested software sequence of commands, as shown in Figure 4-2.

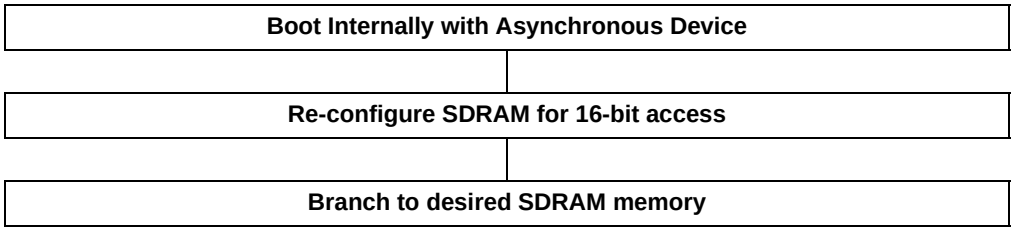


Figure 4-2. Flow chart of Boot Sequence for 16-bit SDRAM Devices

To boot from SDRAM or SyncFLASH, put the ASCII "CRUS" or "SURC" value in the HeaderID at one of the following locations (this location is Base + 0x0):

0xC000_0000
0xF000_0000

Code execution will start at address Base + 0x4. The ARM Core will be in SVC mode.

Alternatively, to boot from SDRAM or SyncFLASH, put the ASCII "CRUS" or "SURC" value in the HeaderID at one of the following locations (this is Base + 0x1000):

0xC000_1000
0xF000_1000

Code execution will start at address Base + 0x0. The ARM Core will be in SVC mode.

4.2.5 Synchronous Memory Operation

If running from Synchronous memory, before issuing a software reset, perform this procedure:

1. Run from SDRAM
2. Perform a software reset (SWRST bit in DEVCFG register)

Copyright 2007 Cirrus Logic

3. Run the internal boot code and boot from FLASH
4. Set the PLL back to use the external clock
5. Set up the SDRAM
6. Load the programs to SDRAM
7. Run from SDRAM

Copyright 2007 Cirrus Logic

Chapter 5
System Controller 5

5.1 Introduction

The System Controller (Syscon) provides:

- Clock control
- Power management
- System configuration management

These central resources are controlled by a set of software-locked registers, which can be used to prevent accidental accesses. Syscon generates the various bus and peripheral clocks and controls the system startup configuration.

5.1.1 System Startup

System startup begins with the assertion of a reset signal. There are five different categories of reset events. In order of decreasing effect, the reset events are:

- **PRSTn** (external pin for power-on reset)
- **RSTOn** (external pin for user reset)
- Three-key reset externally generated by a Keypad (behaves like user reset)
- Watchdog reset (internally generated)
- Software reset (internally generated)

During the time that any reset is active, the system is halted until it exits the reset state.

When the device starts with an external **PRSTn** or **RSTOn**, certain hardware configurations are determined, and some system configuration information will be recorded so that software can access it. See the details in “System Reset” on page 5-1 and “Hardware Configuration Control” on page 5-2.

5.1.2 System Reset

The device system reset consists of several events and signals. It has four levels of reset control:

- Power-on-reset, controlled by **PRSTn** pin. It resets the entire processor with no exceptions.
- User reset, controlled by **RSTOn** pin. While active, it resets the entire processor, except

Copyright 2007 Cirrus Logic

certain system variables such as RTC, SDRAM refresh control/global configuration, and the Syscon registers.

Note: If PLLs are enabled, user reset does NOT disable or reset the PLLs. They retain their frequency settings.

- Three-key reset. When F2, F4, and F7 are pressed, a user reset occurs.
- Software reset and watchdog reset. They perform the functions of the user reset, but are under software control.

"Watchdog" on page 19-3 and "PwrSts" on page 5-14 registers contain the information regarding which reset event occurred. Note that only the Watchdog timer contains information about a user-generated 3-key reset.

5.1.3 Hardware Configuration Control

The Hardware Configuration controls provide a mechanism to place the system into various boot configurations. In addition, one of several external boot memory options can be selected at system wake up.

The Hardware Configuration controls are defined by a set of device pins that are latched into configuration control bits on the rising edge of the **PRSTn** or **RSTOn** pin. The different hardware configuration bits define watchdog behavior, boot mode (internal or external), boot synchronicity, and external boot width. The latched pins are described in Table 5-1.

Table 5-1. Hardware Configuration Control Latched Pins

Pin Name(s)	Action
CSn[1]	Enable/Disable Watchdog reset timer
CSn[2]	Enable/Disable Watchdog reset duration
CSn[3]	Should be pulled-up to "1"
EECLK	Select internal or external boot
EEDAT	Should be pulled-up to "1"
BOOT[1:0]	Select boot mode
ASDO	Select synchronous or asynchronous boot
CSn[7:6]	Select external boot width

The latched version of these signals have an "L" prefix, are stored in the SysCfg register, and are readable by software. Note that the signals **EECLK** and **EEDAT** may have 1 kΩ pull-up resisters if used in an open-drain two-wire serial port application. (The default state assignments will assume these pull-ups.)

The Hardware Control configurations are show in Table 5-2.

Copyright 2007 Cirrus Logic

5

The normal boot function is described in Chapter 4 on page 4-1.

Serial boot is functionally identical to normal boot except that the SBoot bit in the SysCfg register is set. This mode is available for a software configuration option that is readable by the boot code.

In either normal boot or serial boot mode, once the processor starts up, it will begin to execute the instruction at logical address 0x0000_0000. Various configuration options are provided to select a memory device for booting from at address location 0. The options are listed in Table 5-2.

Table 5-2. Boot Configuration Options

EECLK	EEDAT	BOOT1	BOOT0	ASDO	CSn[7:6]	Boot Configuration
0	1	0	0	1	0 0 0 1 1 0 1 1	External boot fusing Sync boot mode and SDCSn3. The media type must be either SROM or SyncFLASH. The selection of the bus width is determined by latched **CSn[7:6]** value: 16-bit 16-bit 32-bit 32-bit
0	1	0	0	0	0 0 0 1 1 0 1 1	External boot using Async boot mode and CSn0. The selection of the bus width is determined by latched **CSn[7:6]** value: 8-bit 16-bit 32-bit 32-bit
1	1	0	1	x	xx	Internal boot from UART1.
1	1	0	0	x	xx	Internal SPI boot if HeaderID is found.
1	1	0	0	1	0 0 0 1 1 0 1 1	Internal boot using Sync boot mode at the chip select where the HeaderID exists. The selection of the bus width is determined by latched **CSn[7:6]** value: 16-bit 16-bit 32-bit 32-bit See memory map in Table 2-7 on page 2-16 for SYNC boot mode.
1	1	0	0	0	0 0 0 1 1 0 1 1	Internal boot using Async boot mode at the chip select where the HeaderID exists. The selection of the bus width is determined by latched **CSn[7:6]** value: 8-bit 16-bit 32-bit 32-bit See memory map in Table 2-7 on page 2-16 for ASYNC boot mode.

Copyright 2007 Cirrus Logic

Note: ASYNC boot mode is the preferred boot mode type for new designs.

5

5.1.4 Software System Configuration Options

There are several system configuration options selectable by the DeviceCfg and SysCfg registers. These registers provide the selection of several pin multiplexing options and also provide software access to the system reset configuration options. Please refer to the descriptions of the registers, "DeviceCfg" on page 5-25 and "SysCfg" on page 5-34, for a detailed explanation.

5.1.5 Clock Control

The EP93xx uses a flexible system to generate required clocks. The clock system generates up to 20 independent clock frequencies, some with very tight accuracy requirements, all from a single external low-frequency crystal or other external clock source. The ARM Core is designed so that once it has been configured, its CPU speed, bus speeds, and video clocks may be set to a number of different speeds without affecting the speeds of other clocks in the processor.

5.1.5.1 Oscillators and Programmable PLLs

The EP93xx has an interface to two external crystal oscillators: 32.768 KHz and 14.7456 MHz. To generate the required high-frequency clocks, the processor uses two phase-locked-loops (PLLs) to multiply the incoming 14.7456 MHz low frequency signal to much higher frequencies that are then divided down by programmable dividers to produce needed clocks. The PLLs operate independently of one another.

Figure 5-1 shows the PLL1 structure used in the EP93xx. Since PLL2 is identical to PLL1, wherever the phrase "PLL1" is used in the figure, it applies to PLL2 as well.

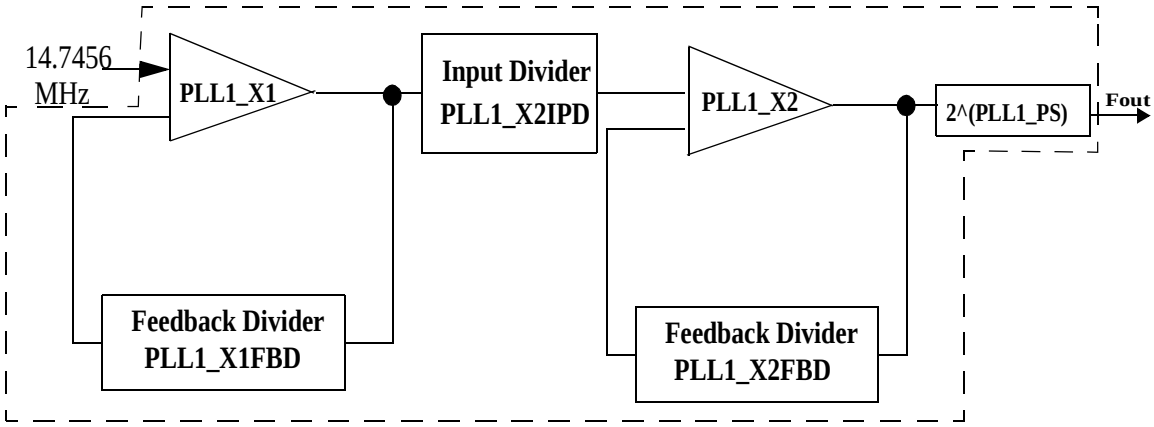


Figure 5-1. Phase Locked Loop (PLL) Structure

Copyright 2007 Cirrus Logic

Both PLLs are software programmable (each value is defined in "ClkSet1" on page 5-18 and "ClkSet2" on page 5-20 registers, respectively). The frequency of output clock Fout is determined by:

$$Fout = 14.7456MHz \cdot \frac{(PLL1_X1FBD + 1) \times (PLL1_X2FBD + 1)}{(PLL1_X2IPD + 1) \times 2^{PLL1_PS}}$$

Here PLL1_X1FBD, PLL1_X2FBD, PLL1_X2IPD and PLL1_PS are the bit fields in the "ClkSet1" register. The user must be aware of the requirements of PLL operation. They are:

- PLL1_X1 desired reference clock frequency range is > 11.058 MHz and < 200 MHz
- PLL1_X1 output frequency range is > 294 MHz and < 368 MHz
- PLL1_X2 desired reference clock frequency (after PLL1_X2IPD divider) is > 12.9 MHz and < 200 MHz.
- PLL1_X2 output, BEFORE the PS divide, must be > 290 MHz and <= 528 MHz

The same conditions apply to PLL2 and the "ClkSet2" register.

5.1.5.2 Bus and Peripheral Clock Generation

Figure 5-2 illustrates the clock generation system.

Copyright 2007 Cirrus Logic

5

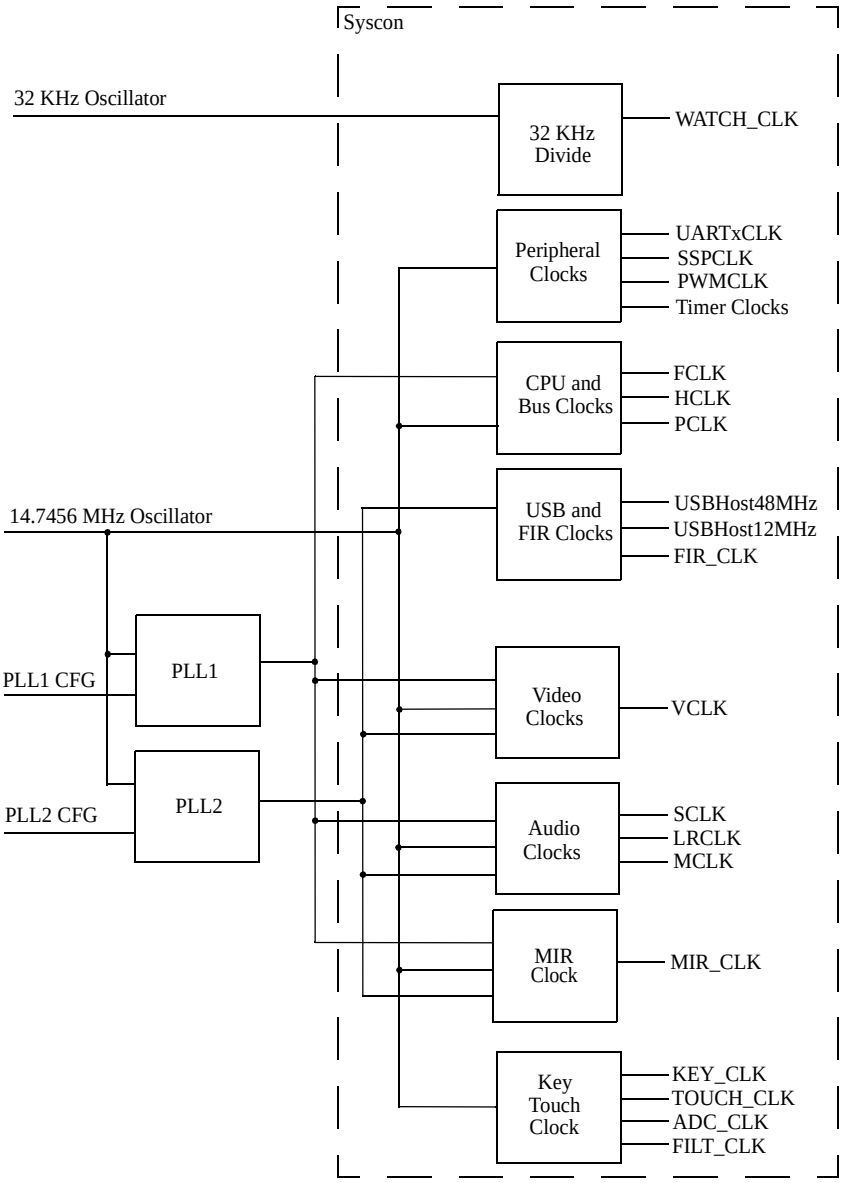


Figure 5-2. Clock Generation System

5.1.5.2.1 Bus Clock Generation

Figure 5-3 shows the generated clocks: the CPU clock (FCLK), the AHB bus clock (HCLK), and the APB bus clock (PCLK).

Copyright 2007 Cirrus Logic

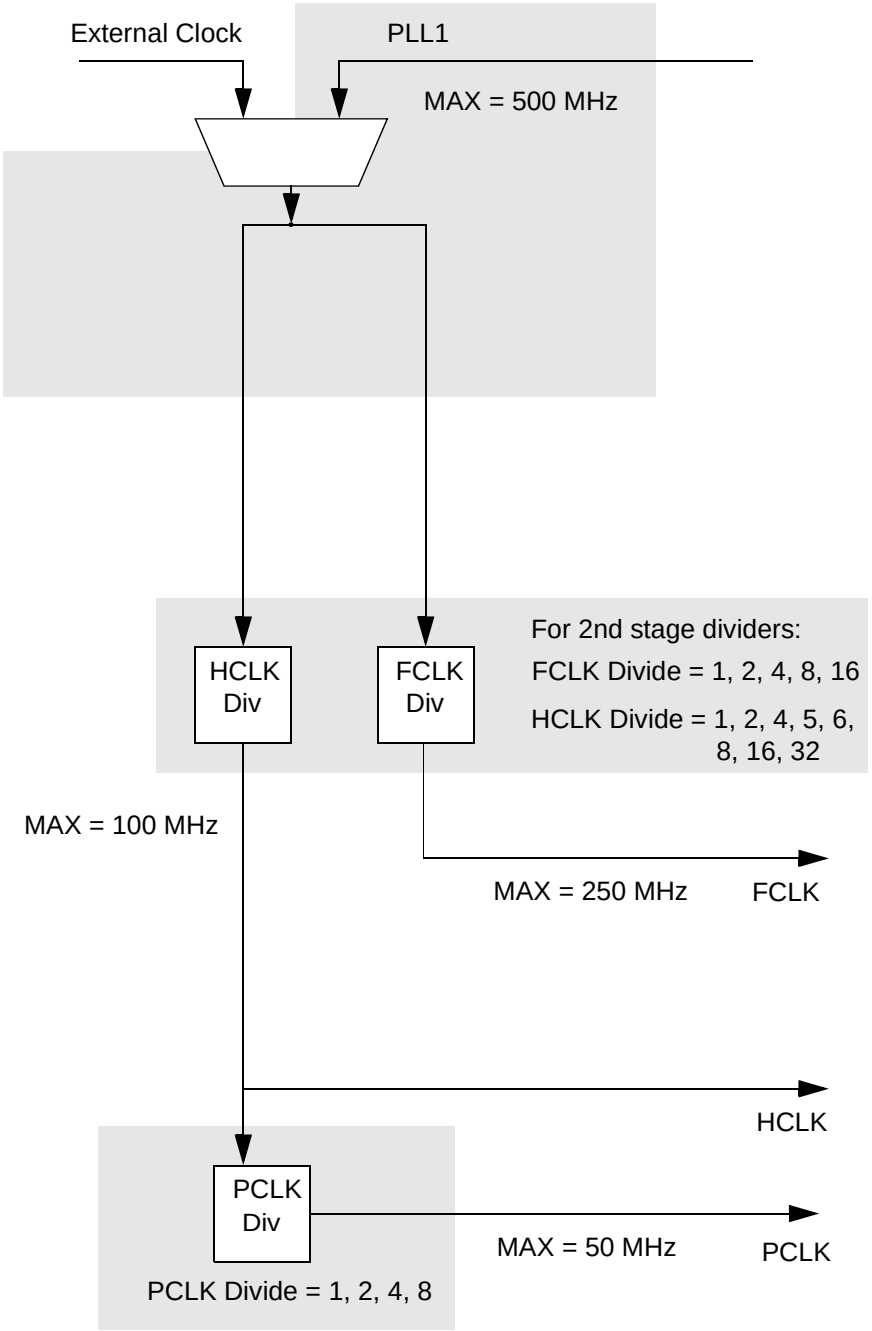


Figure 5-3. Bus Clock Generation

There are some limitations of each clock. FCLK must be <=200 MHz, HCLK<=100 MHz and PCLK<=50 MHz and FCLK >= HCLK > PCLK. Refer to register, "ClkSet1" on page 5-18, for the detailed configuration information regarding the divider bit fields.

Copyright 2007 Cirrus Logic

Even though FCLK is the usual CPU clock, HCLK can optionally be used instead. Processor clocking modes are:

- Async mode
- Sync mode
- Fast Bus mode

Both Async mode and Sync mode use FCLK. FCLK can be faster than HCLK, which would yield higher performance. Async mode and Sync mode have different clock skew requirements between FCLK and HCLK, and therefor have different throughput penalties due to clock synchronization. Fast Bus mode bypasses FCLK, and the CPU runs from HCLK. In this mode, the ARM Core potentially has lower performance than with the other two modes. **When the ARM Core starts up, it defaults to Fast Bus mode.** (The selection of clocking modes is determined by the iA and nF bits in ARM co-processor 15 register 1.)

5.1.5.2.2 Peripheral Clock Generation

The MCLK, VCLK, and MIR_CLK generators are three identical blocks. Each block contains a pre-divider of 2, 2.5 and 3 followed by a 7-bit programmer divider. The audio clock SCLK and LRCLK are further divided down from MCLK. The registers, "MIRClkDiv" on page 5-30, "VidClkDiv" on page 5-29, and "I2SClkDiv" on page 5-31, show the details.

USB uses a 48 MHz clock generated by PLL2. USBDIV, in register "ClkSet2" on page 5-20, is used to divide the frequency down from the PLL2 output.

The Key Matrix and Touchscreen Controller clocks are generated from an external 14.7 MHz oscillator. A chain of dividers generates divide-by-2, 4, 8, 16, 32, 64 versions of external oscillator clock. Programmable bits in the "KeyTchClkDiv" on page 5-32 select either a divide-by-4 or a divide-by-16 version of the external oscillator clock for each of the Key Matrix clock and Touchscreen controller.

Table 5-3 describes the speeds and sources for the various clocks.

Table 5-3. Clock Speeds and Sources

Block	Clocks Used	Clock Source
SSP	7.3728 MHz	Divided by 2 from 14.7456 MHz external oscillator
UART1 UART2 UART3	14.7456 MHz 7.3728 MHz	Both are derived from 14.7456 MHz external oscillator
PWM	14.7456 MHz	From the 14.7456MHz external oscillator
AAC	2.9491 MHz	Divided-by-5 from the 14.7456MHz external oscillator
Timers	508.4689 KHz 1.9939 KHz 983 KHz	All divided by the 14.7456 MHz external oscillator
Watchdog	256 Hz	Tap from the 32 KHz RTC clock

Copyright 2007 Cirrus Logic

5.1.5.3 Steps for Clock Configuration

The boot ROM must contain code that performs the following steps for a 14.7456 MHz crystal. The actual register values should be taken from the register descriptions for the desired clock setup.

1. After power up, the reset state of all clock control registers (all bits zero) will ensure that FCLK and HCLK are running at the crystal oscillator frequency of 14.7456 MHz.
2. Configure PLL1 to multiply by the desired value, set HCLK and FCLK rates, and power it up. To do this: write the proper value (taken from the register table) to "ClkSet1" immediately followed by 5 NOP instructions to flush the ARM Core's instruction pipeline. The ARM Core will go into Standby mode while PLL1 stabilizes, then it returns to normal operation at the new clock rates.
3. Configure PLL2 to multiply by the desired value. To do this, write the proper value to "ClkSet2".
4. Wait for PLL2 to stabilize (at least 1 ms)
5. Program all other clock dividers to the desired values and enable them. The clocks won't actually begin running until the clock sources which feed them are enabled. Write the desired values to these registers:
   - "VidClkDiv" on page 5-29
   - "MIRClkDiv" on page 5-30
   - "I2SClkDiv" on page 5-31
   - "KeyTchClkDiv" on page 5-32
6. All peripherals are now running from divided PLL outputs. Once the clocks have been configured, the frequency of any peripheral clock can be changed on-the-fly. To do this, perform a write to the clock register with the new divisor value and then set the appropriate enable bit. This ensures a problem-free change of the clock.

5.1.6 Power Management

The device follows a power-saving design plan. Power management is done by either altering the PLLs or the clock system frequency or by shutting off clocks to unused blocks. Also, there are several system power states to which the device can transition in order to save power. Care must be taken to ensure the clock system is not put into a non-operational state and that clock system dependencies are observed.

5.1.6.1 Clock Gatings

The list of peripherals with PCLK gating is shown Table 5-4. Refer to the appropriate chapter in this user's guide to find detailed information about clock gatings for a specific peripheral.

Copyright 2007 Cirrus Logic

Table 5-4. Peripherals with PCLK Gating

Peripheral	Peripheral/PCLK on with Enable or Register Access	PCLK on with Register Access Only	PCLK Continuous
UART1	**x**	-	-
UART2	**x**	-	-
UART3	**x**	-	-
KEYPAD	-	**x**	-
IRDA	**x**	-	-
SEC	**x**	-	-
I^2S	**x**	-	-
Watchdog	-	-	**x**
TSC	-	**x**	-
PWM	**x**	-	-
AAC	**x**	-	-
SSP	**x**	-	-
RTC	-	-	**x**
GPIO	-	**x**	-

HCLK to the USB Hosts can be gated off as well to further save power. The USH_EN bit in the "PwrCnt" register serves the purpose.

5.1.6.2 System Power States

The EP93xx has three power states:

- Run mode: Normal operation mode.
- Halt: ARM Core stops executing instructions.
- Standby: Power is on, but only SDRAM self-refresh and the RTC run.

Figure 5-4 illustrates the transitions among power states.

Copyright 2007 Cirrus Logic

5

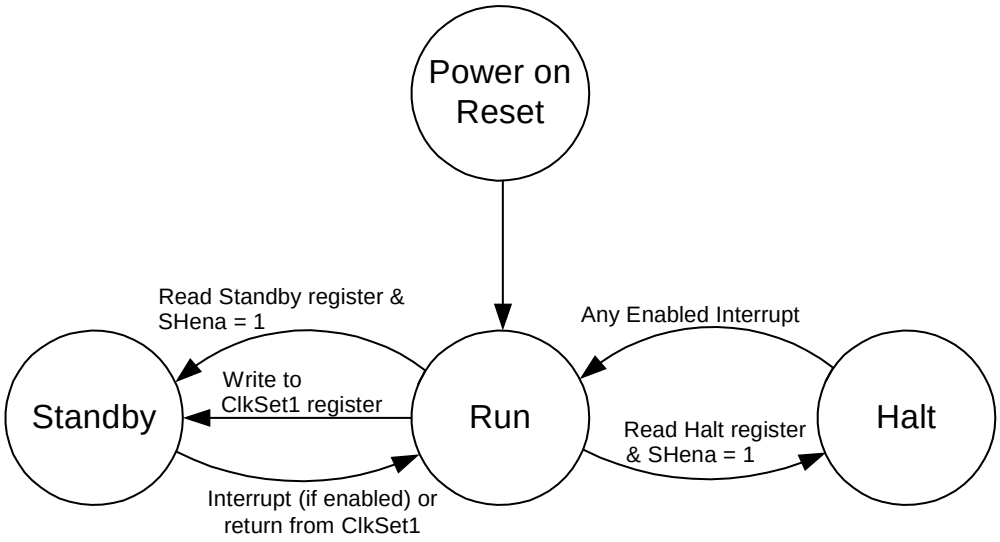


Figure 5-4. Power States and Transitions

5.1.6.2.1 Power-on-Reset Run

After power-on-reset, the ARM Core is automatically in run mode.

5.1.6.2.2 Run Standby Mode

Once in run mode, it is possible to move to the Standby state under these conditions:

- A read from the Standby register location 0x8093_000C when the SHena bit in the "DeviceCfg" register is set to 1. This triggers the system to enter STANDBY mode.
- A write to the "ClkSet1" register.

When the SHena bit is set to 1 and the user reads the Standby register location 0x8093_000C, the EP93xx is forced to transition into the Standby state. After this transition, the state controller will hold the Standby state before re-loading and allowing transition to the Run state.

A write to the "ClkSet1" register will also trigger the system to go into Standby mode. However, the system will automatically come back to normal operation after new clock settings take effect. The amount of time the EP93xx remains in the Standby state depends on whether the PLL is enabled, or if the EP93xx is using the external clock. If the PLL is enabled, the EP93xx will remain in Standby until the PLL is locked. If the EP93xx is in PLL bypass mode (nBYP1 = 1), then the EP93xx will remain in the Standby state for One to two 16.384 kHz clock cycles. This is to ensure a minimum 'off' time. The 16.384 kHz clock, derived from the 32.768 kHz clock, times how long the EP93xx remains in the Standby state.

When the EP93xx normally enters Standby mode, the SDRAM controller puts the external SDRAM into self-refresh before disabling its clocks (see "SDRAM Self Refresh" on page 13-8). This condition is only true if the refresh enable bit (RFSHEN) in the SDRAM controller is

Copyright 2007 Cirrus Logic

set. One example of this is when a power-on-reset is applied and this register bit is cleared. This means that this bit will not be set on boot-up and will have to be set to maintain the memory image for when the device re-enters Standby mode.

5.1.6.2.3 RUN HALT mode

A transition from Run mode to Halt mode is caused by reading the Halt register location 0x8093_0008 with the SHena bit set to 1. This has the effect of gating the CPU clock (FCLK) bus interface, with the APB/AHB system clock, and Memory/DMA system remaining enabled.

5.1.6.2.4 STANDBY RUN mode

There are normally several conditions in which the device can move from Standby mode to Run mode.

These conditions are:

- A falling edge on IRQ interrupt
- A falling edge on FIQ interrupt
- An exit from a "ClkSet1" write
- **PRSTn**
- **RSTOn**

The EP93xx comes out of Standby if an interrupt occurs or when an exit from a ClkSet1 write occurs. If a write is performed to the ClkSet1 register, the EP93xx then enters Standby mode and then automatically comes out of Standby mode and back into the Run state.

5.1.6.2.5 HALT RUN mode

The transition from the Halt state to the Run state is caused by:

- A falling edge on IRQ interrupt
- A falling edge on FIQ interrupt
- **RSTOn**

5.1.7 Interrupt Generation

The Syscon block generates two interrupts: TICK interrupt and Watchdog Expired interrupt.

The block generates the TICK interrupt based upon the 64 Hz clock, which is derived from the 32.768 KHz oscillator. The interrupt becomes active on every rising edge of the internal 64 Hz clock. It can be cleared by writing to the TEOI location.

Watchdog Expired interrupt becomes active on a rising edge of the 64 Hz TICK clock, if the TICK interrupt is still active. In other words, if a TICK interrupt has not been served for a complete TICK period, a watchdog expired interrupt is generated. It can be cleared by writing to the TEOI location as well.

Copyright 2007 Cirrus Logic

5

5.2 Registers

This section contains the detailed register descriptions for registers in the Syscon block. Table 5-5 shows the address map for the registers in this block, followed by a detailed listing for each register.

Table 5-5. Syscon Register List

Address	Name	SW Locked	Type	Size	Description
0x8093_0000	PwrSts	No	R	32	Power/state control state
0x8093_0004	PwrCnt	No	R/W	32	Clock/Debug control status
0x8093_0008	Halt	No	R	32	Reading this location enters Halt mode.
0x8093_000C	Standby	No	R	32	Reading this location enters Standby mode.
0x8093_0018	TEOI	No	W	32	Write to clear Tick interrupt
0x8093_001C	STFClr	No	W	32	Write to clear CLDFLG, RSTFLG and WDTFLG.
0x8093_0020	ClkSet1	No	R/W	32	Clock speed control 1
0x8093_0024	ClkSet2	No	R/W	32	Clock speed control 2
0x8093_0040	ScratchReg0	No	R/W	32	Scratch register 0
0x8093_0044	ScratchReg1	No	R/W	32	Scratch register 1
0x8093_0050	APBWait	No	R/W	32	APB wait
0x8093_0054	BusMstrArb	No	R/W	32	Bus Master Arbitration
0x8093_0058	BootModeClr	No	W	32	Boot Mode Clear register
0x8093_0080	DeviceCfg	Yes	R/W	32	Device configuration
0x8093_0084	VidClkDiv	Yes	R/W	32	Video Clock Divider
0x8093_0088	MIRClkDiv	Yes	R/W	32	MIR Clock Divider, divides MIR clock for MIR IrDA
0x8093_008C	I2SClkDiv	Yes	R/W	32	I^2S Audio Clock Divider
0x8093_0090	KeyTchClkDiv	Yes	R/W	32	Keyscan/Touch Clock Divider
0x8093_0094	ChipID	Yes	R/W	32	Chip ID Register
0x8093_009C	SysCfg	Yes	R/W	32	System Configuration
0x8093_00A0	-	-	-	-	Reserved
0x8093_00C0	SysSWLock	No	R/W	1 bit	Software Lock Register

Copyright 2007 Cirrus Logic

Register Descriptions

5

PwrSts

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CHIPMAN								CHIPID							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDTFLG	RSVD	CLDFLG	TEST_ RESET	RSTFLG	SW_ RESET	PLL2_ LOCK_REG	PLL2_ LOCK	PLL1_ LOCK_REG	PLL1_ LOCK	RTCDIV					

Address:

0x8093_0000 - Read Only

Definition:

The PwrSts system control register is the Power/State control register.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RTCDIV: The 6-bit RTCDIV shows the number of 64-seconds which have elapsed. It is the output of the divide-by-64 chain that divides the 64 Hz TICK clock down to 1 Hz though showing an incrementing count. The MSB is the 1 Hz output; the LSB is the 32 Hz output. It is reset by power-on-reset to 000000b.

PLL1_LOCK: PLL1 lock. This signal goes high when PLL1 is locked and it is at the correct frequency.

PLL1_LOCK_REG: Registered PLL1 lock. This is a one-shot registered signal of the PLL1_LOCK signal. It is only cleared on a power-on-reset, when the device enters the Standby state or when PLL1 is powered down.

PLL2_LOCK: PLL2 lock. This signal goes high when PLL2 is locked, and it is at the correct frequency.

PLL2_LOCK_REG: Registered PLL2 lock. This is a one-shot registered signal of the PLL2_LOCK signal. It is only cleared on a power-on-reset, when ClkSet2 is written, the device enters the Standby state, or PLL2 is powered down.

SW_RESET: Software reset flag. This bit is set if the software reset has been activated. It is cleared by writing to the STFClr location. On power-on-reset, it is reset to 0b.

Copyright 2007 Cirrus Logic

5

RSTFLG: Reset flag. This bit is set if the user reset button has been pressed; forcing the **RSTOn** input low. It is cleared by writing to the STFClr location. On power-on-reset, it is reset to 0b.

TEST_RESET: Test reset flag. This bit is set if the test reset has been activated; it is cleared by writing to the STFClr location. On power-on-reset, it is reset to 0b.

CLDFLG: Cold start flag. This bit is set if the device has been reset with a power-on-reset; it is cleared by writing to the STFClr location. On power-on-reset, it is set to 1b.

WDTFLG: Watchdog Timer flag. This bit is set if the Watchdog timer resets the system. It is cleared by writing to the STFClr location. It is reset to 0.

CHIPID: Chip ID bits. This 8-bit register determines the Chip Identification for the device. For the device, this value is 0x20.

CHIPMAN: This 8-bit register determines the Chip Manufacturer ID for the device. For the device, this value is 0x43.

PwrCnt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIR_EN	**RSVD**	**UART BAUD**	**USH_EN**	**DMA M2M CH1**	**DMA M2M CH0**	**DMA M2P CH8**	**DMA M2P CH9**	**DMA M2P CH6**	**DMA M2P CH7**	**DMA M2P CH4**	**DMA M2P CH5**	**DMA M2P CH2**	**DMA M2P CH3**	**DMA M2P CH0**	**DMA M2P CH1**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															

Address:

0x8093_0004 - Read / Write

Definition:

The PwrCnt system control register is the Clock/Debug control status register.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

Copyright 2007 Cirrus Logic

DMA M2M/P CHx: These bits enable the clocks to the DMA controller channels. Note that a channels-enable bit MUST be asserted before any register within the DMA controller can be read or written. At least one ARM instruction cycle must occur between writing to this register to enable the DMA Controller channel and actually accessing it. The number of cycles will depend on the setting of HCLK and PCLK division in the "ClkSet1" or "ClkSet2" register. To save power, ensure that all these bits are disabled (low) if the DMA controller is not being used. On a system reset, the register will be reset to zero.

USH_EN: This bit is used to gate the HCLK to the USB Host block in order to save power. It is reset to zero, thus gating off the HCLK. It can be set to one to turn on the HCLK to the USB Host. This bit must be set before any register within the USB Host can be accessed. At least one ARM instruction cycle must occur between writing to this register bit and actually accessing the USB Host. The number of cycles will depend on the setting of HCLK and PCLK division in the "ClkSet1" and "ClkSet2" register.s

This bit is also used to gate the 48 MHz and 12 MHz clocks to the USB Host block in order to save power. It is reset to zero, thus gating off the USB Host clocks. By setting this to one, the USB Host clocks are enabled. At least one ARM instruction cycle must occur between writing to this register bit and actually accessing the USB Host. The number of cycles will depend on the wake-up time for PLL2. To find out if PLL2 has locked on to its frequency, the PLL2_LOCK bit in the PwrSts register can be read.

UARTBAUD: This bit controls the clock input to the UARTs. When cleared, the UARTs are driven by the 14.7456 MHz clock divided by 2 (7.3728 MHz). This gives a maximum baud-rate of 230 Kbps. When set, the UARTs are driven by the 14.7456 MHz clock directly, giving an increased maximum baud rate of 460 Kbps. This bit is 0 on reset.

FIR_EN: This bit is used to gate the FIRCLK to the IrDA block in order to save power. It is reset to zero, thus gating off the FIRCLK. Setting this bit to one will turn on the 48 MHz clock to the IrDA.

Copyright 2007 Cirrus Logic

Standby and Halt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															

Address:

Standby - 0x8093_000C - Read Only
Halt - 0x8093_0008 - Read Only

Definition:

The Standby and Halt registers allow entry into the power saving modes. A read to the Halt location will initiate a request for the system to enter Halt mode, if the SHena bit is set in the DeviceCfg register in Syscon. Likewise a read to Standby will request entry into Standby only when the SHena bit is set.

Note: When a read is performed to the Standby location, it must be immediately followed by 5 NOP instructions. This is needed to flush the instruction pipeline in the ARM920T core. Writes to these locations have no effect.

Bit Descriptions:

RSVD: There are no readable bits in this register.

TEOI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															

Address:

0x8093_0018 - Write

Definition:

Writing to the TEOI location will clear the periodic Watchdog expired interrupt (WEINT) and the 64 Hz TICK interrupt (TINT). Any data written to the register triggers the clearing.

Bit Descriptions:

RSVD: There are no readable bits in this register.

Copyright 2007 Cirrus Logic

5

STFClr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															

Address:

0x8093_001C - Write

Definition:

Writing to the STFClr location will clear the CLDFLG, WDTFLG and RSTFLG in the register, "PwrSts" on page 5-14. Any data written to the register triggers the clearing.

Bit Descriptions:

RSVD: There are no readable bits in this register.

ClkSet1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				FCLK DIV			SMC ROM	nBYP1	HCLK DIV			PCLK DIV		PLL1_PS	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL1 X1FBD1				PLL1 X2FBD2							PLL1 X2IPD				

Address:

0x8093_0020 - Read/Write

Definition:

The ClkSet1 system control register is one of two register that control clock speeds.

Note: When a write is performed to the ClkSet1 location, it must be immediately followed by 5 NOP instructions. This is needed to flush the instruction pipeline in the ARM920T core. Writing to this register will cause the the device to enter Standby for between 8 ms to 16 ms. Reading from this register will not cause an entry into Standby mode.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

PLL1_X2IPD: These 5 register bits set the input divider for PLL1 operation. On power-on-reset the value is set to 00111b (7 decimal).

Note: The value in the register is the actual coefficient minus one.

Copyright 2007 Cirrus Logic

5

PLL1_X2FBD2: These 6 register bits set the first feedback divider bits for PLL1. On power-on-reset the value is set to 000111b (7 decimal).

Note: The value in the register is the actual coefficient minus one.

PLL1_X1FBD1: These 5 register bits set the second feedback divider bits for PLL1. On power-on-reset the value is set to 10011b (19 decimal).

Note: The value in the register is the actual coefficient minus one.

PLL1_PS: These two bits determine the final divide on the VCO clock signal in PLL1.
00 - Divide by 1
01 - Divide by 2
10 - Divide by 4
11 - Divide by 8

On power-on-reset these bits are reset to 11b (3 decimal).

Note: This means that PLL1 FOUT is programmed to be 36,864,000 Hz on startup.

Note: The value in the register is the actual coefficient minus one.

PCLKDIV: These two bits set the divide ratio between the HCLK AHB clock and the APB clock (PCLK)
00 - Divide by 1
01 - Divide by 2
10 - Divide by 4
11 - Divide by 8

On power-on-reset the value is set to 00b.

Note: Care must be taken to make the correct selection of PCLK divide for the HCLK frequency used, so that the required minimum ratio between PCLK and the peripheral clock is not violated

HCLKDIV: These three bits set the divide ratio between the VCO output and the bus clock (HCLK)

000 - Divide by 1	100 - Divide by 6
001 - Divide by 2	101 - Divide by 8
010 - Divide by 4	110 - Divide by 16
011 - Divide by 5	111 - Divide by 32

On power-on-reset the value is set to 000b.

Copyright 2007 Cirrus Logic

nBYP1: This bit selects the clock source for the processor clock dividers. With this bit clear, the system wakes up and boots with the PLL bypassed and uses an external clock source. With nBYP1 set, the system runs with the PLL generated clock. The default for this bit is to boot/run from external clock source.

SMCROM: If set, this bit will gate off the HCLK to the Static Memory Controller when in Halt mode and therefore save power. When in Halt mode, there are no Instruction Code fetches occurring and therefore if there are no DMA operations in progress that may require the SMC, there will be no accesses to this controller. It may therefore be safely disabled when in Halt mode. This bit is 0b on reset.

FCLKDIV: These three bits set the divide ratio between the VCO output and processor clock. On power-on-reset the value is set to 000b.

000 - Divide by 1
001 - Divide by 2
010 - Divide by 4
011 - Divide by 8
100 - Divide by 16

For FCLKDIV values equal to 1xxb (except for 100b), the divide ratio will be divide by 1.

ClkSet2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
USB DIV				RSVD								nBYP2	PLL2_EN	PLL2_PS	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL2 X1FBD1				PLL2 X2FBD2								PLL2 X2IPD			

Address:

0x8093_0024 - Read/Write

Definition:

The ClkSet2 register is used for setting the dividers internally to PLL2 and to the USB Host divider. The reset setting for PLL2 creates a frequency of 48 MHz. The default divider for USB_DIV is divide by 1, which will produce the USB host clock frequency and FIR clock frequency of 48 MHz.

Bit Descriptions:

PLL2_X2IPD: These 5 register bits set the input divider for PLL2 operation. On power-on-reset the value is set to 10111b (23 decimal).

Note: The value in the register is the actual coefficient minus one.

Copyright 2007 Cirrus Logic

PLL2_X2FBD2: These 6 register bits set the first feedback divider bits for PLL2. On power-on-reset the value is set to 11000b (24 decimal).

Note: The value in the register is the actual coefficient minus one.

PLL2_X1FBD1: These 5 register bits set the second feedback divider bits for PLL2. On power-on-reset the value is set to 11000b (24 decimal).

Note: The value in the register is the actual coefficient minus one.

PLL2_PS: These two bits determine the final divide function on the VCO clock signal in PLL2.
00 - Divide by 1
01 - Divide by 2
10 - Divide by 4
11 - Divide by 8

On power-on-reset these bits are reset to 11b (3 decimal).

Note: This means that PLL2 FOUT is programmed to be 48,000,000 Hz on startup.

Note: The value in the register is the actual coefficient minus one.

PLL2_EN: This bit enables PLL2. If set, PLL2 is enabled. If this bit is zero, PLL2 is disabled. On power-on-reset the value is set to 0b.

nBYP2: This bit selects the clock source for the processor clock dividers. If set, PLL2 is the clock source. If this bit is set to zero, the external clock is the clock source. On power-on-reset, this bit defaults to 0b.

USBDIV: These four bits set the divide ratio between the PLL2 output and the USB clock.

0000 - Divide by 1	1000 - Divide by 9
0001 - Divide by 2	1001 - Divide by 10
0010 - Divide by 3	1010 - Divide by 11
0011 - Divide by 4	1011 - Divide by 12
0100 - Divide by 5	1100 - Divide by 13
0101 - Divide by 6	1101 - Divide by 14
0110 - Divide by 7	1110 - Divide by 15
0111 - Divide by 8	1111 - Divide by 1

On power-on-reset these bits are reset to 0000b.

Copyright 2007 Cirrus Logic

ScratchReg0, ScratchReg1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Value															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value															

Address:

ScratchReg0 - 0x8093_0040, Read/Write
ScratchReg1 - 0x8093_0044, Read/Write

Default:

0x0000_0000

Definition:

Each of these locations provide a 32-bit read/write scratch register, that can be used as a general purpose storage. These registers are reset to zero only on a power-on-reset. A System Reset will have no effect.

Bit Descriptions:

Value: This is a 32-bit read/write location.

APBWait

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															NO_WRITE_WAIT

Address:

0x8093_0050, Read/Write

Definition:

The APBWait register controls the insertion of wait states for APB peripherals.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

NO_WRITE_WAIT: Used in the AHB/APB bridge to not insert an AHB wait during writes, if set. If reset, a wait state is added by forcing HREADY = 0 during ST_WRITE. This bit resets to 0x0001.

Copyright 2007 Cirrus Logic

BusMstrArb

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				RSVD		MAC ENFIQ	MAC ENIRQ	USH ENFIQ	USH ENIRQ	DMA_ ENFIQ	DMA_ ENIRQ	PRI CORE	RSVD	PRI_ORD	

Address:

0x8093_0054 - Read/Write

Definition:

The Bus Master arbitration register (BusMstrArb) is used to configure the AHB master priority order.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

PRI_ORD: Used to set the priority of the AHB arbiter. The priority order is shown in Table 5-6. This field resets to 00.

Table 5-6. Priority Order for AHB Arbiter

Priority Number	PRIOR 00 (Reset value)	PRIOR 01	PRIOR 10	PRIOR 11
1	Raster Cursor	Raster	Raster	Raster
2	MAC	Raster Cursor	Raster Cursor	DMA
3	USB	MAC	DMA	MAC
4	DMA	USB	USB	USB
5	ARM920T	ARM920T	MAC	Raster Cursor
6	Raster	DMA	ARM920T	ARM920T

PRI_CORE: When this bit is set the Core will become highest priority following a grant to one of the following: Raster, Raster Cursor, MAC, USB and DMA. If the Core then requests the bus, it is then placed in the priority order selected by PRI_ORD after it is granted, until one of the above masters is granted the bus, and is placed on top of the priority scheme.

DMA_ENIRQ: When set the arbiter will degrant DMA from the AHB bus and will ignore subsequent requests from DMA if an IRQ is active. When IRQ is cleared the DMA request is allowed again. There is no impact on other masters. Reset to 0.

Copyright 2007 Cirrus Logic

DMA_ENFIQ: When set the arbiter will degrant DMA from the AHB bus and will ignore subsequent requests from DMA if an FIQ is active. When FIQ is cleared the DMA request is allowed again. There is no impact on other masters. Reset to 0.

USH_ENIRQ: When set the arbiter will degrant USB host from the AHB bus and will ignore subsequent requests from the USB Host if an IRQ is active. When IRQ is cleared, the USB Host request is allowed again. There is no impact on other masters. Reset to 0.

USH_ENFIQ: When set the arbiter will degrant USB Host from the AHB bus and will ignore subsequent requests from USB Host if an FIQ is active. When FIQ is cleared, the USB Host request is allowed again. There is no impact on other masters. Reset to 0.

MAC_ENIRQ: When set the arbiter will degrant Ethernet MAC from the AHB bus and will ignore subsequent requests from the MAC if an IRQ is active. When IRQ is cleared, the MAC request is allowed again. There is no impact on other masters. Reset to 0.

MAC_ENFIQ: When set the arbiter will degrant the Ethernet MAC from the AHB bus and will ignore subsequent requests from the MAC if an FIQ is active. When FIQ is cleared, the MAC request is allowed again. There is no impact on other masters. Reset to 0.

BootModeClr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															

Address:

0x8093_0058 - Write Only

Definition:

The BootModeClr register is a write-to-clear register. Reset activates the boot ROM remap function causing the internal boot ROM to map to address zero, if internal boot is selected. Writing BootModeClr removes the internal ROM address remap, restoring normal address space.

Bit Descriptions:

RSVD: There are no readable bits in this register.

Copyright 2007 Cirrus Logic

DeviceCfg

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWRST	**D1onG**	**D0onG**	**IonU2**	**GonK**	**TonG**	**MonG**	**U3EN**	**CPENA**	**A2onG**	**A1onG**	**U2EN**	**EXVC**	**U1EN**	**TIN**	**RSVD**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HC3IN	**HC3EN**	**HC1IN**	**HC1EN**	**HonIDE**	**GonIDE**	**PonG**	**EonIDE**	**I2Son SSP**	**I2Son AC97**	**0**	**RASOn P3**	**RAS**	**ADCPD**	**KEYS**	**SHena**

Address:

0x8093_0080 - Read/Write, Software locked

Default:

0x0000_0000

Definition:

Device Configuration Register. This register controls the operation of major system functions.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

0: This bit must be written as "0".

SHena: Standby/Halt enable. When 1, allows the system to enter Standby or Halt on a read from the Standby and Halt registers, respectively.

KEYS: Key matrix inactive.
1 - Key Matrix controller inactive,
0 - Key Matrix controller active.

ADCPD: ADC Power Down.
1 - ADC and clocks are powered down.
0 - ADC and clocks are active. ADCPD must be zero for normal touch screen operation and for direct ADC operation.

RAS: Raster inactive.
1 - Disables video pixel clock to most of the Raster engine,
0 - Normal video clock to Raster engine.

RasOnP3: Raster On SDRAM Port 3.
1 - The Raster video refresh SDRAM accesses use the system primary AHB to get video data.
0 - Raster video refresh uses the private AHB on SDRAM Port 0.

Copyright 2007 Cirrus Logic

I2SonAC97: Audio - I²S on AC97 pins. The I²S block uses the AC97 pins. See Audio Interface pin assignments in Table 5-7.

Note: The I²S should be enabled on only one set of pins. Therefore I2SonAc97 and I2SonSSP are mutually exclusive. Setting both I2SonAc97 and I2SonSSP will cause unexpected behavior.

I2SonSSP: Audio - I²S on SSP pins. The I²S block uses the SSP pins. MCLK is not available in this pin option. See Audio Interface pin assignments in Table 5-7.

Note: The I²S should be enabled on only one set of pins. Therefore I2SonAc97 and I2SonSSP are mutually exclusive. Setting both I2SonAc97 and I2SonSSP will cause unexpected behavior.

Table 5-7. Audio Interfaces Pin Assignment

Pin Name	Normal Mode	I²S on SSP Mode	I²S on AC'97 Mode
	Pin Description	Pin Description	Pin Description
SCLK1	SPI Bit Clock	I²S Serial Clock	SPI Bit Clock
SFRM1	SPI Frame Clock	I²S Frame Clock	SPI Frame Clock
SSPRX1	SPI Serial Input	I²S Serial Input	SPI Serial Input
SSPTX1	SPI Serial Output	I²S Serial Output	SPI Serial Output
		(No I²S Master Clock)	
ARSTn	AC'97 Reset	AC'97 Reset	I²S Master Clock
ABITCLK	AC'97 Bit Clock	AC'97 Bit Clock	I²S Serial Clock
ASYNC	AC'97 Frame Clock	AC'97 Frame Clock	I²S Frame Clock
ASDI	AC'97 Serial Input	AC'97 Serial Input	I²S Serial Input
ASDO	AC'97 Serial Output	AC'97 Serial Output	I²S Serial Output

EonIDE: GPIO Port E on IDE pins:

0 - GPIO Port E used for IDE

1 - GPIO Port E used for GPIO

PonG: PWM 1 output on EGPIO pin

GonIDE: GPIO Port G on IDE pins

0 - GPIO Port G used for IDE

1 - GPIO Port G used for GPIO

HonIDE: GPIO Port H on IDE pins

Copyright 2007 Cirrus Logic

0 - GPIO Port H used for IDE

1 - GPIO Port H used for GPIO

HC3IN: HDLC3 clock in. This bit has no effect unless HC3EN is 1.
1 = pin EGPIO[3] is an input and drives an external HDLC clock to UART3.
0 = pin EGPIO[3] is an output driven by UART3.

HC3EN: HDLC3 clock enable.
1 = pin EGPIO[3] is used to for an HDLC clock with UART3.
0 = pin EGPIO[3] is not used.

HC1IN: HDLC1 clock in. This bit has no effect unless HC3EN is 0 and HC1EN is 1.
1 = pin EGPIO[3] is an input and drives an external HDLC clock to UART1.
0 = pin EGPIO[3] is an output driven by UART1.

HC1EN: HDLC1 clock enable. This bit has no effect unless HC3EN is 0.
1 = pin EGPIO[3] is used for an HDLC clock with UART1.
0 = pin EGPIO[3] is not used.

TIN: Touchscreen controller inactive.
1 - Touchscreen controller to inactive state,
0 - Touchscreen controller active.
To use the ADC converter independent of the Touch screen controller, the Touchscreen controller must be enabled and set inactive. The ADC can then be operated using the direct access registers. The TIN bit does not affect the ADC power state. ADC power down is directly controlled by the ADCPD bit.

U1EN: UART1 Enable.
1 - UART1 baud rate clock is active.
0 - UART1 clock is off.

EXVC: External Video Clock.
1 - Raster engine uses external pixel clock and the SPCLK pin is configured as an input,
0 - Raster engine uses internal pixel clock and the SPCLK pin is configured as an output.

U2EN: UART2 Enable.
1 - UART2 baud rate clock is active.
0 - UART2 clock is off.

Copyright 2007 Cirrus Logic

5

A1onG: I^2S Audio Port 1 on GPIO.
1 - I^2S Port 1 pins are mapped to EGPIO. SDI1 is on EGPIO[5], SDO1 is on EGPIO[4].
0 - EGPIO[5:4] are not used.

A2onG: I^2S Audio Port 2 on GPIO.
1 - I^2S Port 2 pins are mapped to EGPIO. SDI2 is on EGPIO[13], SDO2 is on EGPIO[6].
0 - EGPIO[13] and EGPIO[6] are not used.

CPENA: Co-processor Enable.
1 - MaverickCrunch co-processor is enabled.
0 - Co-processor is disabled and will not accept instructions.

U3EN: UART3 Enable.
1 - UART3 baud rate clock is active.
0 - UART3 clock is off.

MonG: Modem on GPIO.
1 - Modem support signals use EGPIO[0] pins.
0 - Modem support signals do not use EGPIO[0] pins

TonG: TENn on GPIO. This bit has no effect unless HC3EN and HC1EN are 0.
1 - UART3 TENn signal drives EGPIO[3].
0 - EGPIO[3] used by GPIO.

GonK: GPIO on Key Matrix.
1 - Key Matrix pins are configured for GPIO operation,
0 - Key Matrix pins are controlled by other options.
The GonK has precedence over the Key Matrix controller. The SPI0, when mapped to Key Matrix pins, has precedence over GPIO. When the Key Matrix pins are configured for SPI0, the pins unused by SPI0 can be used for GPIO.

IonU2: IrDA on UART2.
1 - UART2 is used as an IrDA interface,
0 - UART2 is a normal UART.

D0onG: External DMA0 hardware handshake signals mapped to EGPIO pins.
1 - Signals mapped.
0 - Signals not supported.

D1onG: External DMA1 hardware handshake signals mapped to EGPIO pins.
1 - Signals mapped.
0 - Signals not supported.

Copyright 2007 Cirrus Logic

SWRST: Software reset. A one to zero transition of this bit initiates a software reset.

VidClkDiv

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VENA	ESEL	PSEL	RSVD			PDIV		RSVD	VDIV						

Address:

0x8093_0084 - Read/Write, Software locked

Default:

0x0000_0000

Definition:

Configures video clock for the raster engine. Selects input to VCLK dividers from either PLL1 or PLL2, and defines a programmable divide value.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

VENA: Enable VCLK divider.

ESEL: External clock source select.
0 - use the external **XTALI** clock input as the clock source.
1 - use one of the internal PLLs selected by PSEL as the clock source.

PSEL: PLL source select.
1 - select PLL2 as the clock source.
0 - select PLL1 as the clock source.

PDIV: Pre-divider value. Generates divide by 2, 2.5, or 3 from the clock source.
00 - Disable clock
01 - Divide-by-2
10 - Divide-by-2.5
11 - Divide-by-3

VDIV: VCLK divider value. Forms a divide-by-N of the pre-divide clock output. VCLK is the source clock divided by PDIV divided by N. Must be at least two.

Copyright 2007 Cirrus Logic

MIRClkDiv

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MENA	ESEL	PSEL	RSVD			PDIV		RSVD	MDIV						

Address:

0x8093_0088 - Read/Write, Software locked

Default:

0x0000_0000

Definition:

Configures MIR clock for the MIR IrDA. Selects input to MIR clock dividers from either PLL1 or PLL2, and defines a programmable divide value.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

MENA: Enable MIR_CLK divider.

ESEL: External clock source select.
0 - Use the external **XTALI** clock input as the clock source.
1 - Use one of the internal PLLs selected by PSEL as the clock source.

PSEL: PLL source select.
1 - Select PLL2 as the clock source.
0 - Select PLL1 as the clock source.

PDIV: Pre-divider value. Generates divide by 2, 2.5, or 3 from the clock source.
00 - Disable clock
01 - Divide-by-2
10 - Divide-by-2.5
11 - Divide-by-3

MDIV: MIR_CLK divider value. Forms a divide-by-N of the pre-divide clock output. MIR_CLK is the source clock divided by PDIV divided by N.

Copyright 2007 Cirrus Logic

I2SClkDiv

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SENA	SLAVE	ORIDE	RSVD								DROP	SPOL	LRDIV		SDIV

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MENA	ESEL	PSEL	RSVD			PDIV		RSVD	MDIV						

Address:

0x8093_008C - Read/Write, Software locked

Default:

0x0000_0000

Definition:

Configures the I^2S block audio clocks **MCLK, SCLK**, and **LRCLK**.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

SENA: Enable audio clock generation.

SLAVE: I^2S slave. Configures the I^2S clock system to operate as a slave. **SCLK** and **LRCLK** are chip inputs. The clock configuration controls in this register are ignored in slave mode.

ORIDE: Override I^2S master configuration.
1 - Override the SAI_MSTR_CLK_CFG from the I^2S block and use the I2SClkDiv Register settings.
0 - Use the I2S SAI_MSTR_CLK_CFG signals.

DROP: Drop SCLK clocks.
1 - When in 64x mode, drop 8 SCLKs.
0 - Do not drop SCLKs.

SPOL: SCLK polarity. Defines the SCLK edge that aligns to LRCLK transitions.
1 - LRCLK transitions on the falling SCLK edge.
0 - LRCLK transitions on the rising SCLK edge.

LRDIV: LRCLK divide select.
00 - LRCK = SCLK / 32
01 - LRCK = SCLK / 64
10 - LRCK = SCLK / 128
11 - Reserved

Copyright 2007 Cirrus Logic

5

SDIV: SCLK divide select.
1 - SCLK = MCLK / 4,
0 - SCLK = MCLK / 2.

MENA: Enable master clock generation.

ESEL: External clock source select.
0 - Use the external **XTALI** clock input as the clock source.
1 - Use one of the internal PLLs selected by PSEL as the clock source.

PSEL: PLL source select.
1 - Select PLL2 as the clock source.
0 - Select PLL1 as the clock source.

PDIV: Pre-divider value. Generates divide by 2, 2.5, or 3 from the clock source.
00 - Disable clock
01 - Divide-by-2
10 - Divide-by-2.5
11 - Divide-by-3

MDIV: MCLK divider value. Forms a divide-by-N of the pre-divide clock output. **MCLK** is the source clock divided by PDIV divided by N.

KeyTchClkDiv

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSEN	RSVD														ADIV

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEN	RSVD														KDIV

Address:
0x8093_0090 - Read/Write, Software locked

Default:
0x0000_0000

Definition:
Configures the Key Matrix, Touchscreen, and ADC clocks. Touchscreen clock is a fixed divide-by-4 from the ADC clock. Touch Filter clock is a fixed divide-by-2 from the ADC clock.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

TSEN: Touchscreen and ADC clock enable

Copyright 2007 Cirrus Logic

ADIV:	ADC clock divider value. 0 - ADC Clock is divide-by-16 from the external oscillator. 1 - ADC Clock is divide-by-4 from the external oscillator.
KEN:	Key matrix clock enable. This clock is divided from the slow clock source.
KDIV:	Key matrix clock divider value. 0 - Key Matrix Clock is divide-by-16 from the external oscillator. 1 - Key Matrix Clock is divide-by-4 from the external oscillator.

CHIP_ID

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV				RSVD				0					RSVD	0	RSVD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID															

Address:

0x8093_0094 - Read Only

Definition:

Chip ID register.

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
REV:	Revision: Reads chip Version number: 0011 - Rev D0 0100 - Rev D1 0101 - Rev E0 0110 - Rev E1 0111 - Rev E2
0:	Reads zero.
ID[15:0]:	Chip ID Number, reads 9213.

Copyright 2007 Cirrus Logic

SysCfg

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV				RSVD											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD							SBOOT	LCSn7	LCSn6	LASDO	LEEDA	LEECLK	RSVD	LCSn2	LCSn1

Address:

0x8093_009C - Read/Write, Software locked

Default:

0x0000_0000

Definition:

System Configuration Register. Provides various system configuration options.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

REV: Revision: Reads chip Version number:
0000 - Rev A
0001 - Rev B
0010 - Rev C
0011 - Rev D0
0100 - Rev D1
0101 - Rev E0

SBOOT: Serial Boot Flag.
1 - hardware detected Serial Boot selection,
0 - hardware detected Normal Boot. This bit is read-only.

LCSn7, LCSn6: Latched version of **CSn7** and **CSn6** respectively. These are used to define the external bus width for the boot code.

LASDO: Latched version of **ASDO** pin. Used to select synchronous versus asynchronous boot device.

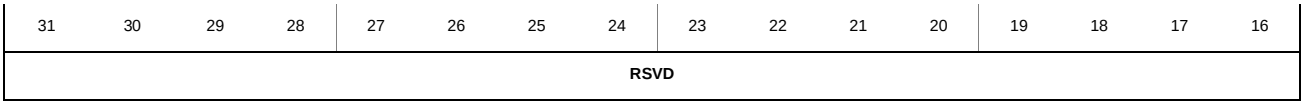
LEEDA: Latched version of **EEDAT** pin.

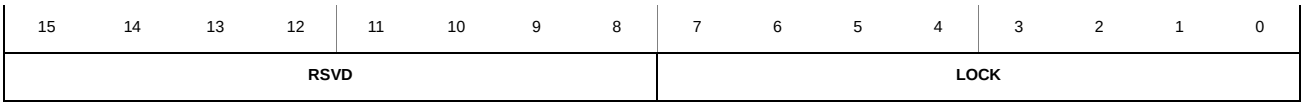
LEECLK: Define Internal or external boot:
1 - Internal
0 - External

Copyright 2007 Cirrus Logic

LCSn1, LCSn2: Define Watchdog startup action:
00 - Watchdog disabled, Reset duration disabled
01 - Watchdog disabled, Reset duration active
10 - Watchdog active, Reset duration disabled
11 - Watchdog active, Reset duration active

SysSWLock





Address:

0x8093_00C0 - Read/Write

Default:

0x0000_0000

Definition:

Syscon Software Lock Register. Provides software control port for all Syscon locked registers. Writing the LOCK field to 0xAA opens the lock. Reading the register will return 0x0000_0001 when the lock is open, and all zeros when the lock is closed (locked).

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

LOCK: Lock code value. This field must be written to a value of 0xAA to open the software lock. Reads 0x01 when the lock is open, 0x00 when the lock is closed.

Copyright 2007 Cirrus Logic

Copyright 2007 Cirrus Logic

Chapter 6
Vectored Interrupt Controller 6

6.1 Introduction

The EP93xx processors contain two cascaded Vectored Interrupt Controllers (VIC). A Vectored Interrupt has improved latency compared with a simple interrupt controller, since it provides direct information about where the interrupt's service routine is located and eliminates levels of software arbitration.

Each individual Vectored Interrupt Controller can handle up to 32 interrupts, but there are more than 32 interrupts in this design. Therefore two VICs are connected in a daisy-chain, which allows the system to handle up to 64 interrupt sources.

There are up to 16 vectored interrupts and 16 non-vectored interrupts available on each VIC. Vectored interrupts can only generate an IRQ interrupt. Non-vectored interrupts can generate either an IRQ interrupt or a FIQ interrupts. Vectored Interrupt Requests (IRQ) provide an address for an Interrupt Service Routine (ISR). Reading from the vector interrupt address register, VICxVectAddr, provides the address of the ISR, and indicates to the interrupt priority hardware that the interrupt is being serviced. Writing to the VICxVectAddr register indicates to the interrupt priority hardware that the interrupt has been serviced, allowing lower priority interrupts to go active.

Registers in the VIC use a bit position for each different interrupt source. The bit position is fixed, but the handling of each interrupt is configurable by the VIC. Software can generate software interrupts by controlling each request line.

The VIC provides a software interface to the interrupt system. Two levels of interrupts are available:

- Fast Interrupt Request (FIQ) for fast, low latency interrupt handling
- Interrupt Request (IRQ) for more general interrupts

All interrupt inputs to the VIC are presented as active-high level sensitive signals. Any conditioning needed to achieve this is performed by the block generating the interrupt request. In the case of external interrupts, the GPIO block takes care of the conditioning.

Note: Some GPIO signals are not configurable but are used as inputs by other functional blocks. **EGPIO[2:1]** are routed to the DMA controller to allow for external DMA requests.

Note: An interrupt vector may be overwritten when two interrupts occur simultaneously. If a VIC2 interrupt is immediately followed by a VIC1 interrupt, the VIC1 address will incorrectly be the default handler address for 2 HCLK cycles. To work around this problem, first check for pending non-vectored VIC1 interrupts in the interrupt routine. If there are none then return from interrupt. The interrupt will immediately re-occur with the correct vector address.

Copyright 2007 Cirrus Logic

6

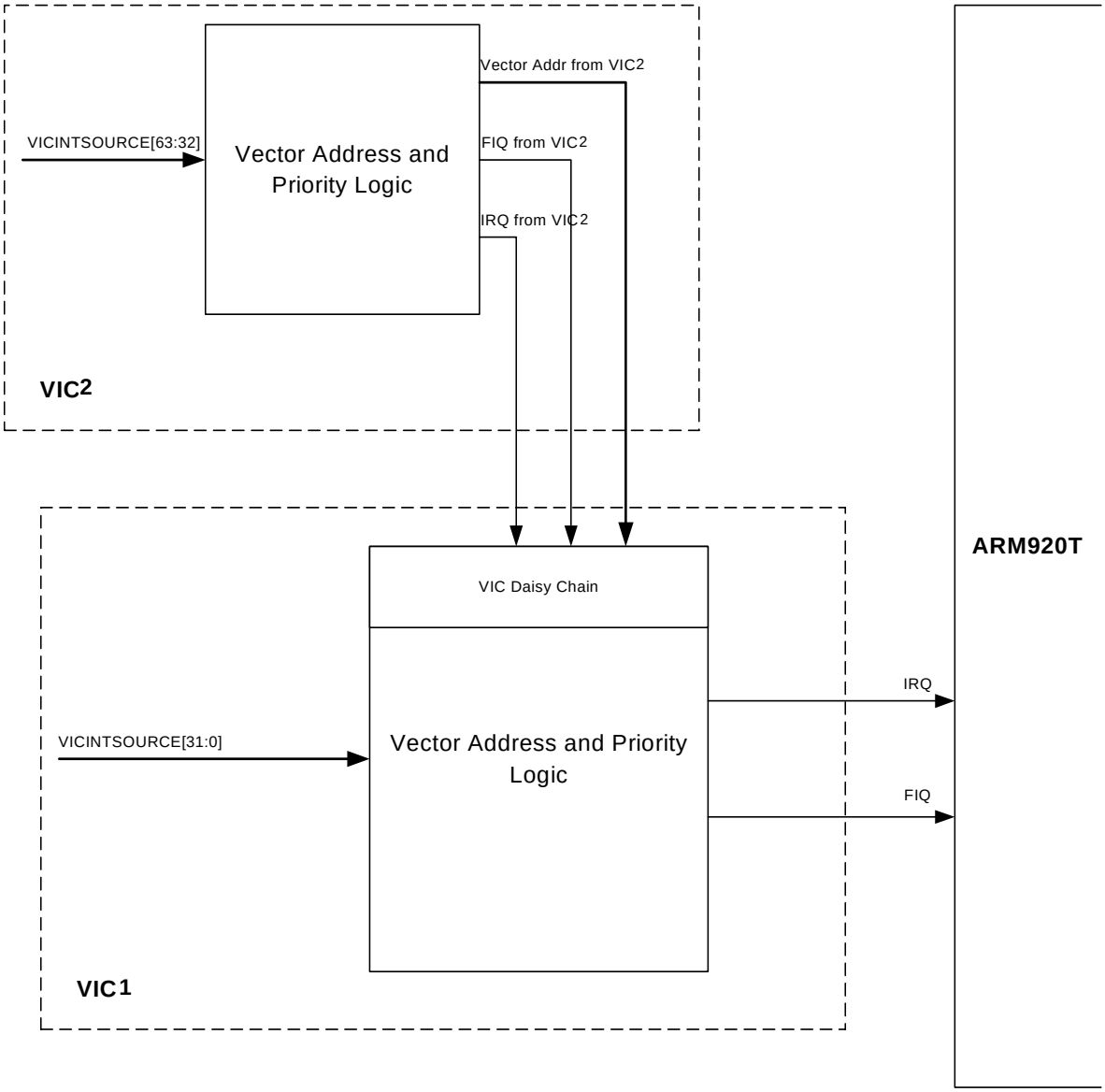


Figure 6-1. Vectored Interrupt Controller Block Diagram

6.1.1 Interrupt Priority

A FIQ interrupt has the highest priority (because the ARM9 core will always treat FIQ as higher priority), followed by vectored interrupt 0 to vectored interrupt 15. Non-vectored IRQ interrupts have the lowest priority. Any of the non-vectored Interrupts can be either FIQ or IRQ (the interrupt type is determined by programming the appropriate register, 'VICxIntSelect' on page 6-11).

Copyright 2007 Cirrus Logic

Any 16 of the 32 interrupts (per VIC) can be designated as 'vectored' by programming the Vector address registers, 'VICxVectAddr0' on page 6-15 and the Vector Control registers, 'VICxVectCntl0,' on page 6-17.

An interrupt is designated as either IRQ or FIQ by programming the VICxIntSelect register. The IRQ and FIQ request logic has an asynchronous path. This allows interrupts to be asserted when the clock is disabled.

Software can generate a specific interrupt by writing a '1' to the associated bit in the VICxSoftInt register.

6.1.2 Interrupt Configuration

Table 6-1 shows Interrupt Configuration.

Table 6-1. Interrupt Configuration

VIC Interrupt Source	Name	Description
0	-	Unused
1	-	Unused
2	COMMRX	ARM Communication Rx for Debug
3	COMMTX	ARM Communication Tx for Debug
4	TC1UI	TC1 under flow interrupt (Timer Counter 1)
5	TC2UI	TC2 under flow interrupt (Timer Counter 2)
6	AACINTR	Advanced Audio Codec interrupt
7	DMAM2P0	DMA Memory to Peripheral Interrupt 0
8	DMAM2P1	DMA Memory to Peripheral Interrupt 1
9	DMAM2P2	DMA Memory to Peripheral Interrupt 2
10	DMAM2P3	DMA Memory to Peripheral Interrupt 3
11	DMAM2P4	DMA Memory to Peripheral Interrupt 4
12	DMAM2P5	DMA Memory to Peripheral Interrupt 5
13	DMAM2P6	DMA Memory to Peripheral Interrupt 6
14	DMAM2P7	DMA Memory to Peripheral Interrupt 7
15	DMAM2P8	DMA Memory to Peripheral Interrupt 8
16	DMAM2P9	DMA Memory to Peripheral Interrupt 9
17	DMAM2M0	DMA Memory to Memory Interrupt 0
18	DMAM2M1	DMA Memory to Memory Interrupt 1
19	-	Reserved
20	-	Reserved
21	-	Reserved
22	-	Reserved
23	UART1RXINTR1	UART 1 Receive Interrupt
24	UART1TXINTR1	UART 1 Transmit Interrupt
25	UART2RXINTR2	UART 2 Receive Interrupt
26	UART2TXINTR2	UART 2 Transmit Interrupt
27	UART3RXINTR3	UART 3 Receive Interrupt
28	UART3TXINTR3	UART 3 Transmit Interrupt
29	INT_KEY	Keyboard Matrix Interrupt
30	INT_TOUCH	Touch Screen Controller Interrupt
31	-	Reserved

Copyright 2007 Cirrus Logic

6

Table 6-1. Interrupt Configuration

VIC Interrupt Source	Name	Description
32	INT_EXT[0]	External Interrupt 0
33	INT_EXT[1]	External Interrupt 1
34	INT_EXT[2]	External Interrupt 2
35	TINTR	64 Hz Tick Interrupt
36	WEINT	Watchdog Expired Interrupt
37	INT_RTC	RTC Interrupt
38	INT_IrDA	IrDA Interrupt
39	INT_MAC	Ethernet MAC Interrupt
40	-	Reserved
41	INT_PROG	Raster Programmable Interrupt
42	CLK1HZ	1 Hz Clock Interrupt
43	V_SYNC	Video Sync Interrupt
44	INT_VIDEO_FIFO	Raster Video FIFO Interrupt
45	INT_SSP1RX	SSP Receive Interrupt
46	INT_SSP1TX	SSP Transmit Interrupt
47	-	Reserved
48	-	Reserved
49	-	Reserved
50	-	Reserved
51	TC3UI	TC3 under flow interrupt (Timer Counter 3)
52	INT_UART1	UART 1 Interrupt
53	SSPINTR	Synchronous Serial Port Interrupt
54	INT_UART2	UART 2 Interrupt
55	INT_UART3	UART 3 Interrupt
56	USHINTR	USB Host Interrupt
57	INT_PME	Ethernet MAC PME Interrupt
58	INT_DSP	ARM Core Interrupt
59	GPIOINTR	GPIO Combined interrupt
60	I2SINTR	I2S Block Combined interrupt
61	-	Unused
62	-	Unused
63	-	Unused

6.1.3 Interrupt Details

Details of the interrupts described in Table 6-1 are:

COMMRX — ARM Communication Channel Receive. When high, COMMRX indicates that the communications channel receive buffer contains data waiting to be read by the ARM Core. Refer to the *ARM Technical Reference Manual*.

COMMTX — ARM Communication Channel Transmit. When high COMMTX indicates that the communications channel transmit buffer is empty. Refer to the *ARM Technical Reference Manual*.

Copyright 2007 Cirrus Logic

6

TC1UI	Timer Counter 1 Under Flow Interrupt. When Timer Counter 1 has underflowed (reached zero), this interrupt becomes active on the next falling edge of the timer's clock. The interrupt is cleared by writing any value to the "Timer1Clear," register. See Chapter 18, "Timers".
TC2UI	Timer Counter 2 Under Flow Interrupt. When Timer Counter 2 has underflowed (reached zero), this interrupt becomes active on the next falling edge of the timer's clock. The interrupt is cleared by writing any value to the "Timer2Clear," register. See Chapter 18, "Timers".
AACINTR	Advanced Audio CODEC Interrupt. See Chapter 22, "AC'97 Controller".
DMAM2P0	Internal Memory-to-Peripheral and Peripheral-to-Memory Channel 0 Interrupt. See Chapter 10, "DMA Controller".
DMAM2P1	Internal Memory-to-Peripheral and Peripheral-to-Memory Channel 1 Interrupt. See Chapter 10, "DMA Controller".
DMAM2P2	Internal Memory-to-Peripheral and Peripheral-to-Memory Channel 2 Interrupt. See Chapter 10, "DMA Controller".
DMAM2P3	Internal Memory-to-Peripheral and Peripheral-to-Memory Channel 3 Interrupt. See Chapter 10, "DMA Controller".
DMAM2P4	Internal Memory-to-Peripheral and Peripheral-to-Memory Channel 4 Interrupt. See Chapter 10, "DMA Controller".
DMAM2P5	Internal Memory-to-Peripheral and Peripheral-to-Memory Channel 5 Interrupt. See Chapter 10, "DMA Controller".
DMAM2P6	Internal Memory-to-peripheral and Peripheral-to-memory Channel 6 Interrupt. See Chapter 10, "DMA Controller".
DMAM2P7	Internal Memory-to-Peripheral and Peripheral-to-Memory Channel 7 Interrupt. See Chapter 10, "DMA Controller".
DMAM2P8	Internal Memory-to-Peripheral and Peripheral-to-Memory Channel 8 Interrupt. See Chapter 10, "DMA Controller".
DMAM2P9	Internal Memory-to-Peripheral and Peripheral-to-Memory Channel 9 Interrupt. See Chapter 10, "DMA Controller".
DMAM2M0	Memory-to-Memory (incorporating external M2P/P2M) Channel 0 Interrupt. See Chapter 10, "DMA Controller".
DMAM2M1	Memory-to-Memory (incorporating external M2P/P2M) Channel 1 Interrupt. See Chapter 10, "DMA Controller".
UART1RXINTR1	UART 1 Receive Interrupt. See Chapter 14, "UART1 With HDLC and Modem Control Signals"

Copyright 2007 Cirrus Logic

UART1TXINTR1	UART 1 Transmit Interrupt. See Chapter 14, "UART1 With HDLC and Modem Control Signals".
UART1RXINTR2	UART 2 Receive Interrupt. See Chapter 15, "UART2"".
UART1TXINTR2	UART 2 Transmit Interrupt. See Chapter 15, "UART2"".
UART1RXINTR3	UART 3 Receive Interrupt. See Chapter 16, "UART3 With HDLC Encoder".
UART1TXINTR3	UART 3 Transmit Interrupt. See Chapter 16, "UART3 With HDLC Encoder".
INT_KEY	Key Matrix Interrupt. See Chapter 26, "Keypad Interface".
INT_TOUCH	Touch Screen Controller Interrupt. This is the general interrupt from the TSC. See Chapter 25, "Analog Touch Screen Interface".
INT_EXT[0]	External Interrupt 0.
INT_EXT[1]	External Interrupt 1.
INT_EXT[2]	External Interrupt 2.
TINTR	64Hz TICK Interrupt. This interrupt becomes active on every rising edge of the internal 64Hz clock. The 64Hz clock is derived from a 15-stage ripple counter that divides the 32.768kHz oscillator input down to 1Hz for the real time clock. This interrupt is cleared by writing any value to the "RTCSts" register. See Chapter 20, "Real Time Clock With Software Trim"
WEINT	Watchdog Expired Interrupt. This interrupt will become active on a rising edge of the periodic 64Hz tick interrupt clock if the TICK interrupt (TINT) is still active. That is, if a tick interrupt has not been serviced for a complete tick period. Both WEINT and TINT interrupts are cleared by writing any value to the "RTCSts" register, see Chapter 20, "Real Time Clock With Software Trim". Failure to service this interrupt does not cause a system reset and the action taken on receipt of this interrupt is system dependent.
INT_RTC	Real Time Clock interrupt. See Chapter 20, "Real Time Clock With Software Trim".
INT_IrDA	IrDA Interrupt. See Chapter 17, "IrDA".
INT_MAC	Ethernet MAC Interrupt. See Chapter 9, "1/10/100 Mbps Ethernet LAN Controller".
INT_PROG	Programmable Interrupt. See Chapter 7, "Raster Engine With Analog/LCD Integrated Timing and Interface".

Copyright 2007 Cirrus Logic

CLK1HZ	1 Hz clock interrupt. See Chapter 20, "Real Time Clock With Software Trim".
V_SYNC	Vertical or Composite Sync/Frame Pulse Interrupt. See Chapter 7, "Raster Engine With Analog/LCD Integrated Timing and Interface".
INT_VIDEO_FIFO	Video FIFO Interrupt. See Chapter 7, "Raster Engine With Analog/LCD Integrated Timing and Interface"
INT_SSP1RX	SSP Receive Interrupt. See Chapter 23 "Synchronous Serial Port".
INT_SSP1TX	SSP Transmit Interrupt. See Chapter 23 "Synchronous Serial Port".
TC3UI	Timer Counter 3 Underflow Interrupt. This interrupt becomes active on the next falling edge of the timer counter 3 clock after the timer counter has under flowed (reached zero). The interrupt is cleared by writing any value to the "Timer3Clear" register. See Chapter 18, "Timers".
INT_UART1	UART 1 General Interrupt. This interrupt is active if any UART1 interrupt is active. Interrupt service routines will need to read the relevant status bits within UART1 to determine the source of the interrupt. All these sources are individually maskable within UART1. See Chapter 15, "UART1".
SSPINTR	Synchronous Serial Port (SSP) Interrupt. See Chapter 23 "Synchronous Serial Port".
INT_UART2	UART 2 General Interrupt. This interrupt is active if any UART2 interrupt is active. Interrupt service routines will need to read the relevant status bits within UART2 to determine the source of the interrupt. All these sources are individually maskable within UART2. See Chapter 15, "UART2".
INT_UART3	UART 3 General Interrupt. This interrupt is active if any UART3 interrupt is active. Interrupt service routines will need to read the relevant status bits within UART3 to determine the source of the interrupt. All these sources are individually maskable within UART3. See Chapter 16, "UART3 With HDLC Encoder".
USHINTR	USB Host Interrupt. See Chapter 11, "USB Host Controller".
INT_PME	PME interrupt. See Chapter 23 "Synchronous Serial Port".

Copyright 2007 Cirrus Logic

6

INT_DSP	ARM Core interrupt.
GPIOINTR	Combined Interrupt from Any Bit in Ports A or B. See Chapter 28, "GPIO Interface"
I2SINTR	Combined Interrupt of All Sources from the I^{2}S Controller. See Chapter 21, "I^{2}S Controller"

6.2 Registers

The 2 VIC blocks have an identical register definition. The offset from the respective base address is the same:

- VIC1 Base address: 0x800B_0000
- VIC2 Base Address: 0x800C_0000

Using the ARM MMU, it is possible to remap the VIC base address to 0xFFFF_F000, giving a lower interrupt latency. Table 6-2 indicates the address offset from the base address.

Table 6-2. VICx Register Summary

Address	Type	Width	Reset Value	Name	Description
VIC base + 0000	Read	32	0x0000_0000	VICxIRQStatus	IRQ status register
VIC base + 0004	Read	32	0x0000_0000	VICxFIQStatus	FIQ status register
VIC base + 0008	Read	32	-	VICxRawIntr	Raw interrupt status register
VIC base + 000C	Read /Write	32	0x0000_0000	VICxIntSelect	Interrupt select register
VIC base + 0010	Read /Write	32	0x0000_0000	VICxIntEnable	Interrupt enable register
VIC base + 0014	Write	32	-	VICxIntEnClear	Interrupt enable clear register
VIC base + 0018	Read /Write	32	0x0000_0000	VICxSoftInt	Software interrupt register
VIC base + 001C	Read /Write	32	-	VICxSoftIntClear	Software interrupt clear register
VIC base + 0020	Read /Write	1	0x0	VICxProtection	Protection enable register
VIC base + 0030	Read /Write	32	0x0000_0000	VICxVectAddr	Vector address register
VIC base + 0034	Read /Write	32	0x0000_0000	VICxDefVectAddr	Default vector address register
VIC base + 0100	Read /Write	32	0x0000_0000	VICxVectAddr0	Vector address 0 register
VIC base + 0104	Read /Write	32	0x0000_0000	VICxVectAddr1,	Vector address 1 register
VIC base + 0108	Read /Write	32	0x0000_0000	VICxVectAddr2,	Vector address 2 register
VIC base + 010C	Read /Write	32	0x0000_0000	VICxVectAddr3,	Vector address 3 register
VIC base + 0110	Read /Write	32	0x0000_0000	VICxVectAddr4,	Vector address 4 register
VIC base + 0114	Read /Write	32	0x0000_0000	VICxVectAddr5,	Vector address 5 register
VIC base + 0118	Read /Write	32	0x0000_0000	VICxVectAddr6	Vector address 6 register
VIC base + 011C	Read /Write	32	0x0000_0000	VICxVectAddr7,	Vector address 7 register
VIC base + 0120	Read /Write	32	0x0000_0000	VICxVectAddr8,	Vector address 8 register
VIC base + 0124	Read /Write	32	0x0000_0000	VICxVectAdd9,	Vector address 9 register
VIC base + 0128	Read /Write	32	0x0000_0000	VICxVectAddr10,	Vector address 10 register
VIC base + 012C	Read /Write	32	0x0000_0000	VICxVectAddr11,	Vector address 11 register
VIC base + 0130	Read /Write	32	0x0000_0000	VICxVectAdd12,	Vector address 12 register
VIC base + 0134	Read /Write	32	0x0000_0000	VICxVectAddr13,	Vector address 13 register
VIC base + 0138	Read /Write	32	0x0000_0000	VICxVectAddr14,	Vector address 14 register
VIC base + 013C	Read /Write	32	0x0000_0000	VICxVectAddr15	Vector address 15 register
VIC base + 0200	Read /Write	6	0x00	VICxVectCntl0,	Vector control 0 register
VIC base + 0204	Read /Write	6	0x00	VICxVectCntl1,	Vector control 1 register

Copyright 2007 Cirrus Logic

Table 6-2. VICx Register Summary

Address	Type	Width	Reset Value	Name	Description
VIC base + 0208	Read /Write	6	0x00	VICxVectCntl2,	Vector control 2 register
VIC base + 020C	Read /Write	6	0x00	VICxVectCntl3,	Vector control3 register
VIC base + 0210	Read /Write	6	0x00	VICxVectCntl4,	Vector control 4 register
VIC base + 0214	Read /Write	6	0x00	VICxVectCntl5,	Vector control 5 register
VIC base + 0218	Read /Write	6	0x00	VICxVectCntl6,	Vector control 6 register
VIC base + 021C	Read /Write	6	0x00	VICxVectCntl7,	Vector control 7 register
VIC base + 0220	Read /Write	6	0x00	VICxVectCntl8,	Vector control 8 register
VIC base + 0224	Read /Write	6	0x00	VICxVectCntl9,	Vector control 9 register
VIC base + 0228	Read /Write	6	0x00	VICxVectCntl10,	Vector control 10 register
VIC base + 022C	Read /Write	6	0x00	VICxVectCntl11,	Vector control 11 register
VIC base + 0230	Read /Write	6	0x00	VICxVectCntl12,	Vector control 12 register
VIC base + 0234	Read /Write	6	0x00	VICxVectCntl13,	Vector control 13 register
VIC base + 0238	Read /Write	6	0x00	VICxVectCntl14,	Vector control 14 register
VIC base + 023C	Read /Write	6	0x00	VICxVectCntl15	Vector control 15 register
VIC base + 0FE0	Read	8	0x90	VICxPeriphID0	VIC Identification register bits 7:0 (see Note below)
VIC base + 0FE4	Read	8	0x11	VICxPeriphID1	VIC Identification register bits 15:8 (see Note below)
VIC base + 0FE8	Read	8	0x04	VICxPeriphID2	VIC Identification register bits 23:16 (see Note below)
VIC base + 0FEC	Read	8	0x00	VICxPeriphID3	VIC Identification register bits 31:24 (see Note below)

Note: The Reset Values of the VICxPeriphID[3:0] registers collectively show the identification number for the Vectored Interrupt Controller (VIC). The read-only Reset Values are hard-wired. Consequently, the VICxPeriphID[3:0] registers are not included in the following Register Descriptions.

Register Descriptions

VICxIRQStatus

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IRQStatus															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQStatus															

Address:

VIC1IRQStatus: 0x800B_0000 - Read Only
VIC2IRQStatus: 0x800C_0000 - Read Only

Copyright 2007 Cirrus Logic

6

Definition:

IRQ Status Register. The VICxIRQStatus register provides the status of interrupts after IRQ masking.
Interrupts 0 - 31 are in VIC1IRQStatus.
Interrupts 32 - 63 are in VIC2IRQStatus.

Bit Descriptions:

IRQStatus: Shows the status of the interrupts after masking by the VICxIntEnable and VICxIntSelect registers. A "1" indicates that the interrupt is active, and generates an interrupt to the ARM Core.

VICxFIQStatus

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIQStatus															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIQStatus															

Address:

VIC1FIQStatus: 0x800B_0004 - Read Only
VIC2FIQStatus: 0x800C_0004 - Read Only

Definition:

FIQ Status Register. The VICxFIQStatus register provides the status of the interrupts after FIQ masking.

Bit Descriptions:

FIQStatus: Shows the status of the interrupts after masking by the VICxIntEnable and VICxIntSelect registers. A "1" indicates that the interrupt is active, and generates an interrupt to the ARM Core.

VICxRawIntr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RawIntr															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RawIntr															

Address:

VIC1RawIntr: 0x800B_0008 - Read Only
VIC2RawIntr: 0x800C_0008 - Read Only

Copyright 2007 Cirrus Logic

Definition:

The VICxRawIntr register provides the status of the source interrupts (and software interrupts) to the interrupt controller.

Bit Descriptions:

RawIntr: Shows the status of the interrupts before masking by the enable registers. A "1" indicates that the corresponding interrupt request is active before masking.

VICxIntSelect

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IntSelect															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IntSelect															

Address:

VIC1IntSelect: 0x800B_000C - Read/Write
VIC2IntSelect: 0x800C_000C - Read/Write

Definition:

Interrupt Select Register. The VICxIntSelect register selects whether the corresponding interrupt source generates an FIQ or an IRQ interrupt.

Bit Descriptions:

IntSelect: Selects type of interrupt for interrupt request:
1 = FIQ interrupt
0 = IRQ interrupt.

VICxIntEnable

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IntEnable															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IntEnable															

Address:

VIC1IntEnable: 0x800B_0010 - Read/Write
VIC2IntEnable: 0x800C_0010 - Read/Write

Default: 0x0000_0000

Copyright 2007 Cirrus Logic

Definition:

Interrupt Enable Register. The VICxIntEnable register enables the interrupt requests by unmasking the interrupt sources. On reset, all interrupts are disabled (masked).

Bit Descriptions:

IntEnable: Enables the interrupt request lines:
1 - Interrupt enabled. Allows interrupt request to ARM Core.
0 - Interrupt disabled.

VICxIntEnClear

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IntEnable Clear															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IntEnable Clear															

Address:

VIC1IntEnClear: 0x800B_0014 - Write Only
VIC2IntEnClear: 0x800C_0014 - Write Only

Default: Don't Care

Definition:

Interrupt Enable Clear Register. The VICxIntEnClear register clears bits in the VICxIntEnable register.

Bit Descriptions:

IntEnable Clear: Clears bits in the VICxIntEnable register. Writing a bit to "1" clears the corresponding bit in the VICxIntEnable register. Any bits written to "0" have no effect.

VICxSoftInt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SoftInt															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SoftInt															

Address:

VIC1SoftInt: 0x800B_0018 - Read/Write
VIC2SoftInt: 0x800C_0018 - Read/Write

Copyright 2007 Cirrus Logic

Default: Don't Care

Definition:

Software Interrupt Register. The VICxSoftInt register is used to generate software interrupts.

Bit Descriptions:

SoftInt: Writing a bit to "1" generates a software interrupt for the corresponding source interrupt before interrupt masking. Writing a bit to "0" has no effect.

VICxSoftIntClear

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SoftIntClear															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SoftIntClear															

Address:

VIC1SoftIntClear: 0x800B_001C - Write Only
VIC2SoftIntClear: 0x800C_001C - Write Only

Default: Don't Care

Definition:

Software Interrupt Clear Register. The VICxSoftIntClear register clears bits in the VICxSoftInt register.

Bit Descriptions:

SoftIntClear: Clears bits in the VICxSoftInt register. Writing a bit to "1" clears the corresponding bit in the VICxSoftInt register. Writing a bit to "0" has no effect.

VICxProtection

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															Protection

Address:

VIC1Protection: 0x800B_0018 - Read/Write
VIC2Protection: 0x800C_0018 - Read/Write

Copyright 2007 Cirrus Logic

6

Definition:

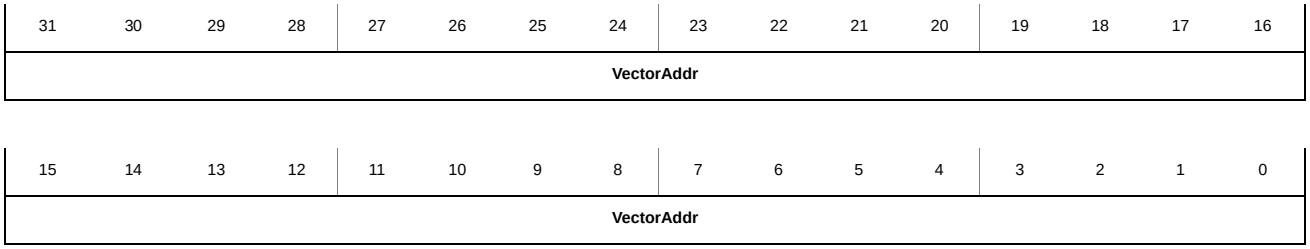
Protection Enable Register. The VICxProtection register enables or disables protected register access. If the bus master cannot generate accurate protection information, leave this register in its reset state to allow User mode access.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

Protection: Enables or disables protected register access. When enabled, only Privileged mode accesses (reads and writes) can access the interrupt controller registers. When disabled, both User mode and Privileged mode can access the registers. This bit is cleared to '0' on reset, and can only be accessed in Privileged mode.

VICxVectAddr



Address:

VIC1VectAddr: 0x800B_0030 - Read/Write
VIC2VectAddr: 0x800C_0030 - Read/Write

Definition:

Vector Address Register. The VICxVectAddr register contains the Interrupt Service Routine (ISR) address of the currently active interrupt.

Note: Reading from this register provides the address of the ISR, and indicates to the priority hardware that the interrupt is being serviced. Writing to this register indicates to the priority hardware that the interrupt has been serviced. The register should be used as follows:

- The ISR reads the VICxVectAddr register when an IRQ interrupt is generated
- At the end of the ISR, the VICxVectAddr register is written with any value in order to update the priority hardware.

Reading or writing to the register at other times can cause incorrect operation.

Note: If you are using the VIC and a program/debugger ever reads address VIC_BASE + 0x30, a value must be written to VIC_BASE + 0x30. If not, only higher priority interrupts are enabled and there are no higher priority interrupts. Therefore, no more interrupts will occur. If you use the VIC in Vectored Interrupt mode, this is not an issue.

Copyright 2007 Cirrus Logic

If you are not using the priority level in the VIC, write the VICxVectAddr register with any value (in order to disable the interrupt priority) at the beginning of your program.

It is not always clear when the ARM debuggers read the VICxVectAddr register, so it is recommended that if you are using a debugger, do not read the VIC registers via a memory window. If you must read the VIC registers, read only the VIC registers that are needed.

Bit Descriptions:

VectorAddr: Contains the address of the currently active ISR. Any writes to this register clear the interrupt.

VICxDefVectAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DefaultVectorAddr															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DefaultVectorAddr															

Address:

VIC1DefVectAddr: 0x800B_0034 - Read/Write
VIC2DefVectAddr: 0x800C_0034 - Read/Write

Definition:

Default Vector Address Register. The VICxDefVectAddr register contains the default ISR address.

Bit Descriptions:

DefaultVectorAddr: Contains the address of the default ISR handler.

VICxVectAddr0

VICxVectAddr1,

VICxVectAddr2,

VICxVectAddr3,

VICxVectAddr4,

VICxVectAddr5,

VICxVectAddr6

Copyright 2007 Cirrus Logic

6

VICxVectAddr7,

VICxVectAddr8,

VICxVectAdd9,

VICxVectAddr10,

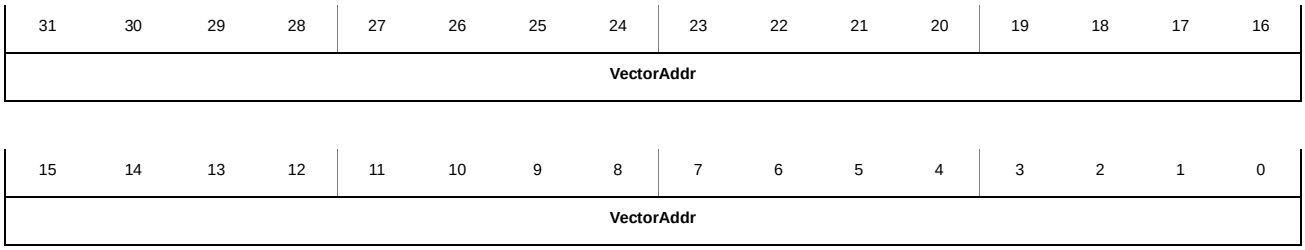
VICxVectAddr11,

VICxVectAdd12,

VICxVectAddr13,

VICxVectAddr14,

VICxVectAddr15



Address:

VIC1VectAddr0: 0x800B_0100 - Read/Write
VIC1VectAddr1: 0x800B_0104 - Read/Write
VIC1VectAddr2: 0x800B_0108 - Read/Write
VIC1VectAddr3: 0x800B_010C - Read/Write
VIC1VectAddr4: 0x800B_0110 - Read/Write
VIC1VectAddr5: 0x800B_0114 - Read/Write
VIC1VectAddr6: 0x800B_0118 - Read/Write
VIC1VectAddr7: 0x800B_011C - Read/Write
VIC1VectAddr8: 0x800B_0120 - Read/Write
VIC1VectAddr9: 0x800B_0124 - Read/Write
VIC1VectAddr10: 0x800B_0128 - Read/Write
VIC1VectAddr11: 0x800B_012C - Read/Write
VIC1VectAddr12: 0x800B_0130 - Read/Write
VIC1VectAddr13: 0x800B_0134 - Read/Write
VIC1VectAddr14: 0x800B_0138 - Read/Write
VIC1VectAddr15: 0x800B_013C - Read/Write
VIC2VectAddr0: 0x800C_0100 - Read/Write
VIC2VectAddr1: 0x800C_0104 - Read/Write
VIC2VectAddr2: 0x800C_0108 - Read/Write
VIC2VectAddr3: 0x800C_010C - Read/Write
VIC2VectAddr4: 0x800C_0110 - Read/Write
VIC2VectAddr5: 0x800C_0114 - Read/Write

Copyright 2007 Cirrus Logic

VIC2VectAddr6: 0x800C_0118 - Read/Write
VIC2VectAddr7: 0x800C_011C - Read/Write
VIC2VectAddr8: 0x800C_0120 - Read/Write
VIC2VectAddr9: 0x800C_0124 - Read/Write
VIC2VectAddr10: 0x800C_0128 - Read/Write
VIC2VectAddr11: 0x800C_012C - Read/Write
VIC2VectAddr12: 0x800C_0130 - Read/Write
VIC2VectAddr13: 0x800C_0134 - Read/Write
VIC2VectAddr14: 0x800C_0138 - Read/Write
VIC2VectAddr15: 0x800C_013C - Read/Write

Definition:

Vector Address Registers. The 32 VICxVectAdd0 through VICxVectAdd15 registers contain the ISR vector addresses, that is, the addresses of the ISRs for the particular 16 interrupts that are vectored.

Bit Descriptions:

VectorAddr: Contains ISR vector address.

VICxVectCntl0,

VICxVectCntl1,

VICxVectCntl2,

VICxVectCntl3,

VICxVectCntl4,

VICxVectCntl5,

VICxVectCntl6,

VICxVectCntl7,

VICxVectCntl8,

VICxVectCntl9,

VICxVectCntl10,

VICxVectCntl11,

VICxVectCntl12,

VICxVectCntl13,

VICxVectCntl14,

Copyright 2007 Cirrus Logic

VICxVectCntl15

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD										E	IntSource				

Address:

VIC1VectCntl0: 0x800B_0200 - Read/Write
VIC1VectCntl1: 0x800B_0204 - Read/Write
VIC1VectCntl2: 0x800B_0208 - Read/Write
VIC1VectCntl3: 0x800B_020C - Read/Write
VIC1VectCntl4: 0x800B_0210 - Read/Write
VIC1VectCntl5: 0x800B_0214 - Read/Write
VIC1VectCntl6: 0x800B_0218 - Read/Write
VIC1VectCntl7: 0x800B_021C - Read/Write
VIC1VectCntl8: 0x800B_0220 - Read/Write
VIC1VectCntl9: 0x800B_0224 - Read/Write
VIC1VectCntl10: 0x800B_0228 - Read/Write
VIC1VectCntl11: 0x800B_022C - Read/Write
VIC1VectCntl12: 0x800B_0230 - Read/Write
VIC1VectCntl13: 0x800B_0234 - Read/Write
VIC1VectCntl14: 0x800B_0238 - Read/Write
VIC1VectCntl15: 0x800B_023C - Read/Write
VIC2VectCntl0: 0x800C_0200 - Read/Write
VIC2VectCntl1: 0x800C_0204 - Read/Write
VIC2VectCntl2: 0x800C_0208 - Read/Write
VIC2VectCntl3: 0x800C_020C - Read/Write
VIC2VectCntl4: 0x800C_0210 - Read/Write
VIC2VectCntl5: 0x800C_0214 - Read/Write
VIC2VectCntl6: 0x800C_0218 - Read/Write
VIC2VectCntl7: 0x800C_021C - Read/Write
VIC2VectCntl8: 0x800C_0220 - Read/Write
VIC2VectCntl9: 0x800C_0224 - Read/Write
VIC2VectCntl10: 0x800C_0228 - Read/Write
VIC2VectCntl11: 0x800C_022C - Read/Write
VIC2VectCntl12: 0x800C_0230 - Read/Write
VIC2VectCntl13: 0x800C_0234 - Read/Write
VIC2VectCntl14: 0x800C_0238 - Read/Write
VIC2VectCntl15: 0x800C_023C - Read/Write

Definition:

Vector Control Registers. The 32 VICxVectCntl0 through VICxVectCnt15 registers select the interrupt source for the vectored interrupt.

Copyright 2007 Cirrus Logic

Note: Vectored interrupts are only generated if the interrupt is enabled. The specific interrupt is enabled in the VICxIntEnable register, and the interrupt is set to generate an IRQ interrupt in the VICxIntSelect register. This prevents multiple interrupts being generated from a single request if the controller is incorrectly programmed.

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
E:	Enables vector interrupt. This bit is cleared to '0' on reset.
IntSource:	Selects interrupt source by number. You can select any of the 32 interrupt sources.

Copyright 2007 Cirrus Logic

Copyright 2007 Cirrus Logic

Chapter 7

Raster Engine With Analog/LCD Integrated Timing and Interface

7

7.1 Introduction

Note: This chapter applies only to the EP9307, EP9312, and EP9315 processors. For additional information regarding the use of t he EP93XX Raster Engine, see the application note, AN269, "Using the EP93xx's Raster Engine" at:

http://www.cirrus.com/en/pubs/appNote/AN269REV1.pdf.

The Raster engine is capable of providing data and timing signals for a variety of displays. The engine has fully programmable video interface timings for progressive, dual scan, and interlaced displays. This programmable interface also allows the raster engine to generate a First Line Marker on the VSYNC line required by many low cost passive LCD displays. Separate DAC interface signals are provided to allow analog RGB signal generation for analog LCD displays or CRTs. The circuitry is also designed to generate CCIR656 4:2:2 YCrCb digital video output signals for interfacing with an NTSC encoder.

The Raster engine has an 18-bit pixel output bus. The engine also includes support for an 8-bit parallel display interface for attaching to low-end display modules with integrated controller and frame buffer. All control register accesses are memory mapped as single word values and cannot be accessed as 8-bit or 16-bit memory values.

The Raster engine also provides hardware accelerated cursor support. The cursor size is programmable up to 64 pixels wide by 64 pixels in height, and it can be stored anywhere in memory as a 2 bpp bitmap image. The Raster Cursor accesses system memory to fetch the cursor image data that will be automatically blended with the video image.

The Raster Display AHB bus master can be attached directly to SDRAM Port 0 via a side-band bus or to any SDRAM port connected to the system AHB. If the raster engine is connected to the system AHB, the selection bits in the VideoAttribs register determine which of the 4 SDRAM chip selects are used for display buffer access. The choice of which bus to use should be based on video bandwidth requirements and should be selected before video services are activated. For systems with low to moderate video bandwidth, the Raster Display can be attached to SDRAM Port 0 via the side-band bus. This setup allows some parallelism in bus traffic, but suffers from slow AHB access to external memory. If the video bandwidth requirements are high, or there is an expectation of low competing traffic, then the Raster Display should be attached to the AHB and the Arbiter priority should be set to give the Raster Display highest priority. This attachment gets the best bandwidth available for the display, but other system performance will suffer.

Copyright 2007 Cirrus Logic

The Raster engine also supports several hardware blinking modes, and 8-bit addressed lookup tables for grayscale or expanding color depth. The Raster also includes a video stream signature generator for built in self-testing.

Examples for some of the possible output modes are shown in Table 7-1.

Table 7-1. Raster Engine Video Mode Output Examples

Display Type	Horizontal Resolution x Vertical Resolution	Video Clock Freq. (MHz)	Frame Buffer Storage Format	Display Data Format	Pixels Per Shift Clock	Pixel Shift Clock Freq. (MHz)	Vertical Frame Rate (Hz)	Notes
VFD	128 x 32	2	4 bpp	Monochrome	8	0.25	400	
LCD	128 x 64	2	4 bpp	Monochrome	4	0.5	230	Parallel Command Word interface
LCD	256 x 128	2	4 bpp	Monochrome	4	0.5	60	-
"QVGA" TFT LCD	320 x 234	6.4	8 bpp	Analog	1	6.4	80	-
QVGA STN LCD	320 x 240	4	4-bit RGB	4-bit RGB	1	4	50	-
HVGA STN LCD	640 x 240	8	4-bit RGB	4-bit RGB	1	8	50	-
"VGA" DC Plasma	640 x 400	16	4 bpp	Monochrome	4	4	60	-
VGA EL	640 x 480	24	4 or 8 bpp	Grayscale	8	3	75	-
VGA STN LCD	640 x 480	24	8 or 16 bpp	18-bit RGB	1	24	75	-
VGATFT LCD	640 x 480	24	8, 16, or 24 bpp	18-bit RGB	1	24	75	-
VGA CRT	640 x 480	32	8, 16, or 24 bpp	Analog	1	NA	85	External DAC
SVGA TFT LCD	800 x 600	40	8, 16, or 24 bpp	18-bit RGB	1	40	80	-
SVGA CRT	800 x 600	50	8, 16, or 24 bpp	Analog	1	NA	85	External DAC
XGA CRT	1024 x 768	75	8, 16, or 24 bpp	Analog	1	NA	80	External DAC
SXGA TFT LCD	1280 x 1024	85	8, 16, or 24 bpp	18 or 24 RGB	1	85	60	24-bits

Copyright 2007 Cirrus Logic

Table 7-1. Raster Engine Video Mode Output Examples

Display Type	Horizontal Resolution x Vertical Resolution	Video Clock Freq. (MHz)	Frame Buffer Storage Format	Display Data Format	Pixels Per Shift Clock	Pixel Shift Clock Freq. (MHz)	Vertical Frame Rate (Hz)	Notes
SXGA CRT	1280 x 1024	110	8, 16, or 24bpp	Analog	1	NA	70	External DAC
HDTV-2 LCD	1280 x 720	50	8, 16, or 24 bpp	24-bit RGB	1	50	50	24-bits
HDTV-2 CRT	1280 x 720	66	8, 16, or 24 bpp	Analog	1	NA	60	External DAC

Since the frame buffer is stored in SDRAM memory, supporting displays with high frame rates at high resolutions will not be practical and sometimes not possible without using displays that have an integrated frame buffer.

7.2 Features

- Hardware pixel blinking
- Dual 256-color Look-up-tables (LUT)
- Grayscale/Color Generation for Monochrome/Passive Low Color Displays
- Flexible frame buffer architecture
- Supports video information in DIB (Device Independent Bitmap) format
- Hardware support for left and right panning of the displayed information
- Supports screen sizes up to 1280 x 1024 pixels, with a pixel depth of 4 bpp, 8 bpp, 16 bpp, 24 bpp packed, or 32 bpp (24 bpp unpacked)

Note: Using the Maximum Resolution causes system performance to slow.

- Pulse Width Modulated output that can be used to provide a DC voltage level for brightness control
- Hardware cursor support with bottom and right edge clipping performed by hardware
- 24-bit color depth, but only 18 bits is bond-out

7.3 Raster Engine Features Overview

7.3.1 Hardware Blinking

The raster engine pipeline contains hardware pixel blinking logic. This circuitry will blink pixels based on the Rate field in the BlinkRate register. For 4 bpp and 8 bpp modes, either multiple or single bit planes may be used to specify blinking pixels by look up in the LUT. This will allow the number of definable blinking pixels to range from all pixel combinations blinking

Copyright 2007 Cirrus Logic

to one pixel combination blinking. For 16 bpp and 24 bpp modes, the LUT blink circuitry is usually bypassed and the blink functions are logic transformations of the pixel data. In addition to logical AND/OR/XOR LUT address translations, the circuitry will support logical blink to background, blink dim, blink bright, and blink to reverse.

7.3.2 Color Look-Up Tables

The raster engine block contains dual color pixel LUTs (Look-Up-Tables). Each LUT will allow the engine to output 256 different pixel combinations of 24-bit pixels in lower color depth modes.

7.3.3 Grayscale/Color Generation for Monochrome/Passive Low Color Displays

The video pipeline includes circuitry that can be configured to provide grayscale or color generation for generating grayscales on monochrome displays or adding color depth on low color LCD displays, respectively. For monochrome displays, the circuitry supports up to 8 grayscale shades including on and off. For low color LCD displays, the circuitry supports up to 512 colors. The circuitry does this by rapidly turning on and off (dithering) pixels based on frame count, screen location, and pixel value. For grayscale displays, the pixel gray appearance is determined by 3 bits of the pixel data. For color depth expansion on LCD displays, the pixel color appearance is determined by 3 bits each from the red, green, and blue portions of the pixel data.

7.3.4 Frame Buffer Organization

The Raster Engine is designed to support video information as DIB (Device Independent Bitmap) format stored in a packed pixel architecture. However, the engine does not require that video information be stored in a packed line architecture. The circuitry allows a different memory organization between video scan out and graphic image memory. Therefore, memory gaps can exist between lines. This means that the graphics memory may be organized wider than the video frame. This type of feature could be used for left and right panning of the displayed information. The video frame buffer can be located in main memory, or in a dedicated video frame area. The beginning of video lines can be located on any word boundary. This architecture allows efficient use of memory regardless of the active video line length. Video screen start registers determine the upper left corner of the video screen. Video word addressing in screen memory is from left to right and then from top to bottom. Four-bit pixels packed within video words are organized in DIB format with the left most pixel in the

Copyright 2007 Cirrus Logic

most significant location on a per byte basis. Table 7-2 demonstrates pixel packing within words in a byte oriented Frame Buffer organization.

Table 7-2. Byte Oriented Frame Buffer Organization

As stored in memory	In pixel output order (progressive scan)

4 bits per pixel

32-bit Word							
Byte 3		Byte 2		Byte1		Byte 0	
bit 31	bit 24	bit 23	bit 16	bit 15	bit 8	bit 7	bit 0
Pixel 6	Pixel 7	Pixel 4	Pixel 5	Pixel 2	Pixel 3	Pixel 0	Pixel 1

Pixel 0 is first pixel out (upper left corner of screen) -->

Pixel 0	Pixel 1	Pixel 2	Pixel 3	Pixel 4	Pixel 5	Pixel 6	Pixel 7
bit 7	bit 0	bit 15	bit 8	bit 23	bit 16	bit 31	bit 24
Byte 0		Byte1		Byte 2		Byte 3	
32-bit Word							

8 bits per pixel

32-bit Word							
Byte 3		Byte 2		Byte1		Byte 0	
bit 31	bit 24	bit 23	bit 16	bit 15	bit 8	bit 7	bit 0
Pixel 3		Pixel 2		Pixel 1		Pixel 0	

Pixel 0		Pixel 1		Pixel 2		Pixel 3	
bit 7	bit 0	bit 15	bit 8	bit 23	bit 16	bit 31	bit 24
Byte 0		Byte1		Byte 2		Byte 3	
32-bit Word							

15 or 16 bits per pixel

32-bit Word							
Byte 3		Byte 2		Byte1		Byte 0	
bit 31	bit 24	bit 23	bit 16	bit 15	bit 8	bit 7	bit 0
Pixel 1				Pixel 0			

Pixel 0				Pixel 1			
bit 15	bit 8	bit 7	bit 0	bit 31	bit 24	bit 23	bit 16
Byte1		Byte 0		Byte 3		Byte 2	
32-bit Word							

24 bits per pixel packed

32-bit Word 0							
Byte 3		Byte 2		Byte 1		Byte 0	
bit 31	bit 24	bit 23	bit 16	bit 15	bit 8	bit 7	bit 0
Pixel 1 Blue		Pixel 0 Red		Pixel 0 Green		Pixel 0 Blue	

32-bit Word 1							
Byte 7		Byte 6		Byte 5		Byte 4	
bit 31	bit 24	bit 23	bit 16	bit 15	bit 8	bit 7	bit 0
Pixel 2 Green		Pixel 2 Blue		Pixel 1 Red		Pixel 1 Green	

32-bit Word 2							
Byte B		Byte A		Byte 9		Byte 8	
bit 31	bit 24	bit 23	bit 16	bit 15	bit 8	bit 7	bit 0

Pixel 0		Pixel 1		Pixel 2		Pixel 3	
Red		Red		Red		Red	
bit 23	bit 16	bit 15	bit 8	bit 7	bit 0	bit 31	bit 24
Byte 2		Byte 5		Byte 8		Byte B	
Word 0		Word 1		Word 2		Word 2	
Green		Green		Green		Green	
bit 15	bit 8	bit 7	bit 0	bit 31	bit 24	bit 23	bit 16
Byte 1		Byte 4		Byte 7		Byte A	
Word 0		Word 1		Word 1		Word 2	
Blue		Blue		Blue		Blue	
bit 7	bit 0	bit 31	bit 24	bit 23	bit 16	bit 15	bit 8
Byte 0		Byte 3		Byte 6		Byte 9	
Word 0		Word 0		Word 1		Word 2	

Copyright 2007 Cirrus Logic

Table 7-2. Byte Oriented Frame Buffer Organization (Continued)

As stored in memory				In pixel output order (progressive scan)
Pixel 3 Red	Pixel 3 Green	Pixel 3 Blue	Pixel 2 Red	

32 bits per pixel (24 bits per pixel unpacked)

32-bit Word 0							
Byte 3		Byte 2		Byte 1		Byte 0	
bit 31	bit 24	bit 23	bit 16	bit 15	bit 8	bit 7	bit 0
Unused		Pixel 0 Red		Pixel 0 Green		Pixel 0 Blue	

32-bit Word 1							
Byte 7		Byte 6		Byte 5		Byte 4	
bit 31	bit 24	bit 23	bit 16	bit 15	bit 8	bit 7	bit 0
Unused		Pixel 1 Red		Pixel 1 Green		Pixel 1 Blue	

32-bit Word 2							
Byte B		Byte A		Byte 9		Byte 8	
bit 31	bit 24	bit 23	bit 16	bit 15	bit 8	bit 7	bit 0
Unused		Pixel 2 Red		Pixel 2 Green		Pixel 2 Blue	

Pixel 0		Pixel 1		Pixel 2		Pixel 3	
Red		Red		Red		Red	
bit 23	bit 16	bit 23	bit 16	bit 23	bit 16	bit 23	bit 16
Byte 2		Byte 6		Byte B		Byte E	
Word 0		Word 1		Word 2		Word 3	
Green		Green		Green		Green	
bit 15	bit 8	bit 15	bit 8	bit 15	bit 8	bit 15	bit 8
Byte 1		Byte 5		Byte A		Byte D	
Word 0		Word 1		Word 2		Word 3	
Blue		Blue		Blue		Blue	
bit 7	bit 0	bit 7	bit 0	bit 7	bit 0	bit 7	bit 0
Byte 0		Byte 4		Byte 8		Byte C	
Word 0		Word 1		Word 2		Word 3	

Compressed images for remapping

1 bit per pixel

32-bit Word										
Byte 3			Byte 2			Byte1			Byte 0	
bit 31		bit 24	bit 23		bit 16	bit 15		bit 8	bit 7		bit 0
Pixel 24		Pixel 31	Pixel 16		Pixel 23	Pixel 8		Pixel 15	Pixel 0		Pixel 7

7.3.5 Frame Buffer Memory Size

Several screens may be available for video display depending on screen size, pixel depth, and amount of memory dedicated to video images. The screen size can be up to 1280 x 1024 pixels, the pixel depth can be 4 bpp, 8 bpp, 16 bpp, 24 bpp packed, or 32 bpp (24 bpp unpacked).

7.3.6 Pulse Width Modulated Brightness

The circuitry provides a pulse width modulated brightness control output, Bright, that can be used in conjunction with an external resistor and capacitor to provide a DC voltage level for

Copyright 2007 Cirrus Logic

brightness control. The Bright output signal can also be used for direct pulse width modulated CCFL brightness control that can be synchronized to the display frame rate.

7.3.7 Hardware Cursor

The Raster Engine provides hardware cursor support. The cursor size is programmable up to 64 pixels wide by 64 pixels in height. The cursor is stored anywhere in memory as a 2 bpp image. The 2 bpp image pixel information implies transparent, inverted, cursor color 1, or cursor color 2. The cursor hardware must be supplied this information:

- Image starting address
- Two cursor colors
- An X screen location and a Y screen location
- A cursor size

Using this information, the hardware overlays the cursor in the output video stream. Bottom and right edge clipping is performed by hardware. Some extra calculations and register setups are required for cursor support during dual scan display mode.

7.4 Functional Details

The Raster Engine's block diagram is shown in Figure 7-1. The video pipeline consists of several major sections; VILOSATI, video FIFO, pixel mux, blink logic, color LUT, RGB mux, output shift logic, grayscale circuitry, hardware cursor logic, YCrCb encoder, and video timing section. A video stream signature generator is also included for built in self testing.

Copyright 2007 Cirrus Logic

7

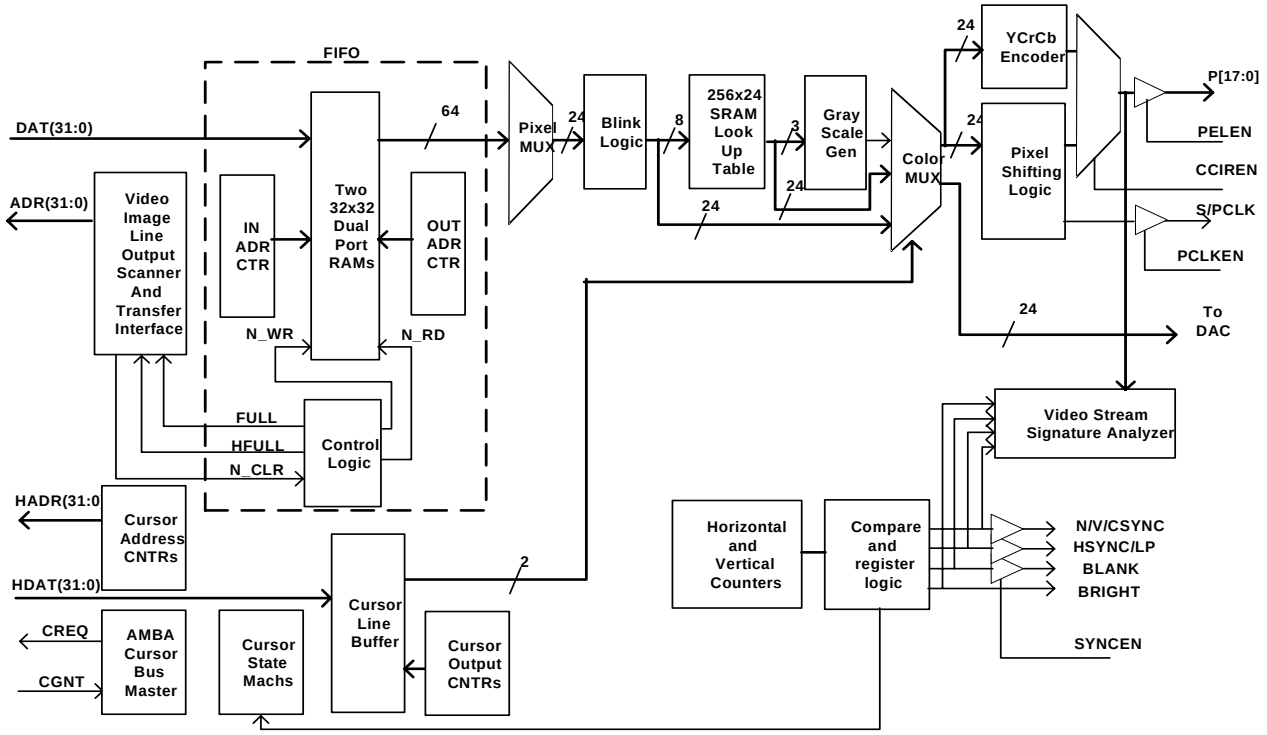


Figure 7-1. Raster Engine Block Diagram

7.4.1 VILOSATI (Video Image Line Output Scanner and Transfer Interface)

The Raster Engine's video image line output scanner and transfer interface connects to a either a dedicated DMA port on the SDRAM controller or to AHB access to the SDRAM controller and reads the video image from SDRAM to the video FIFO. VILOSATI keeps track of image location, width, and depth for both progressive and dual scanned images. It responds to controls from the FIFO for more video information. During single scan operation, when the FIFO level falls below a programmable fill level (FIFOLevel defaults to a value of 16 words), the FULL signal is inactive and VILOSATI attempts to initiate an unspecified length incrementing burst of at least 16 words. The VILOSATI will initiate incrementing unspecified length bursts until the FIFO is full. When the FIFO signals that it has emptied below the FIFOLevel again, the image reading process from the frame buffer continues.

Note: FIFOLevel values of greater than 16 words are not recommended due to the possibility of FIFO underflow.

For dual scan operation, the FIFO is split into two halves, where each halve operates with a separate FULL indicator. In dual scan mode, selected by writing DSCAN = '1' to the PixelMode register, the FULL and DS_FULL indicators trigger when either has room for a burst of 8 words (the LSB of FIFOLevel is ignored). For dual and single scan displays, information for the upper left corner of the display begins at the word address stored in the

Copyright 2007 Cirrus Logic

register, "VidScrnPage" on page 7-46. For a dual scan display, information from the upper left corner of the lower half of the display begins at the word address stored in the "VidScrnHPage" register. The "VidScrnPage" and "VidScrnHPage" registers are used to pre-load address counters at the beginning of the video frame.

The VILOSATI continues to service the video FIFO until it has transferred an entire screen image from memory. The size of the screen image is controlled by the values stored in the "ScrnLines" and "LineLength" registers. The "ScrnLines" register defines the total number of displayed (active) lines for the video frame. The "LineLength" register defines the number of words for each displayed (active) video line. A separate register, "VLineStep" on page 7-48, defines the word offset in memory between the beginning of each line and the next line. Setting the VLineStep value larger than the LineLength value provides the capability for image panning as shown in Figure 7-2.

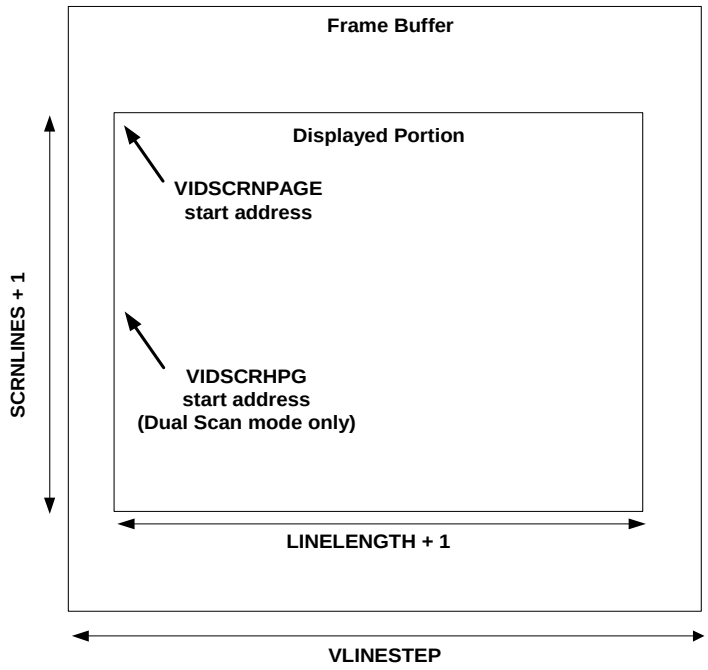


Figure 7-2. Video Buffer Diagram

7.4.2 Video FIFO

The video FIFO is used to buffer data transferred from the image memory to the Video output circuitry without stalling the video data stream. The FIFO consists of a dual port RAM with input and output index counters and control circuitry to operate it as a FIFO memory. The input data bus width to the FIFO is 32 bits. During half page mode, when the display requires scan out of the bottom and top half of the screen at the same time (dual scan), top half (or bottom half) data is stored in every other FIFO location.

When the screen is single scan (scanned out as a single progressive image), FIFO data is stored sequentially. The FIFO output data bus is 64 bits wide and can output even and odd

Copyright 2007 Cirrus Logic

words on both the upper and lower half of the bus. The FIFO has an underflow interrupt indicator that can be used to determine if the system is providing adequate bandwidth and low enough latency to support the selected display pixel depth, resolution, and refresh rate.

7.4.3 Video Pixel MUX

The pixel reconstruction circuitry uses multiplexers and pipe-line registers to 'unpack' the video pixels that are output from the video FIFO. The stored FIFO words are transferred 2 at a time across a 64-bit bus. The multiplexers select a single pixel to go on the 24-bit output bus based on the P value that is written to the "PixelMode" register. The multiplexers are controlled by a pixel counter that also increments based on the PixelMode.P value. The amount and frequency of data read from the FIFO is dependent on the number of bits per pixel. For example, in 8 bpp configuration (PixelMode.P = 0x2), the 64-bit FIFO output is changed for every eight pixels. In dual scan mode, selected by writing DSCAN = '1' to the "PixelMode" register, the upper 32 bits and lower 32 bits are read out in parallel and the upper-half screen and lower-half screen pixels are unpacked and loaded into the video stream sequentially.

7.4.4 Blink Function

The Raster Engine provides blinking pixel control circuitry. This circuitry provides a means to blink pixels at a rate specified by a programmable count of video frames. The number of video frames for a blink cycle is controlled by the "BlinkRate" register. There is only a single blink state bit, so all blinking pixels blink at the same programmed frequency. The most flexible way to blink pixels is to use a look-up-table (LUT). This is done by logically transforming the address into the look-up-table based on whether the pixel is a blink pixel, and whether it is currently in the blink state. For example, a red blinking pixel may be set up to normally address location 0x11 in the look-up-table. When not in the blink state, the color output from this location would be red. In the blink state, the address could be logically modified to 0x21. The color stored at the 0x21 location could be green or black or whatever other color that it is to be used in place of red in the blink state. To define a pixel as blink, some color information must be sacrificed. For every pixel color, there could be a blinking version. This would cut the possible number of system colors in half.

For LUT blinking, the address is modified by using a masked AND/OR/XOR function. The mask is defined in the "BlinkMask" register. Selection of whether the pixel data is ANDed, ORed, or XORed with the mask is set by writing to the M field in the "PixelMode" register.

The LUT blinking solution is only useful for 4 bpp and 8 bpp modes because the total number of colors is limited to 256. The extra bit width in 16 bpp and 24 bpp modes is not used. Therefore, for 16 bpp, and 24 bpp modes, the LUT blink circuitry is usually bypassed (based on the C field in the "PixelMode" register) and the blink function is performed by logical or mathematical operations on the pixel data. These operations can be programmed for Blink to Background, Blink Dimmer, Blink Brighter, or Blink to Offset by writing the appropriate value to the M field in the "PixelMode" register.

When Blink to Background mode is enabled, the blink circuitry replaces any blinking pixel with the "BkgrndOffset" register value. Setting this register to the background screen color in

Copyright 2007 Cirrus Logic

this mode will cause an object to appear and disappear. A drawback to this mode is that it may cause problems with correctly viewing overlapping objects. Blink Brighter and Blink Dimmer modes shift the pixel data values by one bit position. For Blink Brighter, the LSB is dropped, the MSBs are all shifted one bit lower, and the MSB is set to a "1". For Blink Dimmer, the LSB is dropped, the MSBs are all shifted one bit lower, and the MSB is set to a "0". Blink to Offset is simply adding the value in the BkgrndOffset register to blinking pixels. The shifting and offsetting can be programmed to be compatible with the selected pixel organization mode.

Defining blink pixels in 16 bpp and 24 bpp modes also may sacrifice the total number of colors available. A blinking pixel is defined by the "PattrnMask" and "PattrnMask" registers. By using the PattrnMask register, either multiple or single bit planes may be used to specify blinking pixels. This will allow the number of definable blinking pixels to range from all pixel combinations blinking to only one pixel that blinks. This approach allows the option of minimizing the number of lost colors by reducing the number of blinking colors. BlinkPattrn is then used to define the value of the PATTRNMASK bits in the "BlinkPattrn" register that should blink.

7.4.5 Color Look-Up-Tables

The Raster Engine contains two 256 x 24-bit RAMs that are used as color pixel LUTs to provide a selection of 256 colors from a palette of 16 million colors. One LUT is inserted in the video pipeline, while the other is accessible via the AHB. Changing the SWITCH bit in the "LUTSwCtrl" register toggles which LUT is in the pipe and which is accessible by the AHB. The LUTs are mapped to memory addresses and are accessible from the AHB one at a time. During active video display, the LUT switch command is synchronized to the beginning of the next vertical frame. When the video state machine is disabled the LUT switch occurs almost immediately. The status of actual switch occurrence can be monitored by reading the SSTAT bit in the "LUTSwCtrl" register. This bit can be polled, or the frame interrupt can be enabled and used to time the switching. Each LUT can be used for 4 bpp and 8 bpp modes and is usually bypassed for 16 bpp and 24 bpp modes. Control for whether or not the LUTs are used or bypassed altogether in the video pipeline is performed by writing to the appropriate value to C field (Color field) in the "PixelMode" register.

7.4.6 Color RGB Mux

The color RGB mux is necessary for selecting the appropriate pixel format and routing it to the appropriate video output stream. The Color RGB mux formats data for the pixel shift logic, a color DAC interface, or the YCrCb interface. The color RGB mux primary mode of operation is controlled by the "C" value (color value) in the "PixelMode" register. The primary mode of operation selects data from the grayscale generator, from the LUT, or from the video pipeline after the blink logic. When the hardware cursor is enabled by writing CLHEN = '1' in the "CursorDScanLHYLoc" register or CursorXYLoc.CEN = '1' in the "CursorXYLoc" register, CursorColor1/2 data values may be injected into the pipeline, or the primary incoming data may be inverted. The data formatting performed by the color RGB mux also depends on the "C" value (color value) in the "PixelMode" register. When in 16-bit 555 or 565 data modes, the pixel data is reformatted to fit into a 24-bit bus. This includes copying the MSBs for the

Copyright 2007 Cirrus Logic

data into the unused LSBs of the bus to support the full color intensity range. This part of the multiplexing circuitry actually occurs before the blink logic stage. Once selected and conditioned, output data is sent to the pixel shift logic and the YCrCb logic. The data is further conditioned with blanking in another pipeline operation before being sent to a color DAC.

7.4.7 Pixel Shift Logic

The pixel shifting logic on the output of the Video controller circuitry allows for reduced external data and clock rates by performing multiple pixel transfers in parallel. The output can be programmed to transfer a single pixel mapped to an 18-bit pixel output per clock (triple 6 RGB on 18 active data lines), 2 pixels per clock up to 9 bits wide each (18 pixel data lines active), 4 pixels per clock up to 4 bits wide each (16 pixel data lines active), or 8 pixels per clock up to 2 bits wide each (16 pixel data lines active). The interface can be programmed to output 2 2/3 - 3-bit pixels on the lower 8 bits of the bus per pixel clock. The interface can be programmed to operate in dual scan 2 2/3 pixel mode, placing 2 2/3 pixels from the upper and lower halves of the screen on the lower 8 bits of the bus and the next 8 bits of the bus per clock respectively. In dual scan mode, selected by writing DSCAN = '1' to the "PixelMode" register, every other pixel in the pipeline is from the other half of the display. Therefore, the dual scan output transfer modes that are supported are 1 upper/1 lower pixel, 2 upper/2 lower pixels, and 4 upper/4 lower pixels corresponding to the 2 pixels per clock, 4 pixels per clock and 8 pixels per clock modes.

Table 7-3 shows output pixel transfer modes based on the shift mode "S" value (shift value) and the color mode "C" value (color value) in the "PixelMode" register:

Copyright 2007 Cirrus Logic

Table 7-3. Output Pixel Transfer Modes

Shift Mode	Color Mode	Output Mode	P(23)	P(22)	P(21)	P(20)	P(19)	P(18)	P(17)	P(16)	P(15)	P(14)	P(13)	P(12)	P(11)	P(10)	P(9)	P(8)	P(7)	P(6)	P(5)	P(4)	P(3)	P(2)	P(1)	P(0)
0x0	0x0 0x4 0x8	single pixel per clock up to 24 bits wide	R(1)	R(0)	G(1)	G(0)	B(1)	B(0)	R(7)	R(6)	R(5)	R(4)	R(3)	R(2)	G(7)	G(6)	G(5)	G(4)	G(3)	G(2)	B(7)	B(6)	B(5)	B(4)	B(3)	B(2)
0x0	0x5	single 16-bit 565 pixel per clock	R(3)	R(2)	G(5)	G(4)	B(3)	B(2)	R(4)	R(3)	R(2)	R(1)	R(0)	R(4)	G(5)	G(4)	G(3)	G(2)	G(1)	G(0)	B(4)	B(3)	B(2)	B(1)	B(0)	B(4)
0x0	0x6	single 16-bit 555 pixel per clock	R(3)	R(2)	G(3)	G(2)	B(3)	P(2)	R(4)	R(3)	R(2)	R(1)	R(0)	R(4)	G(4)	G(3)	G(2)	G(1)	G(0)	G(4)	B(4)	B(3)	B(2)	B(1)	B(0)	B(4)
0x1	0x0 0x4 0x8	single 24-bit pixel mapped to 18 bits each clk	X	X	X	X	X	X	R(7)	R(6)	R(5)	R(4)	R(3)	R(2) *	G(7)	G(6)	G(5)	G(4)	G(3)	G(2) *	B(7)	B(6)	B(5)	B(4)	B(3)	B(2) *
0x1	0x5	single 16-bit 565 pixel mapped to 18 bits each clk	X	X	X	X	X	X	R(4)	R(3)	R(2)	R(1)	R(0)	R(4) *	G(5)	G(4)	G(3)	G(2)	G(1)	G(0) *	B(4)	B(3)	B(2)	B(1)	B(0)	B(4) *
0x1	0x6	single 16-bit 555 pixel mapped to 18 bits each clk	X	X	X	X	X	X	R(4)	R(3)	R(2)	R(1)	R(0)	R(4) *	G(4)	G(3)	G(2)	G(1)	G(0)	G(4) *	B(4)	B(3)	B(2)	B(1)	B(0)	B(4) *
0x2	0x0 0x8	progressive scan	P1(20) R1(4) *	P1(12) G1(4) *	P1(4) B1(4) *	P0(20) R0(4) *	P0(12) G0(4) *	P0(4) B0(4) *	P1(23) R1(7)	P1(22) G1(6)	P1(21) G1(5)	P1(15) G1(7)	P1(14) G1(6)	P1(13) G1(5)	P1(7) B1(7)	P1(6) B1(6)	P1(5) B1(5)	P0(23) R0(7)	P0(22) R0(6)	P0(21) R0(5)	P0(15) G0(7)	P0(14) G0(6)	P0(13) G0(5)	P0(7) B0(7)	P0(6) B0(6)	P0(5) B0(5)
		2 pixels per shift clock dual scan	Lower P(20) R(4) *	Lower P(12) G(4) *	Lower P(4) B(4) *	Upper P(20) R(4) *	Upper P(12) G(4) *	Upper P(4) B(4) *	Lower P(23) R(7)	Lower P(22) R(6)	Lower P(21) R(5)	Lower P(15) G(7)	Lower P(14) G(6)	Lower P(13) G(5)	Lower P(7) B(7)	Lower P(6) B(6)	Lower P(5) B(5)	Upper P(23) R(7)	Upper P(22) R(6)	Upper P(21) R(5)	Upper P(15) G(7)	Upper P(14) G(6)	Upper P(13) G(5)	Upper P(7) B(7)	Upper P(6) B(6)	Upper P(5) B(5)
0x3	0x0 0x8	progressive scan	P3(14) G3(6) *	P3(6) B3(6) *	P2(14) B2(6) *	P2(6) B2(6) *	P1(14) G1(6) *	P1(6) B1(6) *	P0(14) G0(6) *	P0(6) B0(6) *	P3(23) R3(7)	P3(22) R3(6) *	P3(15) G3(7)	P3(7) B3(7)	P2(23) R2(7)	P2(22) R2(6) *	P2(15) G2(7)	P2(7) B2(7)	P1(23) R1(7)	P1(22) R1(6) *	P1(15) G1(7)	P1(7) B1(7)	P0(23) R0(7)	P0(22) R0(6) *	P0(15) G0(7)	P0(7) B0(7)
		4 pixels per shift clock dual scan	Lower P1(14) G1(6) *	Lower P1(6) B1(6) *	Upper P1(14) G1(6) *	Upper P1(6) B1(6) *	Lower P0(14) G0(6) *	Lower P0(6) B0(6) *	Upper P0(14) G0(6) *	Upper P0(6) B0(6) *	Lower P1(23) R1(7)	Lower P1(22) R1(6) *	Lower P1(15) G1(7)	Lower P1(7) B1(7)	Upper P1(23) R1(7)	Upper P1(22) R1(6) *	Upper P1(15) G1(7)	Upper P1(7) B1(7)	Lower P0(23) R0(7)	Lower P0(22) R0(6) *	Lower P0(15) G0(7)	Lower P0(7) B0(7)	Upper P0(23) R0(7)	Upper P0(22) R0(6) *	Upper P0(15) G0(7)	Upper P0(7) B0(7)
0x4	0x0 0x8	progressive scan	P7(23) R7 *	P6(23) R6 *	P5(23) R5 *	P4(23) R4 *	P3(23) R3 *	P2(23) R2 *	P1(23) R1 *	P0(23) R0 *	P7(15) G7 *	P7(7) B7 *	P6(15) G6 *	P6(7) B6 *	P5(15) G5 *	P5(7) B5 *	P4(15) G4 *	P4(7) B4 *	P3(15) G3 *	P3(7) B3 *	P2(15) G2 *	P2(7) B2 *	P1(15) G1 *	P1(7) B1 *	P0(15) G0 *	P0(7) B0 *
		8 pixels per shift clock dual scan	Lower P3(23) R3 *	Upper P3(23) R3 *	Lower P2(23) R2 *	Upper P2(23) R2 *	Lower P1(23) R1 *	Upper P1(23) R1 *	Lower P0(23) R0 *	Upper P0(23) R0 *	Lower P3(15) G3 *	Lower P3(7) B3 *	Upper P3(15) G3 *	Upper P3(7) B3 *	Lower P2(15) G2 *	Lower P2(7) B2 *	Upper P2(15) G2 *	Upper P2(7) B2 *	Lower P1(15) G1 *	Lower P1(7) B1 *	Upper P1(15) G1 *	Upper P1(7) B1 *	Lower P0(15) G0 *	Lower P0(7) B0 *	Upper P0(15) G0 *	Upper P0(7) B0 *
0x5	0x0 0x8	2 2/3 pixels per clock	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	G2	B2	R1	G1	B1	R0	G0	B0
			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	B5	R4	G4	B4	R3	G3	B3	R2
			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	R7	G7	B7	R6	G6	B6	R5	G5

Copyright 2007 Cirrus Logic

Table 7-3. Output Pixel Transfer Modes (Continued)

Shift Mode	Color Mode	Output Mode	P(23)	P(22)	P(21)	P(20)	P(19)	P(18)	P(17)	P(16)	P(15)	P(14)	P(13)	P(12)	P(11)	P(10)	P(9)	P(8)	P(7)	P(6)	P(5)	P(4)	P(3)	P(2)	P(1)	P(0)
0x6	0x0 0x8	dual 2 2/3 pixels per clock	X	X	X	X	X	X	X	X	L G2	L B2	L R1	L G1	L B1	L R0	L G0	L B0	U G2	U B2	U R1	U G1	U B1	U R0	U G0	U B0
			X	X	X	X	X	X	X	X	L B5	L R4	L G4	L B4	L R3	L G3	L B3	L R2	U B5	U R4	U G4	U B4	U R3	U G3	U B3	U R2
			X	X	X	X	X	X	X	X	L R7	L G7	L B7	L R6	L G6	L B6	L R5	L G5	U R7	U G7	U B7	U R6	U G6	U B6	U R5	U G5
**	**	CCIREN subs	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	D(7)	D(6)	D(5)	D(4)	D(3)	D(2)	D(1)	D(0)
**	**	LCDEN subs	**	**	**	**	**	**	**	XECL	YSCL	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**
**	**	ACEN subs	**	**	**	**	**	**	AC	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**

*These bits are an ORed combination of the bit value shown and the next significant bit below (This rounds the color value to nearest color).

**These bits do not get a substitute and are defined to the values controlled by the pixel output mode in the upper part of the table.

Copyright 2007 Cirrus Logic

7.4.8 Grayscale/Color Generator for Monochrome/Passive Low Color Displays

The hardware raster engine has three built in matrix programmable grayscale generators. One generator is located on each of the red, green, and blue internal channels. These generators can be enabled to expand color depth or turn monochrome into grayscale through both spatial and temporal dithering. Dithering means that the circuit turns monochrome pixels on and off in a specific pattern and at a high toggle rate, and uses the integration perception of the human eye along with display persistence to achieve an average luminance between full on and full off. Using one of these generators allows creation of grayscale pixels on a monochrome display. Using all three of the generators with one on each red, green, and blue channel allows generation of additional colors on an 8 color LCD display.

Grayscale shading is accomplished on each channel by altering when and how often a given pixel is active. The setup for when and how often pixels of each value 0-7 are active is programmed into the grayscale look-up-table memory for each channel. The look-up-table for each RGB channel is indexed by 4 values: 3 bits from the input pixel value (0-7), and for each input pixel value either the 3 frame or 4 frame counter, the 3 line or 4 line vertical counter, and the 3 column or 4 column horizontal pixel counter. Pixel values 0-7 in each channel are programmed as to whether a count by 3 or count by 4 counter is used for frame, horizontal, and vertical.

The grayscale circuits are inserted into the video pipeline after the color LUT. The circuitry takes three bits from the output of the color LUT (one from each color) and uses them as the inputs for the grayscale LUT. These three bits are then processed by the grayscale circuitry to generate a new three bit output, based on the configuration of the grayscale LUT. The three bit output of the grayscale LUT is then fed through the pixel shifting logic and out to the Pixel Bus Pins. This provides 8 shades of gray per channel, including all off (black) and full on (white). Each circuit operates six separate 2-bit index counters; FRAME_CNT3, FRAME_CNT4, VERT_CNT3, VERT_CNT4, HORZ_CNT3, and HORZ_CNT4. Based on value of these counters, each grayscale look-up-table is programmed with values that define the on/off dithering operation for their respective three bits of the pixel value.

For example, in color mode 8 with shift mode 0:
Color LUT[23:21] -> Grayscale LUT[2] -> P[17:12] (All pins with Red color data)
Color LUT[15:13] -> Grayscale LUT[1] -> P[11:6] (All pins with Green color data)
Color LUT[7:5] -> Grayscale LUT[0] -> P[5:0] (All pins with Blue color data)

The following setup description refers to a single channel. First, the matrix size for each 3 bits of the pixel value (0 through 7) is defined. The matrix size is from 3 horizontal rows x 3 vertical columns x 3 frames to 4H x 4V x 4F or any combinations of 3 or 4. The grayscale look-up-table is then filled in for each pixel with this matrix information. Because the look-up-table is indexed by 4 values, it can be perceived as a multi-dimensional array. For each of the input pixel values 0-7, a 3H (Horizontal) x 3V (Vertical) x 3F (Frame) cube up to a 4H (Horizontal) x 4V (Vertical) x 4F (Frame) cube can be defined.

Setting the grayscale matrix values in a channel for full off and full on is very straight forward.

Copyright 2007 Cirrus Logic

Assuming that pixel input value 0 is off, setting raster engine base + grayscale LUTx offset + 0x00, 0x20, 0x40, and 0x60 to all '0's ensures that a 0 pixel never turns on. Assuming that pixel 7 is full on, setting raster engine base + grayscale LUTx offset + 0x1C, 0x3C, 0x5C, and 0x7C to all '1's ensures that the value is always on. Table 7-6 shows the format for programming.

7.4.8.1 HORZ_CNT3, HORZ_CNT4 Counters

These free running counters increment after displaying each pixel.

7.4.8.2 VERT_CNT3, VERT_CNT4 Counters

These free running counters increment at the end of every vertical line.

7.4.8.3 FRAME_CNT3, FRAME_CNT4 Counters

These free running counters increment at the end of each frame.

The GrySclLUT supports 3-bit pixel input. Each of the pixel combinations can define a unique combination of VERT, HORZ and FRAME counters, which provides for maximum flexibility in defining the rate at which a given pixel is manipulated as it is being displayed on the screen.

7.4.8.4 HORZ_CNTx (pixel) timing

This timing is controlled by the HORZ_CNTx counter and will indicate what pixel count values will cause a given pixel to be turned on. It is possible to have a pixel turned on for all HORZ counts, zero HORZ counts, or a defined pattern of HORZ counts. This counter is incremented by the pixel clock.

7.4.8.5 VERT_CNTx (line) timing

This timing is controlled by the VERT_CNTx counters and will indicate what line count values will cause a given pixel to be turned on. It is possible to have a pixel turn on for all VERT counts, zero VERT counts, or a defined pattern of VERT counts. This counter is incremented at the end of each line.

7.4.8.6 FRAME_CNTx timing

This timing is controlled by the FRAME_CNTx counters and will indicate when a full frame of video has been displayed. It is possible to have a pixel turn on for all FRAME counts, zero frame counts, or a combination of frame counts. This counter is incremented at the end of each frame.

The GrySclLUT combines all of the above information into a single table. In this way, it is possible to define a pixel to be on in all conditions (all HORZ, VERT, and FRAME counts), zero conditions, or any combination.

Copyright 2007 Cirrus Logic

7.4.8.7 Grayscale Look-Up Table (GrySclLUT)

Table 7-4. Grayscale Lookup Table (GrySclLUT)

Frame Ctr	Vert Ctr	Horz Ctr	VCNT (Lines)	11	11	11	11	10	10	10	10	01	01	01	01	00	00	00	00	GrySclLUT Address *4	
			HCNT (Pixels)	11	10	01	00	11	10	01	00	11	10	01	00	11	10	01	00	Frame	Pixel Value
D18	D17	D16	base+80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00	000
D18	D17	D16	base+84	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	00	001
D18	D17	D16	base+88	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	00	010
D18	D17	D16	base+8C	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	00	011
D18	D17	D16	base+90	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	00	100
D18	D17	D16	base+94	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	00	101
D18	D17	D16	base+98	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	00	110
D18	D17	D16	base+9C	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	00	111
X	X	X	base+A0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	01	000
X	X	X	base+A4	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	01	001
X	X	X	base+A8	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	01	010
X	X	X	base+AC	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	01	011
X	X	X	base+B0	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	01	100
X	X	X	base+B4	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	01	101
X	X	X	base+B8	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	01	110
X	X	X	base+BC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	01	111
X	X	X	base+C0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	000
X	X	X	base+C4	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	10	001
X	X	X	base+C8	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	10	010
X	X	X	base+CC	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	10	011
X	X	X	base+D0	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	10	100
X	X	X	base+D4	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	10	101
X	X	X	base+D8	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	10	110
X	X	X	base+DC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	10	111
X	X	X	base+E0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	000
X	X	X	base+E4	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	11	001
X	X	X	base+E8	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	11	010
X	X	X	base+EC	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	11	011
X	X	X	base+F0	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	11	100
X	X	X	base+F4	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	11	101
X	X	X	base+F8	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	11	110
X	X	X	base+FC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	11	111

Copyright 2007 Cirrus Logic

Where FRAME[1:0] = FRAME_CNT3 or FRAME_CNT4 as defined by FRAME at address Pixel_In,

VCNT[1:0] = VERT_CNT3 or VERT_CNT4 as defined by VERT at address Pixel_In, and

HCNT[1:0] = HORZ_CNT3 or HORZ_CNT4 as defined by HORZ at address Pixel_In.

This is the GrySclLUT table in an easily readable form. To understand how to use this table and to know how to fill the table with correct values requires a good understanding on how the table is used by the grayscale logic.

7.4.8.8 GrySclLUT Timing Diagram

Table 7-5 shows the timing diagram. The clock column represents a free running master clock for the display. This clock controls which pixel is being accessed as the image is being rasterized on the display.

Assume that the first 8 registers have the HCNT, VCNT and FRAME counter registers set up for 4 counts. The last column shows which register is used to retrieve the look up value and the bit position within that register that is used as the source to send to the COLORMUX for the given clock.

Clocks 4, 9, 14, and 19 represent all remaining pixels on the line. Clocks 24 and 29 represent all remaining pixels for the frame. These entries will keep this example table to a manageable size.

The FRAME count and PIXEL value are used to indicate which register contains the data. HCNT and VCNT are used to indicate which bit in the identified register is to be used for the given grayscale value.

Table 7-5. Grayscale Timing Diagram

Clock	HCNT	VCNT	FRAME	PIXEL	Register Address / Value
Clock	HCNT	VCNT	FRAME	PIXEL	Register Address / Value
0	0	0	0	5	(base + 94) / D0
1	1	0	0	5	(base + 94) / D1
2	2	0	0	5	(base + 94) / D2
3	3	0	0	5	(base + 94) / D3
4	“	“	“	“	
5	0	1	0	5	(base + 94) / D4
6	1	1	0	5	(base + 94) / D5
7	2	1	0	5	(base + 94) / D6
8	3	1	0	5	(base + 94) / D7
9	“	“	“	“	
10	0	2	0	5	(base + 94) / D8
11	1	2	0	5	(base + 94) / D9
12	2	2	0	5	(base + 94) / D10
13	3	2	0	5	(base + 94) / D11
14	“	“	“	“	
15	0	3	0	5	(base + 94) / D12
16	1	3	0	5	(base + 94) / D13
17	2	3	0	5	(base + 94) / D14

Copyright 2007 Cirrus Logic

Table 7-5. Grayscale Timing Diagram (Continued)

Clock	HCNT	VCNT	FRAME	PIXEL	Register Address / Value
18	3	3	0	5	(base + 94) / D15
19	"	"	"	"	
20	0	0	1	5	(base + b4) / D0
21	1	0	1	5	(base + b4) / D1
22	2	0	1	5	(base + b4) / D2
23	3	0	1	5	(base + b4) / D3
24	"	"	"	"	
25	0	0	2	5	(base + d4) / D0
26	1	0	2	5	(base + d4) / D1
27	2	0	2	5	(base + d4) / D2
28	3	0	2	5	(base + d4) / D3
29	"	"	"	"	
30	0	0	3	5	(base + f4) / D0
31	1	0	3	5	(base + f4) / D1
32	2	0	3	5	(base + f4) / D2
33	3	0	3	5	(base + f4) / D3

At clock 0, the HCNT, VCNT and FRAME counters are 0x0. The pixel to display is a 5, which translates to register base + 0x94, bit D0. At the next clock tick, the fastest running counter (HCNT) has incremented, but VCNT and FRAME remain the same. Given the same pixel value (5), bit position D1 is used as the value that is sent to the display.

Table 7-6. Programming Format

Frame	Vert	Horz	VCNT (lines)	11	11	11	11	10	10	10	10	01	01	01	01	00	00	00	00	GrySclLUT Address *4	
Ctr	Ctr	Ctr	HCNT (pixels)	11	10	01	00	11	10	01	00	11	10	01	00	11	10	01	00	Frame	Pixel
D18	D17	D16	register address	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0		Value
X	X	X	base + 0x00	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00	000
			base + 0x20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	01	000
			base + 0x40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	000
			base + 0x60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	000
X	X	X	base + 0x1C	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	00	111
			base + 0x3C	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	01	111
			base + 0x5C	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	10	111
			base + 0x7C	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	11	111

Copyright 2007 Cirrus Logic

The values in between full on and full off are more difficult to determine and depend on the display characteristics such as persistence, turn on time, and refresh rate. To achieve difference in shades of gray, it is typical to have more values below the half luminance average due to the higher sensitivity to luminance variations by the human eye at lower levels. Other problems that occur with choosing patterns and the operating matrix parameters are flickering (temporal distortion), walking patterns (spatial distortion), and spatial interference patterns.

Take, for example, a 50% duty cycle. We could define the matrix as a 4Hx4Vx4F as shown in Figure 7-3. However, we effectively halved the refresh rate of these pixels and, depending on the refresh rate of the display, are likely to see flickering for this shade.

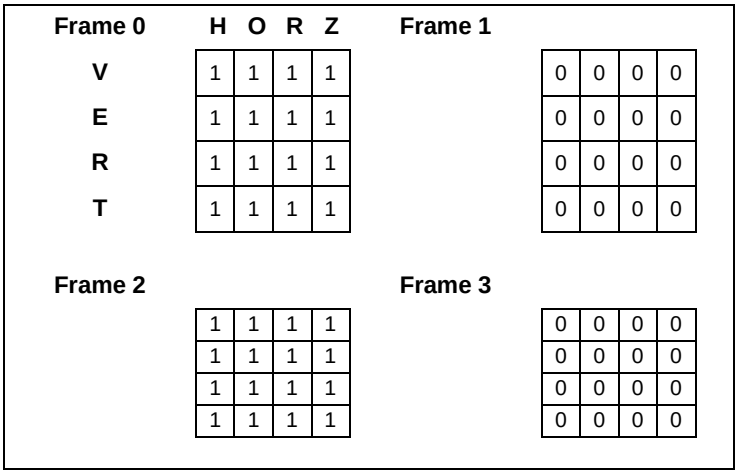


Figure 7-3. Graphics Matrix for 50% Duty Cycle

To avoid flickering, it is better to play a spatial trick and turn on every other pixel so that the eye integrates the on and off pixels between two consecutive frames. However, in the case given in Figure 7-3, a spatial interference can be caused if an image displayed in this grayscale consists of every other column activated. For this case, we would be right back to the flickering problem shown in Figure 7-4. This would be true if we switched to a checker board pattern and displayed a checker board image or almost any other pattern.

Copyright 2007 Cirrus Logic

7

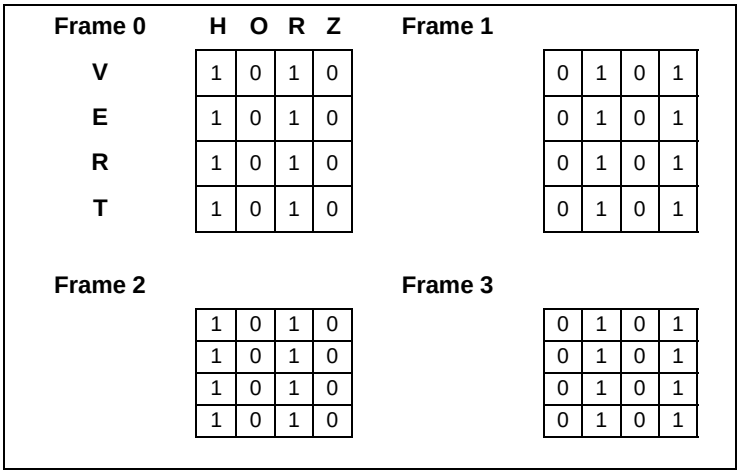


Figure 7-4. Sample Matrix Causing Flickering

To minimize these type of spatial interference patterns, it is better to mix up the pattern sequence similar to that shown in Figure 7-5. Note that the pattern mixes sets of two adjacent pixels with sets of every other pixel. Depending on the display and patterns displayed, this may create another type of apparent image distortion referred to as a walking pattern. One of the matrix indices may need to be changed to count by 3 to eliminate this combination of temporal and spatial distortion.

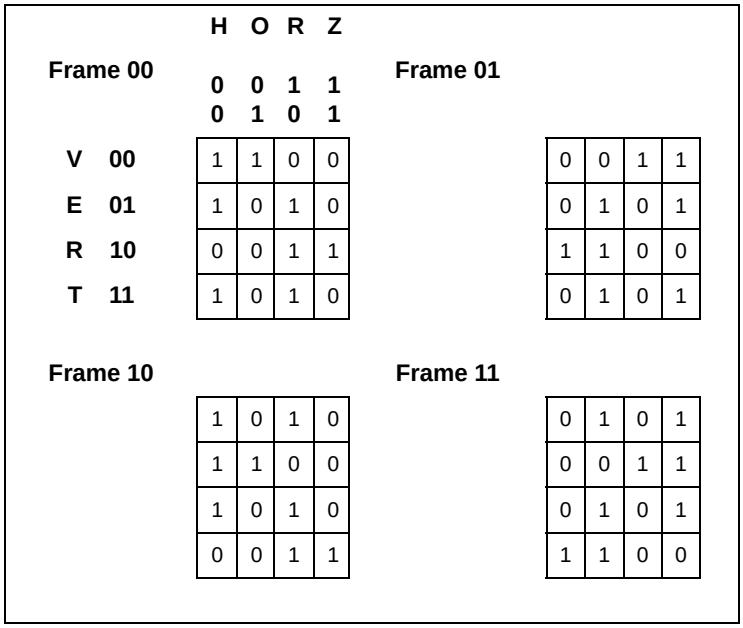


Figure 7-5. Sample Matrix That Avoids Flickering

Copyright 2007 Cirrus Logic

7

Assuming the 3 bit input pattern that represents this 50% duty cycle grayscale is 0x3 (or 011b), the values in Table 7-7 should be used to program this pattern into the grayscale look-up-table.

Table 7-7. Programming 50% Duty Cycle Into Lookup Table

Frame	Vert	Horz	VCNT (lines)	11	11	11	11	10	10	10	10	01	01	01	01	00	00	00	00	GrySclLUT Address *4	
Ctr	Ctr	Ctr	HCNT (pixels)	11	10	01	00	11	10	01	00	11	10	01	00	11	10	01	00	Frame	Pixel
D18	D17	D16	Register Address	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0		Value
1	1	1	base + 0x0C	0	1	0	1	1	1	0	0	0	1	0	1	0	0	1	1	00	011
			base + 0x2C	1	0	1	0	0	0	1	1	1	0	1	0	1	1	0	0	01	011
			base + 0x4C	1	1	0	0	0	1	0	1	0	0	1	1	0	1	0	1	10	011
			base + 0x6C	0	0	1	1	1	0	1	0	1	1	0	0	1	0	1	0	11	011

Since all patterns must be evaluated against their specific use, no more examples for half intensity will be offered. Instead, another example will be used to make a walking distortion more obvious.

Take the example of a one-third luminous intensity grayscale pattern. Assume a 3Hx3Vx3F matrix for this definition. Wanting the intensity to be evenly distributed and given the three frame interval, any cell in the matrix should only be active for one frame. The matrix could be filled in as in Figure 7-6.

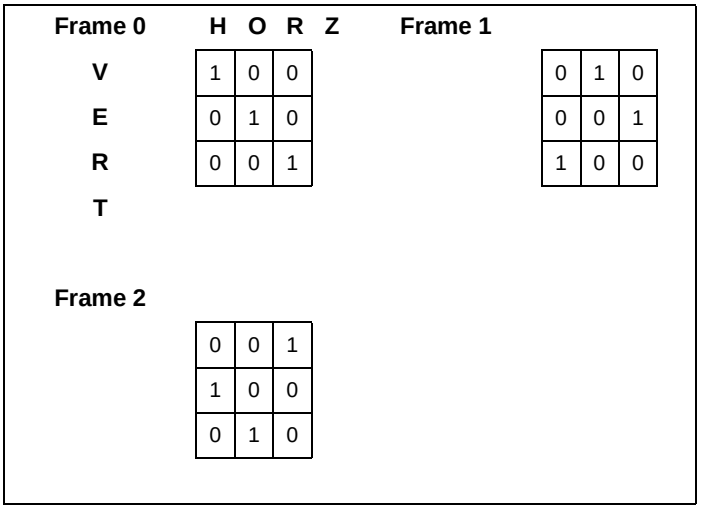


Figure 7-6. Programming for One-third Luminous Intensity

Copyright 2007 Cirrus Logic

Please note that as the frame number progresses, the bit pattern in each row moves to the right one pixel. This type of pattern shown in an area may cause diagonal lines to appear as though they are moving to the right. As previously stated, any image distortion greatly depends on the application. However, the pattern shown in Figure 7-7 will have less of a tendency to demonstrate a walking distortion.

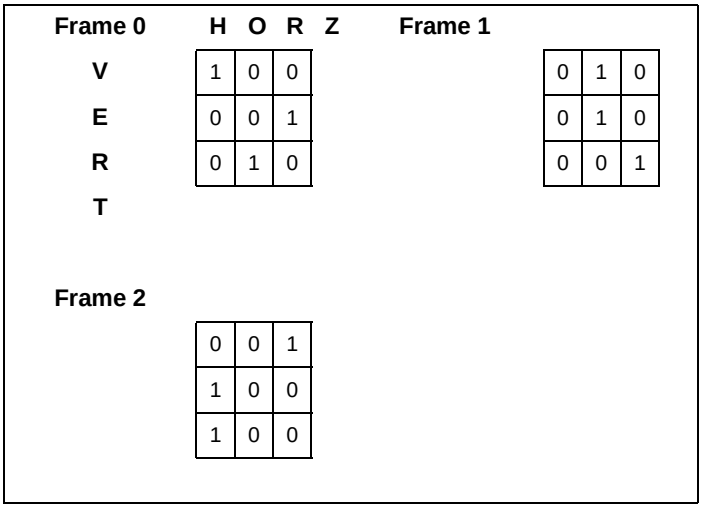


Figure 7-7. Creating Bit Patterns that Move to the Right

Assuming that the 3-bit input pattern that represents this 33% duty cycle grayscale is 0x2 (or 010b), the values in Table 7-8 are used to program this pattern into the grayscale look-up-table. In this mode, the X locations are ignored by the grayscale generation.

Table 7-8. Programming 33% Duty Cycle into the Lookup Table

Frame	Vert	Horz	VCNT (lines)	11	11	11	11	10	10	10	10	01	01	01	01	00	00	00	00	GrySclLUT Address *4	
Ctr	Ctr	Ctr	HCNT (pixels)	11	10	01	00	11	10	01	00	11	10	01	00	11	10	01	00	Frame	Pixel Value
D18	D17	D16	register address	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0		
0	0	0	base + 0x08	X	X	X	X	X	0	1	0	X	1	0	0	X	0	0	1	00	010
			base + 0x28	X	X	X	X	X	1	0	0	X	0	1	0	X	0	1	0	01	010
			base + 0x48	X	X	X	X	X	0	0	1	X	0	0	1	X	1	0	0	10	010
			base + 0x68	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	11	010

Finally, just for demonstration purposes, a matrix with mixed 3 and 4 count axes is shown in Figure 7-8.

Copyright 2007 Cirrus Logic

7

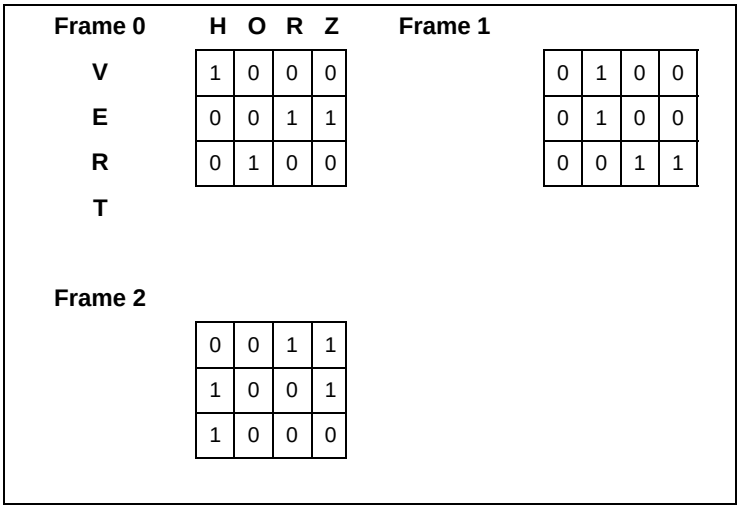


Figure 7-8. Three and Four Count Axis

Assuming that the 3-bit input pattern that represents this 33% duty cycle grayscale is 0x2 or 010b, the values in Table 7-9 are used to program this pattern into the grayscale look-up-table. In this mode, the X locations are ignored by the grayscale generation.

Table 7-9. Programming 33% Duty Cycle into the Lookup Table

Frame	Vert	Horz	VCNT (lines)	11	11	11	11	10	10	10	10	01	01	01	01	00	00	00	00	GrySclLUT Address *4	
Ctr	Ctr	Ctr	HCNT (pixels)	11	10	01	00	11	10	01	00	11	10	01	00	11	10	01	00	Frame	Pixel
D18	D17	D16	Register address	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0		Value
0	0	0	base + 0x08	X	X	X	X	0	0	1	0	1	1	0	0	0	0	0	1	00	010
			base + 0x28	X	X	X	X	1	1	0	0	0	0	1	0	0	0	1	0	01	010
			base + 0x48	X	X	X	X	0	0	0	1	1	0	0	1	1	1	0	0	10	010
			base + 0x68	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	11	010

7.4.9 Hardware Cursor

The raster engine provides support for a hardware cursor. The hardware cursor has a separate bus mastering interface that allows it's image to be stored anywhere in memory. Software provides a location start, reset, size, X and Y position, and two cursor colors. The hardware loads a line at a time from memory and multiplexes the video stream data based on the cursor values. The X and Y locations are compared to the horizontal and vertical counters and trigger the state machine to enable the cursor output overlay.

Copyright 2007 Cirrus Logic

"Start" is the beginning word location of the part of the cursor image to be displayed first. The image is 2-bits per pixel, and is stored linearly. The amount of storage space is dependent on the width and height of the cursor. The two bits per pixel stored define screen image (transparent), invert screen image, display color1, and display color2.

The 2-bits per pixel stored cursor image is displayed as:
00 - Transparent
01 - Invert video stream
10 - CursorColor1 during no blink or CursorBlinkColor1 during blink
11 - CursorColor2 during no blink or CursorBlinkColor2 during blink

Table 7-10. Cursor Memory Organization

	32-bit Word															
Byte	3				2				1				0			
Bit	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2	0
Pixel	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

"Reset" is the beginning word location of the part of the cursor which is displayed next after reaching the last line of the cursor. These locations are necessary for dual scan display of cursor information. If the cursor is totally in the upper half or lower half of the screen, the Start and Reset locations are the same. Otherwise, the cursor will start being overlaid on the video information at the start address, and when the dual scan height counter generates a carry, the cursor overlay will jump to the reset value. The cursor will then continue to be overlaid when the Y location is reached, and will jump to the start address value when the height counter for the upper half generates a carry.

Offsetting these values and changing the width of the cursor to be different from the cursor step value allows the right 48, 32, or 16 pixels of a larger cursor to be displayed only. Furthermore, offsetting the starting X location off of the left edge of the screen will allow pixel placement of the cursor off of the screen edge.

The size is specified as: width adjustable to 16, 32, 48, or 64 pixels, a height in lines up to 64 pixels (controls the top half of the screen only in dual scan mode), a step size for the number of words in a cursor line (up to 4), and a height of up to 64 lines on the bottom half of the screen (used in dual scan mode only).

The Y location value controls the starting vertical Y location of the cursor image. The value is compared to the vertical line counter and should be set by software to be between the active start and active stop vertical line values. The cursor hardware will clip the cursor at the bottom of the screen. To prevent cursor distortion, the new Y location value will not be used until the next frame.

Copyright 2007 Cirrus Logic

The X location value controls the starting horizontal X location of the cursor image. The value is compared to the horizontal pixel counter and should be set by software to be between the active start and active stop horizontal pixel values. The cursor hardware will clip the cursor at the right edge of the screen. This value is also used to control the starting location for the cursor image on the upper half of the screen during dual scan mode. To prevent cursor distortion, the new X location value will not be used until the next frame.

During dual scan display mode, selected by writing DSCAN = '1' to the "PixelMode" register, the lower half Y value controls the starting vertical Y location on the lower half of the screen for the cursor image. The value is compared to the vertical line counter and should be set by software to be between the active start and active stop vertical line values. The cursor hardware will clip the cursor at the bottom of the screen. To prevent cursor distortion, the new location value will not be used until the next frame.

The hardware cursor circuitry has a separate blinking function. The rate is a 50% duty cycle programmable number of vertical frame intervals. When a blink frame is active, the color RGB mux switches in 24-bit "CursorBlinkColor1," or "CursorBlinkColor2" values for "CursorColor1," or "CursorColor2," respectively.

7.4.9.1 Registers Used for Cursor

The registers used for configuring the Hardware Cursor are: "CursorAdrStart" , "CursorAdrReset" , "CursorSize" , "CursorColor1," "CursorColor2," , "CursorXYLoc" and CursorDScanLHYLoc. The following subsections describe the function of each of these registers.

7.4.9.1.1 CursorAdrStart Register

This register contains the memory starting address for the cursor image.

7.4.9.1.2 CursorAdrReset Register

This register contains the address for the part of the cursor that is displayed next after reaching the last line of the cursor. This register is needed to support DUAL scan displays. For non-dual scan displays, this address is the same as that in the CursorAdrStart register.

7.4.9.1.3 CursorSize Register

This register selects four parameters that will impact the cursor size: CSTEP, CLINS, CWID, and DLNS.

CSTEP

Two bits select the cursor step size:

0 0	Step by 1 word or 16 pixels
0 1	Step by 2 words or 32 pixels
1 0	Step by 3 words or 48 pixels
1 1	Step by 4 words or 64 pixels

Copyright 2007 Cirrus Logic

CLINS

Six bits select the height of the cursor image. The height is measured in lines and should be set to a value of one less then the desired number of lines.

CWID

Two bits select the cursor width:

0 0	Width is 1 word or 16 pixels
0	1Width is 2 words or 32 pixels
1 0	Width is 3 words or 48 pixels
1 1	Width is 4 words or 64 pixels

DLNS

Six bits are used in DUAL SCAN mode, where DUAL SCAN mode is selected by writing DSCAN = '1' to the "PixelMode" register.

7.4.9.1.4 CursorColor1 Register

This register is set to the cursor color value that is used when the pixel color value is a 0x2 (10 binary).

7.4.9.1.5 CursorColor2 Register

This register is set to the cursor color value that is used when the pixel color value is a 0x3 (11 binary).

7.4.9.1.6 CursorXYLoc Register

This register provides the place in the X and Y position of the image where the cursor should be inserted. The X position is represented by the XLOC bits and the Y position is represented by the YLOC bits in the "CursorXYLoc" register. The XLOC bits and YLOC bits are compared with the respective counter (YLOC is the line counter, XLOC is the pixel counter). These values must fall between the active start and stop parameters for the display.

This register also contains the enable bit, CEN, for the hardware cursor. Writing a '1' to this bit enables the hardware cursor.

Note: Very rarely, a vertical line appears when the hardware cursor becomes enable or disabled. This line is a few pixels wide and only lasts for one frame. It is hard to catch. In order to prevent this problem: 1. Do not enable/disable the cursor when changing the cursor bitmaps, and 2. When disabling the cursor, change the CursorXYLOc register to point to a blank cursor image.

7.4.9.1.7 CursorDScanLHYLoc Register

See "CursorDScanLHYLoc" register.

Copyright 2007 Cirrus Logic

7.4.10 Video Timing

The video timing circuitry consists of a horizontal down counter and a vertical down counter. Signal timing for a specific video format is generated by programmable values that are compared to the count values.

An AC signal is generated to support either bias voltage switching for LCDs or a field indicator for interlaced video. The An AC signal, if ACEN = '1' in the "VideoAttribs" register, is output on the P[17] pin. The toggle rate of the AC signal is selected by writing to the "ACRate" register.

LCD shifting signals, XECL and YSCL, are generated to support simple LCDs. These signals, if LCDEN = '1' in the "VideoAttribs" register, are output on pixel data pins P[16] and P[15], respectively. XECL is generated every 64 pixel clocks. YSCL is the inversion of HSYNCn.

The Raster Engine provides an end of frame interrupt, when enabled, to the interrupt controller. This interrupt defines when the last information has been sent to the display for the current frame. It indicates the start of an interval when changes can be made to the LUT or source for the displayed image without affecting the display. It must be configured as an edge triggered interrupt. Changes such as a new cursor location or a new screen image location automatically change at this time, under hardware control. The interval for making LUT changes, etc. without affecting the displayed image depends on the display's technology. The time duration is equal to the vertical blanking interval (VLinesTotal duration - VACTIVE duration).

In addition, the programmable VCLR and HCLR fields in the "SigClrStr" register are used as a secondary interrupt during normal operation, where the interrupt can be programmed to trigger at any vertical and horizontal counter combination.

The frequency of the clock used for video timing and the entire video pipeline must meet the requirements of the display type. The video clock frequency is selected by writing to the VidClkDiv register (see Chapter 5). The video circuitry is targeted to run up to 132MHz. This corresponds to a 1280 pixels by 1024 pixels display size, and non-interlaced video at a 80Hz frame refresh.

Note: Total Bus/SDRAM bandwidth is shared between the Raster Engine and other device controllers. The pixel depth, display size, and display refresh rate can be limited by the Bus/SDRAM bandwidth that is available to the Raster Engine.

The programmed values for the video timing section of the raster engine are shown in Figure 7-9, "Progressive/Dual Scan Video Signals" and Figure 7-10, "Interlaced Video Signals". Independent horizontal and vertical down counters are used as a reference for all other signals. The synchronization, blanking, and active video control signalling is generated by comparing programmed values to the counters.

Copyright 2007 Cirrus Logic

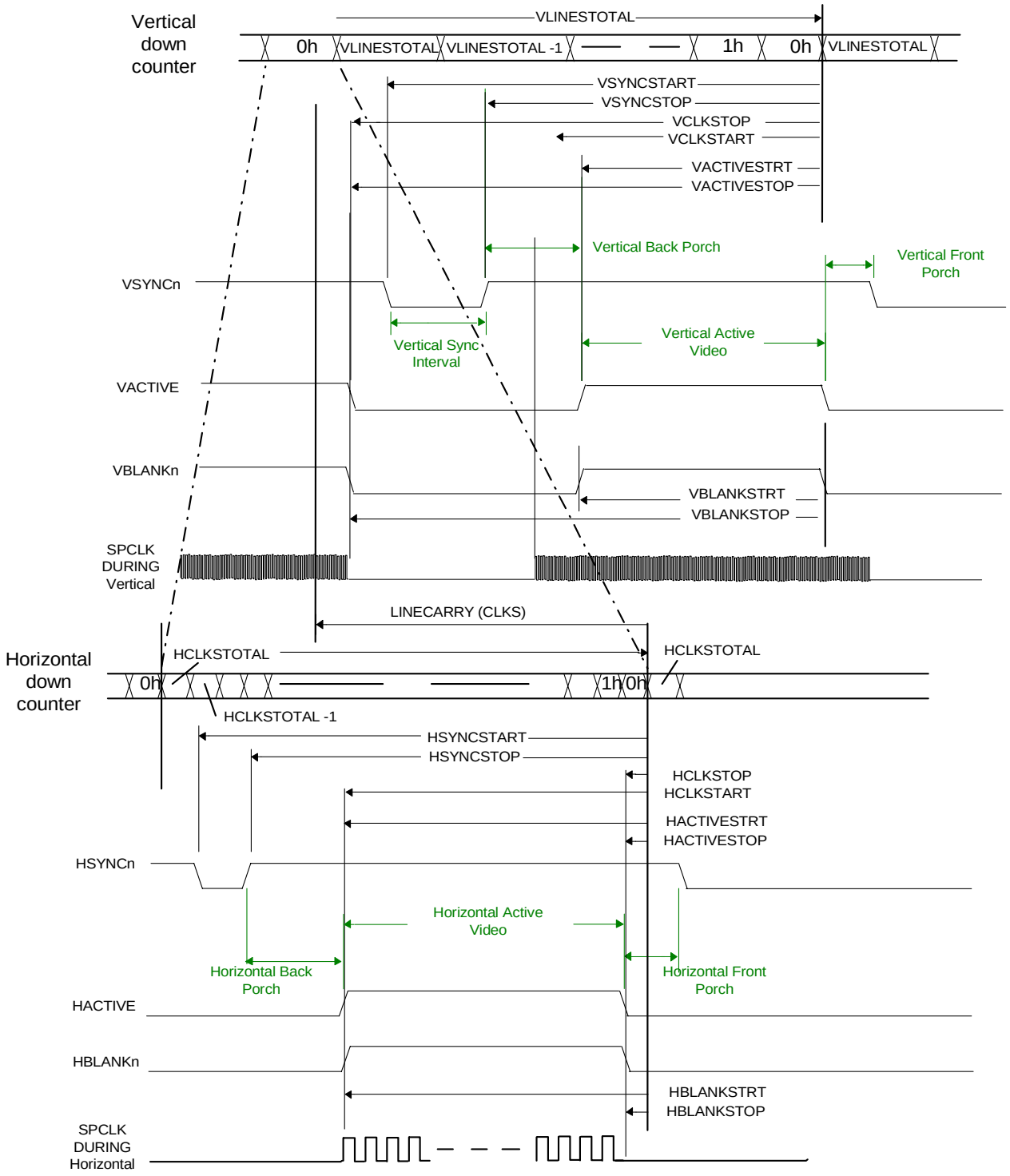


Figure 7-9. Progressive/Dual Scan Video Signals

Copyright 2007 Cirrus Logic

7

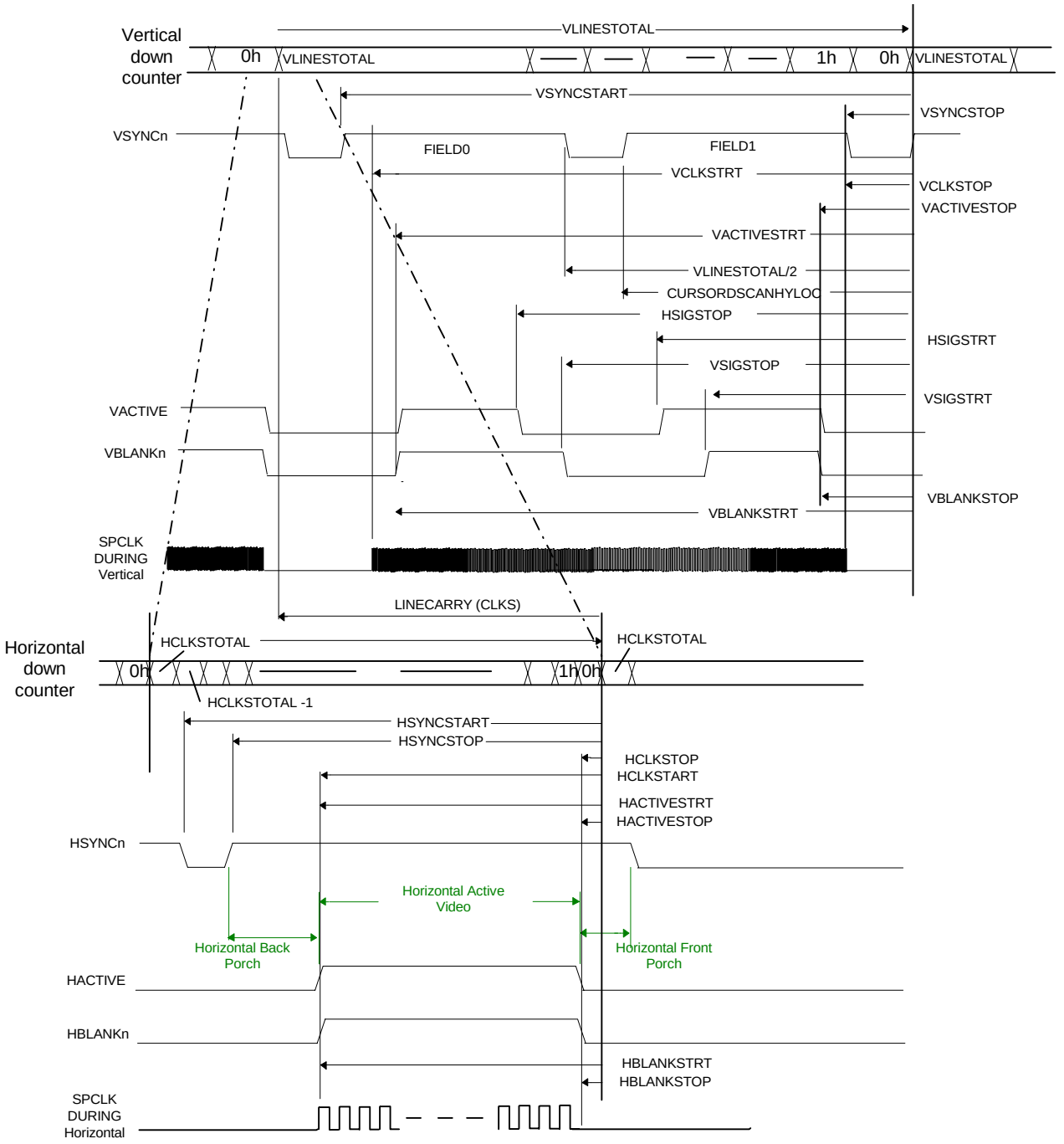


Figure 7-10. Interlaced Video Signals

Copyright 2007 Cirrus Logic

7.4.10.1 Setting the Video Memory Parameters

The Raster Engine uses SDRAM for video frame buffers. The SDRAM locations for the video frame buffers are defined by four registers: "VidScrnPage" , "ScrnLines" , "LineLength" , and "VLineStep" .

7.4.10.1.1 Setting up the VidScrnPage Register

The VidScrnPage register provides the starting address for the video memory relative to the beginning of SDRAM memory space. With the combination of SDSEL in VideoAttribs register, it forms the absolute address for the starting location of the video memory. It is possible to provide for a panning feature by altering the address of the start location at run time. This address also represents the 0,0 pixel position, which is in the upper left corner of the video image.

7.4.10.1.2 Setting up the ScrnLines Register

The "ScrnLines" register is used by the Raster Engine to specify the number of lines of LineLength size that are to be fetched and forwarded to the FIFO. The 'number of lines' must be programmed to be one less than the desired number of lines, because a programmed value of 0x0 specifies a single line. The maximum value is 0x7FF for 2048 lines.

7.4.10.1.3 Setting up the LineLength Register

The "LineLength" register contains the number of 32-bit words that the Raster Engine must fetch from SDRAM for each scan line. This value is always one less than the needed number of 32-bit words because a programmed value of 0x0 specifies a single 32-bit word.

For example, a display width of eighty 8-bit pixels requires that twenty 32-bit words be fetched from the SDRAM video frame buffer for each scan line, since four 8-bit pixels can be packed into a single 32-bit word (80/4=20).

7.4.10.1.4 Setting up the VLineStep Register

At the end of fetching LineLength of data for the first scan line, the Raster Engine will take the value in the "VLineStep" register and add it to the base address ("VidScrnPage") to determine the starting SDRAM address for the next scan line. Generally, this value is the same as LineLength + 0x1. However, it is possible to have an image in SDRAM that is larger then the current display. This larger image can be cropped by the proper programming of "VidScrnPage" , "VLineStep" , and "ScrnLines" registers.

7.4.10.1.5 Memory Setup Example

Assume that a video display is 640 x 480 with a color depth of 4 bpp and that the start of video memory (display pixel coordinate 0,0) is the address determined by SDSEL + 0x1000. The register settings for this example are:

VidScrnPage = 0x1000 (assume SDSEL = 0)

ScrnLines = 480 - 1 = 479 = 0x1DF

LineLength = (640 x 4bpp / 32) - 1 = 79 = 0x4F

Copyright 2007 Cirrus Logic

VLineStep = 640 x 4bpp/32

7

7.4.10.2 PixelMode

Pixel data is transferred from the FIFO to the Video Pixel Mux two 32-bit words at a time (total of 64 bits). Bits[2:0] of the "PixelMode" register specify the pixel depth as shown in Table 7-11. The Video Pixel MUX uses the "PixelMode" register to determine how many pixels are contained in the 64 bits of data. The Video Pixel Mux extracts pixel data from the 64-bits and passes that pixel data to the BLINK logic one pixel at a time.

Table 7-11. Bits P[2:0] in the PixelMode Register

bit P2	bit P1	bit P0	Function
0	0	0	Pixel Multiplexor disabled
0	0	1	4 bits per pixel
0	1	0	8 bits per pixel
1	0	0	16 bits per pixel
1	1	0	24 bits per pixel

Note: All other combinations for these three bits are illegal.

7.4.11 Blink Logic

The blink logic facilitates blinking of individual pixels as they move through the video pipeline. The blink frequency is controlled by the "BlinkRate" register. All blinking pixels blink at the same rate.

7.4.11.1 BlinkRate

This value is used to control the number of video frames that occur before the pixel value that is assigned to blink is switched between its non-blinked and blinked values. The actual rate is calculated by:

Blink cycle = 2 x (1 / VCLK) x HClkTotal x VLinesTotal x (255 - BlinkRate)

where:

VCLK is the basic clock rate of the video logic

HClkTotal is the value contained in the "HClkTotal" register

VLinesTotal is the value contained in the "VLinesTotal" register

BlinkRate is the value contained in the "BlinkRate" register

7.4.11.2 Defining Blink Pixels

A blink pixel must be defined before the blink logic is applied to a given pixel. The "BlinkPattrn" and "PattrnMask" registers are used to define the blink pixels.

Copyright 2007 Cirrus Logic

7.4.11.2.1 PattrnMask Register

This register defines which bits in a pixel are blink bits. To enable an individual bit for comparison requires setting that corresponding bit to "1". To disable an individual bit for comparison set the bit position to "0".

For example, in 8bpp mode, the PattrnMask is defined as 0x0000_0080. This means that the MSB of a pixel is used to assist is defined as a blink bit.

7.4.11.2.2 BlinkPattrn Register

The "BlinkPattrn" register is used to further refine which pixel pattern defines a blink pixel. The pixel value is first masked by the PattrnMask value in the "PattrnMask" register and the result is compared to BlinkPattrn value in the "BlinkPattrn" register. If the comparison results in a match, the pixel is considered to be a valid blink pixel.

For example:

An 8-bit pixel is defined as 0xAF (0b1010_1111b).
PattrnMask is defined as 0x0000_00C0.
BlinkPattrn is defined as 0x0000_0080.

PattrnMask = 0xC0 defines the two MSBs of 8-bit pixels as potential blink bits. If the two corresponding MSBs in the BlinkPattrn register are '10' and the two MSBs of the pixel value are '10', then the pixel of value = 0xAF is a blink pixel. In fact, all pixel values of 10xx_xxxx are blink pixels. If BlinkPattrn was changed to 0x0000_0048 above, a pixel of value 0xAF would not be a blink pixel.

7.4.11.2.3 BlinkMask Register

The "BlinkMask" register is only used if the blink mode definition bits M[3:0] in the "PixelMode" register are set for an AND, OR, or XOR operation. The value in the "BlinkMask" register is ANDed (clearing bits), ORed (setting bits), or XORed (inverting bits) with a pixel that addresses the LUT. The mask allows a blinking pixel to jump from a normal color definition location to a blink color definition location in the LUT.

7.4.11.3 Types of Blinking

Once a pixel has been defined as a blink pixel, it is necessary to provide information on how that pixel will blink. The blink type provides determines what operations are performed on the pixel data as it moves through the blink logic to transform it into a blinking pixel.

There are 10 ways to blink a pixel once it has been defined as a blinking pixel. The blink type is defined by the M[3:0] bits in the "PixelMode" register:

0000 - Blink Disabled

0001 - AND Blinking.

The pixel data is ANDed with the "BlinkMask" register. The modified pixel data will continue through the pipeline.

LUT Blink:

Copyright 2007 Cirrus Logic

If the LUT is enabled, the pixel data is passed to the LUT. The new pixel data value will be used to index into the LUT. The value at that index location will be passed on to the Color Mux.

Non LUT Blink:

If the LUT is not enabled, the modified pixel data is moved directly into the Color Mux. This new pixel value is used by the Color Mux as the 'new' value for the blinking pixel.

0010 - OR Blinking:

The pixel data is ORed with the BlinkMask register. The modified pixel data will continue through the pipeline. See AND blinking for details on the differences between LUT and non-LUT blinking.

0011 - XOR Blinking:

The pixel data is XORed with the "BlinkMask" register. The modified pixel data will continue through the pipeline. See AND blinking for details on the differences between LUT and non-LUT blinking.

0100 - Background Blinking:

The pixel data is replaced with the value in the "BkgrndOffset" register and the new pixel value is placed into the pipeline and sent to the Color Mux.

0101 - Offset Single Blinking:

The pixel data is manipulated by adding the value of the "BkgrndOffset" register with the pixel data. The resulting pixel data will be placed into the pipeline and then sent to the Color Mux.

0110 - Offset 888 Blinking:

The 24 bits of data is made up of three 8-bit values that represent the RGB colors. This mode will treat each of the 8 bit values as a single value, and apply the blinking rules defined for the Offset Single Blinking mode.

The "BkgrndOffset" value is itself treated as an 888 pixel where each of the corresponding 8 bits represent the value that will be added to the corresponding color.

0111 through 1011 - Not used

1100 - Dim Single Blinking:

The pixel that is identified as a blinking pixel is manipulated:

1. The LSB is dropped
2. The remaining bits are shifted right by one
3. The MSB is set to '0'

1101 - Bright Single Blinking:

The pixel that is identified as a blinking pixel is manipulated:

Copyright 2007 Cirrus Logic

1. The MSB is dropped
2. The remaining bits are shifted left by one
3. The LSB is set to '1'

1110 - Dim 888 Blinking:

The 24 bits of data is made up of three 8-bit values that represent the RGB colors. Each of the 8 bit values is treated as a single value, and the blinking rules defined for the Dim Single Blinking mode are applied.

1111 - Bright 888 Blinking:

The 24 bits of data is made up of three 8-bit values that represent the RGB colors. Each of the 8 bit values is treated as a single value, and the blinking rules defined for the Bright Single Blinking mode are applied.

7.4.12 Color Mode Definition

One of four modes may be selected to define pixel color: Pixel Look-Up Table Mode, Triple 8-Bit Mode, 16-Bit 565 Mode, and 16-Bit 555 Mode.

7.4.12.1 Pixel Look-up Table Mode

The Raster Engine contains a 256 x 24 bit RAM that is used as pixel look-up-table (LUT) for pixel depths up to 8-bits. Appropriate blink operations, if any, are performed on the pixel data fetched from the video memory and the resulting pixel data value is used as an index into the LUT. The pixel value located at the index position continues through the video pipeline.

The LUT is memory mapped and may be written at any time. However, if it is written during a non-blanking interval, the display may be momentarily corrupted.

Writing 0x0 to the C[3:0] bits (color bits) in the PixelMode register to 0x0 enables the LUT.

7.4.12.2 Triple 8-bit Color Definition Mode

The 24 bits of data is divided into three color planes, where the RED, GREEN, and BLUE each have 8 bits of color definition.

7.4.12.3 16-bit 565 Color Definition Mode

The 16 bits of data is divided into three color planes, where the RED and BLUE each have 5 bits for color definition and the GREEN has 6 bits for color definition.

7.4.12.4 16-bit 555 Color Definition Mode

The 16 bits of data is divided into three color planes, where the RED, GREEN, and BLUE each have 5 bits of color definition. The MSB of the 16-bit data is not used.

Copyright 2007 Cirrus Logic

7.5 Registers

Table 7-12. Raster Engine Register List

Address	Name	SW locked	Type	Size	Description
0x8003_0000	VLinesTotal	Write	Read/Write	11 bits	Total Number of vertical frame lines
0x8003_0004	VSyncStrtStop	Write	Read/Write	22 bits	Vertical sync pulse setup
0x8003_0008	VActiveStrtStop	Write	Read/Write	22 bits	Vertical active setup
0x8003_0228	VBlankStrtStop	Write	Read/Write	22 bits	Vertical blanking setup
0x8003_000C	VClkStrtStop	Write	Read/Write	22 bits	Vertical clock active frame
0x8003_0010	HClkTotal	Write	Read/Write	11 bits	Total Number of horizontal line clocks
0x8003_0014	HSyncStrtStop	Write	Read/Write	22 bits	Horizontal sync pulse setup
0x8003_0018	HActiveStrtStop	Write	Read/Write	22 bits	Horizontal active setup
0x8003_022C	HBlankStrtStop	Write	Read/Write	22 bits	Horizontal blanking setup
0x8003_001C	HClkStrtStop	Write	Read/Write	22 bits	Horizontal clock active frame
0x8003_0020	Brightness	No	Read/Write	16 bits	PWM brightness control
0x8003_0024	VideoAttribs	Write	Read/Write	16 bits	Video state machine parameters
0x8003_0028	VidScrnPage	No	Read/Write	32 bits	Starting address of video screen
0x8003_002C	VidScrnHPage	No	Read/Write	32 bits	Starting address of video screen half page
0x8003_0030	ScrnLines	No	Read/Write	11 bits	Number of active lines scanned to the screen
0x8003_0034	LineLength	No	Read/Write	12 bits	Length in words of data for lines
0x8003_0038	VLineStep	No	Read/Write	13 bits	Memory step for each line
0x8003_003C	LineCarry	Write	Read/Write	11 bits	Horizontal/vertical offset parameter
0x8003_0040	BlinkRate	No	Read/Write	8 bits	Blink counter setup
0x8003_0044	BlinkMask	No	Read/Write	24 bits	Logic mask applied to pixel to perform blink operation
0x8003_0048	BlinkPattrn	No	Read/Write	24 bits	Compare value for determining blinking pixels.
0x8003_004C	PattrnMask	No	Read/Write	24 bits	Mask to limit pattern.
0x8003_0050	BkgrndOffset	No	Read/Write	24 bits	Background color or blink offset value.
0x8003_0054	PixelMode	No	Read/Write	15 bits	Pixel mode definition setup register.
0x8003_0058	ParIIfOut	No	Read/Write	9 bits	Parallel interface write/control register.
0x8003_005C	ParIIfIn	No	Read/Write	8 + 8 bits	Parallel interface read/setup register.

Copyright 2007 Cirrus Logic

Table 7-12. Raster Engine Register List (Continued)

Address	Name	SW locked	Type	Size	Description
0x8003_0060	CursorAdrStart	No	Read/Write	32 bits	Word location of the top left corner of cursor to be displayed.
0x8003_0064	CursorAdrReset	No	Read/Write	32 bits	Location of first word of cursor to be scanned after last line.
0x8003_0068	CursorSize	No	Read/Write	16 bits	Cursor height, width, and step size register.
0x8003_006C	CursorColor1,	No	Read/Write	24 bits	Cursor color overlaid when cursor value is 10.
0x8003_0070	CursorColor1,	No	Read/Write	24 bits	Cursor color overlaid when cursor value is 11.
0x8003_0074	CursorXYLoc	No	Read/Write	11 +1 + 11 bits	Cursor X and Y location register
0x8003_0078	CursorDScanLHYLoc	No	Read/Write	1 + 11 bits	Cursor dual scan lower half Y location register
0x8003_021C	CursorColor2,	No	Read/Write	24 bits	Color when cursor value is 10 and cursor is in blink state.
0x8003_0220	CursorBlinkColor1,	No	Read/Write	24 bits	Color when cursor value is 11 and cursor is in blink state.
0x8003_0224	CursorBlinkRateCtrl	No	Read/Write	1+8 bits	Enable and rate for cursor blinking.
0x8003_007C	RasterSWLock	Read	Read/Write	8 bits	Software Lock register. This register unlocks registers that have a SWLOCK.
0x8003_0080 - 0x8003_00FC	GrySclLUTR,	No	Read/Write	32 x 19	Grayscale matrix Red
0x8003_0200	VidSigRsltVal	No	Read Only	16 bits	Video signature result value.
0x8003_0204	VidSigCtrl	No	Read / Write	32 bits	Video signature Control register.
0x8003_0208	VSigStrtStop	No	Read/Write	11 + 11 bits	vertical signature bounds setup
0x8003_020C	HSigStrtStop	No	Read/Write	11 + 11 bits	Horizontal signature bounds setup
0x8003_0210	SigClrStr	No	Read/Write	11 + 11 bits	Signature clear and store location
0x8003_0214	ACRate	No	Read/Write	11 bits	LCD AC voltage bias control counter setup
0x8003_0218	LUTSwCtrl	No	Read/Write	2 bits	LUT switching control
0x8003_0230	EOLOffset	No	Read/Write	16 bits	End of line offset register
0x8003_0234	FIFOLevel	No	Read/Write	6 bits	FIFO fill level register
0x8003_0280 - 0x8003_02FC	GrySclLUTG,	No	Read/Write	32 x 19	Grayscale matrix Green
0x8003_0300 - 0x8003_037C	GrySclLUTB	No	Read/Write	32 x 19	Grayscale matrix Blue
0x8003_0400 - 0x8003_07FC	ColorLUT	No	Read/Write	256 x 24 RAM	Color Look-Up-Table

Note: The raster engine registers are intended to be word accessed only. Since the least significant bytes of the address bus are not decoded, byte and half word accesses are illegal and may have unpredictable results.

Copyright 2007 Cirrus Logic

VLinesTotal

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD					TOTAL										

Address: 0x8003_0000

Default: 0x0000_0000

Definition: Total horizontal lines that compose a vertical frame

Bit Descriptions:

RSVD: Reserved - Unknown during read

TOTAL: VLines Total - Read/Write

The VLines Total value written to this field specifies the total number of horizontal lines for a video frame including synchronization, blanking, and active lines. This value is used to preset the Vertical down counter. Please refer to video the signalling timing diagrams shown in Figure 7-9 and Figure 7-10.

VSyncStrtStop

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD					STOP										

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD					STRT										

Address: 0x8003_0004

Default: 0x0000_0000

Definition: Vertical Sync Pulse Start/Stop register

Bit Descriptions:

RSVD: Reserved - Unknown during read

STOP: Stop - Read/Write

Copyright 2007 Cirrus Logic

When the Vertical counter counts down to the written STOP value, the VSYNC signal on the V_CSYNC pin will go inactive if CSYNC = '0' and SYNCEN = '1' in the VideoAttribs register. Please refer to the video signalling timing diagrams shown in Figure 7-9 and Figure 7-10.

STRT: Start - Read/Write

When the Vertical counter counts down to the written STRT value, the VSYNC signal on the V_CSYNC pin will go active if CSYNC = '0' and SYNCEN = '1' in the VideoAttribs register.

VActiveStrtStop

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				STOP											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				STRT											

Address: 0x8003_0008

Default: 0x0000_0000

Definition: Vertical Active Start/Stop register

Bit Descriptions:

RSVD: Reserved - Unknown during read

STOP: Stop - Read/Write

The STOP value is the value of the Vertical down counter at which the VACTIVE signal becomes inactive (stops). This indicates the end of the active video portion for the Vertical frame. Please refer to the video signalling timing diagrams in Figure 7-9 and Figure 7-10. VACTIVE is an internal block signal. The active video interval is controlled by the logical OR of VACTIVE and HACTIVE.

STRT: Start - Read/Write

The STRT value is the value of the Vertical down counter at which the VACTIVE signal becomes active (starts). This indicates the start of the active video portion for the Vertical frame. Please refer to the video signalling timing diagrams in Figure 7-9 and Figure 7-10. VACTIVE is an internal block signal. The active video interval is controlled by the logical OR of VACTIVE and HACTIVE.

Copyright 2007 Cirrus Logic

VBlankStrtStop

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				STOP											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				STRT											

Address: 0x8003_0228

Default: 0x0000_0000

Definition: Vertical BLANK signal Start/Stop register

Bit Descriptions:

RSVD: Reserved - Unknown during read

STOP: Stop - Read/Write

The STOP value is the value of the Vertical down counter at which the VBLANKn signal becomes inactive (stops). This is used to generate the BLANKn signal that is used by external devices and indicates the end of the active video portion for the Vertical frame. Please refer to video signalling timing diagrams in Figure 7-9 and Figure 7-10. VBLANKn is an internal block signal. The NBLANK output is a logical AND of NVBLANK and HBLANKn.

STRT: Start - Read/Write

The STRT value is the value of the Vertical down counter at which the VBLANKn signal becomes active (starts). This is used to generate the BLANKn signal that is used by external devices and indicates the start of the active video portion for the Vertical frame. Please refer to video signalling timing diagrams in Figure 7-9 and Figure 7-10. VBLANKn is an internal block signal. The NBLANK output is a logical AND of NVBLANK and HBLANKn.

Copyright 2007 Cirrus Logic

VClkStrtStop

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				STOP											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				STRT											

Address: 0x8003_000C

Default: 0x0000_0000

Definition: Vertical Clock Start/Stop register

Bit Descriptions:

RSVD: Reserved - Unknown during read

STOP: Stop - Read/Write

The STOP timing register contains the value of the Vertical down counter at which the VCLKEN signal goes inactive (stops). This indicates the end of the video clock for the Vertical frame. Please refer to video signalling timing diagrams in Figure 7-9 and Figure 7-10. VCLKEN is an internal block signal. The SPCLK output is enabled by the logical AND of VCLKEN and HCLKEN.

STRT:Start - Read/Write

The STRT timing register contains the value of the Vertical down counter at which the VCLKEN signal becomes active (starts). This indicates the start of the video clock for the Vertical frame. Please refer to video signalling timing diagrams in Figure 7-9 and Figure 7-10. VCLKEN is an internal block signal. The SPCLK output is enabled by the logical AND of VCLKEN and HCLKEN.

Copyright 2007 Cirrus Logic

HClkTotal

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD					TOTAL										

Address: 0x8003_0010

Default: 0x0000_0000

Definition: Total pixel clocks that compose a horizontal line

Bit Descriptions:

RSVD: Reserved - Unknown during read

TOTAL: Total - Read/Write

The HClk Total timing register contains the total number of clocks for a horizontal video line including synchronization, blanking, and active clocks. This value is used to preset the Horizontal down counter. Please refer to video signalling timing diagrams in Figure 7-9 and Figure 7-10.

HSyncStrtStop

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD					STOP										

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD					STRT										

Address: 0x8003_0014

Default: 0x0000_0000

Definition: HorizontaL Sync Start/Stop Register

Bit Descriptions:

RSVD: Reserved - Unknown during read

STOP: Stop - Read/Write

Copyright 2007 Cirrus Logic

The STOP value is the horizontal down counter value at which the HSYNCn signal becomes inactive (stops). When the Horizontal counter counts down to the STOP value, the HSYNCn signal goes inactive. Please refer to video signalling timing diagrams in Figure 7-9 and Figure 7-10.

STRT:Start - Read/Write

The STRT value is the horizontal down counter value at which the HSYNCn signal becomes active (starts). When the Horizontal counter counts down to the STRT value, the HSYNCn signal goes active (starts). Please refer to video signalling timing diagrams in Figure 7-9 and Figure 7-10.

HActiveStrtStop

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD					STOP										

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD					STRT										

Address: 0x8003_0018

Default: 0x0000_0000

Definition: Horizontal Active period Start/Stop register

Note: When horizontal clock gating is required, set the STRT and STOP fields in the HActiveStrtStop register to the STRT and STOP values in HClkStrtStop + 5. This is a programming requirement that is easily overlooked.

Bit Descriptions:

RSVD: Reserved - Unknown during read

STOP: Stop - Read/Write

The STOP value is the value of the Horizontal down counter at which the HACTIVE signal becomes inactive (stops). This indicates the end of the active video portion for the Horizontal line. Please refer to video signalling timing diagrams in Figure 7-9 and Figure 7-10. HACTIVE is an internal block signal. The active video interval is controlled by the logical OR of VACTIVE and HACTIVE.

Copyright 2007 Cirrus Logic

7

STRT: Start - Read/Write

The STRT value is the value of the Horizontal down counter at which the HACTIVE signal becomes active (starts). This indicates the start of the active video portion for the Horizontal line. Please refer to video signalling timing diagrams in Figure 7-9 and Figure 7-10. HACTIVE is an internal block signal. The active video interval is controlled by the logical OR of VACTIVE and HACTIVE.

HBlankStrtStop

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD								STOP							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								STRT							

Address: 0x8003_022C

Default: 0x0000_0000

Definition: Horizontal Blank signal Start/Stop register

Bit Descriptions:

RSVD: Reserved - Unknown during read

STOP: Stop - Read/Write

The STOP value is the value of the Horizontal down counter at which the HBLANK signal becomes inactive (stops). This is used to generate the BLANKn signal that is used by external devices to indicate the end of the active video portion for the Horizontal line. Please refer to video signalling timing diagrams in Figure 7-9 and Figure 7-10. HBLANK is an internal clock signal. The BLANKn output is a logical AND of VBLANK and HBLANK.

STRT:Start - Read/Write

The STRT value is the value of the Horizontal down counter at which the HBLANK signal becomes active (starts). This is used to generate the BLANKn signal that is used by external devices to indicate the start of the active video portion for the Horizontal line. Please refer to video signalling timing diagrams in Figure 7-9 and Figure 7-10. HBLANK is an internal clock signal. The BLANK output is a logical AND of VBLANK and HBLANK

Copyright 2007 Cirrus Logic

HClkStrtStop

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				STOP											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				STRT											

Address: 0x8003_001C

Default: 0x0000_0000

Definition: Horizontal Clock Active Start/Stop register

Note: When horizontal clock gating is required, set the STRT and STOP fields in the HActiveStrtStop register to the STRT and STOP values in HClkStrtStop + 5. This is a programming requirement that is easily overlooked.

Bit Descriptions:

RSVD: Reserved - Unknown during read

STOP: Stop - Read/Write

The STOP value is the value of the Horizontal down counter at which the HCLKEN signal becomes inactive (stops). This indicates the end of the video clock for the Horizontal frame. Please refer to video signalling timing diagrams in Figure 7-9 and Figure 7-10. HCLKEN is an internal clock signal. The SPCLK output is enabled by the logical AND of VCLKEN and HCLKEN.

STRT: Start - Read/Write

The STRT value is the value of the Horizontal down counter at which the HCLKEN signal becomes active (starts). This indicates the start of the video clock for the Horizontal frame. Please refer to video signalling timing diagrams in Figure 7-9 and Figure 7-10. HCLKEN is an internal clock signal. The SPCLK output is enabled by the logical AND of VCLKEN and HCLKEN.

Copyright 2007 Cirrus Logic

VidScrnPage

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				PAGE											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAGE														NA	

Address: 0x8003_0028

Default: 0x0000_0000

Definition: Video Screen Page Register

Bit Descriptions:

RSVD: Reserved - Unknown during read

PAGE: Video Screen Page Starting SDRAM Address - Read/Write

Corresponds to the word address relative to the beginning of SDRAM of the upper left corner of the video screen to be scanned out. The absolute AHB address for the video screen page is determined by the combination of this bit field as well as the SDSEL bit held in the "VideoAttribs" register.

NA: Not Assigned. Will return written value during a read.

VidScrnHPage

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				PAGE											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAGE														NA	

Address: 0x8003_002C

Default: 0x0000_0000

Definition: Video Screen Half Page Register

Bit Descriptions:

RSVD: Reserved - Unknown during read

Copyright 2007 Cirrus Logic

PAGE: Video Screen Half-page Starting SDRAM Address - Read/Write

If DSCAN = '1' in the PixelMode register, the Video Screen Half-page Starting SDRAM Address value written to this field corresponds to the upper left corner of the bottom half of the video screen.

NA: Not Assigned. Will return written value during a read.

ScrnLines

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				LINES											

Address: 0x8003_0030

Default: 0x0000_0000

Definition: Video Screen Lines Register

Bit Descriptions:

RSVD: Reserved - Unknown during read

LINES: Lines - Read/Write

The Lines value written to this field specifies the number of lines to be scanned to the display during normal and half-page mode operation.

LineLength

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				LEN											

Address: 0x8003_0034

Default: 0x0000_0000

Definition: Video Line Length Register

Bit Descriptions:

RSVD: Reserved. Unknown during read.

Copyright 2007 Cirrus Logic

LEN: Length - Read/Write

The Length value written to this field specifies, in 32-bit words, the length of video lines that are scanned to the display. Please see "Setting up the LineLength Register" on page 7-31 and "Memory Setup Example" on page 7-31.

The remainder of the last word in a video line may not be used as long as the blanking time is greater than the remaining number of pixels. The extra pixels will enter the video chain, but will exit the pipeline during the blanking interval. When the end of LEN is reached, STEP in the VLineStep register is added to the address for video data.

VLineStep

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				STEP											

Address: 0x8003_0038

Default: 0x0000_0000

Definition: Video Line Step Size Register

Bit Descriptions:

RSVD: Reserved - Unknown during read

STEP: Step - Read/Write

When the end of the video line is reached (see LEN in LineLength register), the Step value written to this field (specified in 32-bit words) is added to the address for every video line that is scanned to the display. Please see "Memory Setup Example" on page 7-31.

This allows the screen width to be smaller than the video image width in SDRAM.

Copyright 2007 Cirrus Logic

LineCarry

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				LCARY											

Address: 0x8003_003C

Default: 0x0000_0000

Definition: Horizontal Line Carry Value register

Bit Descriptions:

RSVD: Reserved - Unknown during read

LCARY: Line Carry - Read/Write

When the Horizontal down counter counts down to the written LCARY value, a carry is sent to increment the Vertical counter. This provides for timing skew between the vertical and horizontal video signals. Please refer to the video signalling timing diagrams in Figure 7-9 and Figure 7-10.

EOLOffset

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET															

Address: 0x8003_0230

Default: 0x0000_0000

Definition: End-of-line Offset Register.

Bit Descriptions:

RSVD: Reserved - Unknown during read

OFFSET: Offset - Read/Write

The Offset value written to this field is added to the address at the end of every other video line if the Offset value is not 0x0. This allows splitting the left and right halves of the display.

Copyright 2007 Cirrus Logic

If the Offset value is 0x0, no offset is used and addressing proceeds normally.

7

Other Video Registers

Brightness

Address: 0x8003_0020

Default: 0x0000_0000

Definition: Brightness Control register.

Bit Descriptions:

RSVD: Reserved - Unknown during read

CMP: Compare - Read/Write

The Compare value written to this field determines the brightness control duty cycle (see CNT below) - that is, when the brightness signal to the BRIGHT pin is '1' or '0'.

CNT: Count - Read/Write

The Count value written to this field specifies the number of horizontal lines counted during a brightness waveform period. The counter counts down from the Count value to 0x0.

The CNT value and the CMP value are used to construct a brightness control waveform on the BRIGHT pin by this relationship:

When Count > Compare, or Count = Compare, the brightness signal to the BRIGHT pin is '0'.

When Count < Compare, the brightness signal to the BRIGHT pin is '1'.

The BRIGHT pin is '0' (zero% brightness) after reset.

Copyright 2007 Cirrus Logic

VideoAttribs

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD									SDSEL		BKPXD	DVERT	DHORZ	EQUSER	INTRLC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT	INTEN	PIFEN	CCIREN	RSVD	LCDEN	ACEN	INVCLK	BLKPOL	HSPOL	V/CPOL	CSYNC	DATEN	SYNCEN	PCLKEN	EN

Address: 0x8003_0024

Default: 0x0000_0000

Definition: Video Signal Attributes register.

Bit Descriptions:

RSVD: Reserved - Unknown during read

SDSEL: SDRAM Selector - Read/Write

Writing to these two bits defines which SDCSn[3:0] pin is used to access the video frame buffer in SDRAM:

00 SDCSn[0]
01 SDCSn[1]
10 SDCSn[2]
11 SDCSn[3]

SDCSn[3] is selected by default on hardware reset.

BKPXD: Blank Pixel Data - Read/Write

Writing BKPXD = '1' forces the pixel data on the P[17:0] pins to be 0x0 when the blanking signal on the BLANK pin is '0'.

0 - Disable

1 - Enable

This allows the use of an inexpensive external DAC that does not contain data blanking logic.

DVERT: Double Vertical - Read/Write

Writing DVERT = '1' forces the values of the defined bit-fields in the VLinesTotal, VSyncStrtStop, VActiveStrtStop, VBlankStrtStop, and VClkStrtStop registers to be doubled (2X programmed value) when used.

0 - Disable

1 - Enable

Copyright 2007 Cirrus Logic

DHORZ: Double Horizontal - Read/Write

Writing DHORZ = '1' forces the values of the defined bit-fields in the HClkTotal, HSyncStrtStop, HActiveStrtStop, HBlankStrtStop, and HClkStrtStop registers to be doubled (2X programmed value) when used.

0 - Disable

1 - Enable

EQUSER: Equalization/Serration - Read/Write

If SYNCEN = '1' and CSYNC = '1' (both defined below), writing EQUSER = '1' forces equalization and serration pulses to be inserted into the composite synchronization signal on the V_CSYNC pin.

0 - Disable

1 - Enable

INTRLC: Interlace - Read/Write

Writing INTRLC = '1' enables interlaced frame timing.

0 - Disable

1 - Enable

INT: Interrupt - Read/Write

If INTEN = '1', an INT = '1' status indicates that the end of active video interrupt has occurred.

0 - No interrupt

1 - Interrupt occurred

Write "0" to clear, write "1" to test.

INTEN: Interrupt Enable - Read/Write

Writing INTEN = '1' enables the end of active video interrupt.

0 - Disable
1 - Enable

PIFEN: Parallel Interface Enable - Read/Write

0 - Enable interface for normal display operation
1 - Enable interface for Smart Panel operation

Writing PIFEN = '1' redefines the signals on these pins for Smart Panel operation:

Copyright 2007 Cirrus Logic

V_CSYNC --> D7 (Smart Panel)

HSYNC --> D6

BLANK --> D5

P17 --> D4

P3 --> D3

P[2:0] --> D[2:0]

SPCLK --> E

A Smart Panel has an integrated controller and frame buffer. Smart Panel R/W and RS signals must be implemented via GPIOs and controlled via software.

CCIREN: CCIR Enable - Read/Write

The value written to this bit selects which video output signals are generated:

0 - Normal signals

1 - CCIR656 YCrCb digital video signals

LCDEN: LCD Enable - Read/Write

The value written to this bit specifies the function of the signals to the P[16] pin and P[15] pin:

0 - Pixel data bits 16 and 15 are routed to pins P16 and P15, respectively

1 - XECL and YSCL signals are routed to pins P16 and P15, respectively. The XECL and YSCL signals are used to enable LCD drivers and register shifting

ACEN: AC Enable - Read/Write

Writing ACEN = '1' routes an LCD AC Waveform to pin P17.

0 - Pixel data bit 17 is routed to pin P17

1 - LCD AC Wave Form is routed to pin P17. The waveform toggles with each new vertical frame.

INVCLK: Invert Pixel Clock - Read/Write

The value written to this bit selects the active edge of SPCLK on the SPCLK pin:

0 - Pixel data output changes on the rising edge of the clock on the SPCLK pin

Copyright 2007 Cirrus Logic

1 - Pixel data output changes on falling edge of the clock on the SPCLK pin

BLKPOL: Blank Polarity - Read/Write

The value written to this bit selects the polarity of the blanking signal on the BLANK pin:

0 - BLANK is active LOW (default)

1 - BLANK is active HIGH

HSPOL: Horizontal Sync Polarity - Read/Write

The value written to this bit selects the polarity of the horizontal synchronization signal on the HSYNC pin:

0 - HSYNC is active LOW (default)

1 - HSYNC is active HIGH

V/CPOL: Vertical / Composite Polarity - Read/Write

The value written to this bit selects the polarity of the synchronization signal on the V_CSYNC pin:

0 - V_CSYNC is active LOW (default)

1 - V_CSYNC is active HIGH

CSYNC: Composite Sync - Read/Write

The value written to this bit selects whether the Vertical Sync or the Composite Sync signal is routed to the V_CSYNC pin:

0 - Vertical Sync

1 - Composite Sync

DATEN: Pixel Data Enable - Read/Write

The value written to this bit selects whether pixel data is output to the P[x] pins, or not:

0 - Pixel data output disabled

1 - Pixel data output enabled

SYNCEN: Video Sync Enable - Read/Write

The value written to this bit selects whether synchronization signals are output to the H_SYNC and V_CSYNC pins, or not:

0 - Video SYNC outputs disabled

1 - Video SYNC outputs enabled

Copyright 2007 Cirrus Logic

PCLKEN: Pixel Clock Enable - Read/Write

The value written to this bit selects whether the pixel clock or smart panel clock are output to the SPCLK pin, or not:

0 - SPCLK pin at high impedance

1 - PCLK or SCLK active on SPCLK pin

The PIFEN bit above selects PCLK vs. SCLK.

EN: Enable Video State Machine - Read/Write

The value written to this bit selects whether the video state machine is enabled, or not:

0 - Video state machine off

1 - Video state machine enabled

RasterSWLock

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								SWLCK							

Address: 0x8003_007C

Default: 0x0000_0000

Definition: Raster Software Lock register

Bit Descriptions:

RSVD: Reserved - Unknown during read

SWLCK: Software Lock - Read/Write

WRITE: Writing 0X0000_00AA to this register will unlock all locked registers until the next block access.

READ: During a read operation, SWLCK[0] has this meaning:

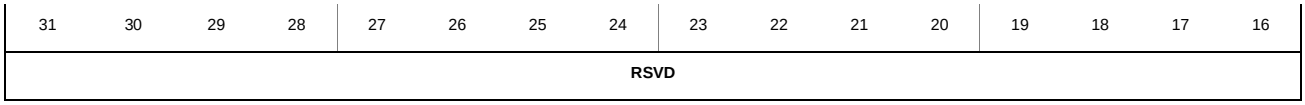
1 - Unlocked for current bus access

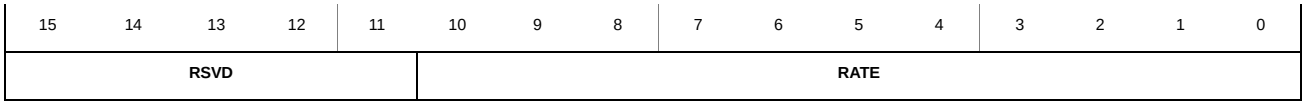
0 - Locked

The Read feature of the RasterSWLock register is used for testing the locking function.

Copyright 2007 Cirrus Logic

ACRate





Address: 0x8003_0214

Default: 0x0000_0000

Definition: AC Toggle Rate register

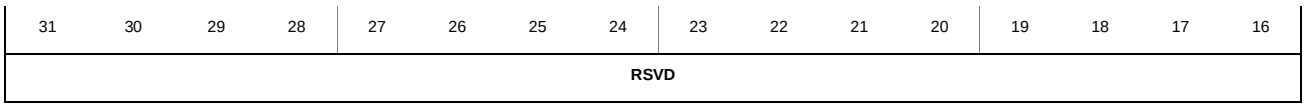
Bit Descriptions:

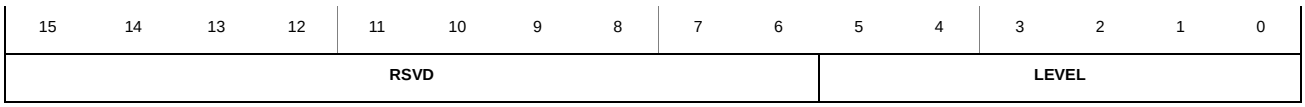
RSVD: Reserved - Unknown during read

RATE: Rate - Read/Write

The RATE field must be written with a value that is one less than the number of horizontal video lines before the AC LCD bias signal is to toggle. Care must be taken when choosing this value while using the grayscale dithering algorithms, as a DC build-up may occur if the pixel timing for the 'on' state of the pixel is concurrent with the bias frequency.

FIFOLevel





Address: 0x8003_0234

Default: 0x0000_0010

Definition: FIFO Refill Level register

Bit Descriptions:

RSVD: Reserved - Unknown during read

LEVEL: Level - Read/Write

Copyright 2007 Cirrus Logic

This field should be written with a value that specifies the number of words that the FIFO empties before the FIFO requests that it be refilled. Values greater than 16 should be used with extreme caution as they can cause the Raster Engine to underflow, causing video jitter or other visual defects.

PixelMode

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRBSW	DSCAN	C						M				S			P

Address: 0x8003_0054

Default: 0x0000_0000

Definition: Pixel Mode register

Bit Descriptions:

RSVD: Reserved - Unknown during read

0: Must be written as '0'

TRBSW: Two and Two-Thirds Red/Blue Swap - Read/Write

Writing a Two and two-thirds Red/Blue Swap value to this bit selects the ordering of Red and Blue pixels for data shifted displays:

0 - Normal: Blue is the low order bits followed by green and red

1 - Reverse: Red is low order bits followed by green and blue

DSCAN: Dual Scan - Read/Write

Copyright 2007 Cirrus Logic

Writing a Dual Scan value to this bit selects whether the display is used in single scan mode, or dual scan mode where the display is divided into a 'top' half and a 'bottom' half. In dual scan mode, the video frame buffer in SDRAM must be organized such that 'top' and 'bottom' pixels alternate in consecutive locations. 'Top' and 'bottom' pixels are fetched and input to the Raster Engine's video pipeline. The output shifter is set up to drive the top and bottom half screen data at the same time. Dual scan mode is intended for passive matrix LCD screens that require both halves of the screen to be scanned out at the same time. However, dual scan mode could also be used to drive two separate synchronized displays, each with different data.

0 - Single Scan (full page)

1 - Dual Scan (two half pages)

C: Color - Read/Write

The Color Mode is specified by selecting a value from Table 7-13 and writing it to this field.

Table 7-13. Color Mode Definition Table

C3	C2	C1	C0	Color Mode
0	0	0	0	Use LUT Data
0	1	0	0	Triple 8 bits per channel
0	1	0	1	16-bit 565 color mode
X	1	1	0	16-bit 555 color mode
1	X	X	X	Grayscale Palettes Enabled

M: Mode - Read/Write

The Blink Mode is specified by selecting a value from Table 7-14 and writing it to this field.

Table 7-14. Blink Mode Definition Table

M3	M2	M1	M0	Blink Mode
0	0	0	0	Blink Mode Disabled
0	0	0	1	Pixels ANDed with Blink Mask
0	0	1	0	Pixels ORed with Blink Mask
0	0	1	1	XORed with Blink Mask
0	1	0	0	Blink to background register Value

Copyright 2007 Cirrus Logic

Table 7-14. Blink Mode Definition Table (Continued)

M3	M2	M1	M0	Blink Mode
0	1	0	1	Blink to offset color single value mode
0	1	1	0	Blink to offset color 888 mode (555,565)
0	1	1	1	Undefined
1	1	0	0	Blink dimmer single value mode
1	1	0	1	Blink brighter single value mode
1	1	1	0	Blink dimmer 888 mode (555,565)
1	1	1	1	Blink brighter 888 mode (555,565)

S: Shift - Read/Write

The Shift Mode is specified by selecting a value from Table 7-15 and writing it to this field.

Table 7-15. Output Shift Mode Table

S2	S1	S0	Shift Mode
0	0	0	1 - pixel per pixel clock (up to 24 bits wide)
0	0	1	1 - pixel mapped to 18 bits each pixel clock
0	1	0	2 - pixels per shift clock (up to 9 bits wide each)
0	1	1	4 - pixels per shift clock (up to 4 bits wide each)
1	0	0	8 - pixels per shift clock (up to 2 bits wide each)
1	0	1	2 2/3 3-bit pixels per clock over 8 bit bus
1	1	0	Dual Scan 2 2/3 3-bit pixels per clock over 8-bit bus
1	1	1	Undefined - Defaults to 1 - pixel per pixel clock

P: Pixel - Read/Write

The number of bits per pixel that are output on the P[x] pins is specified by selecting a value from Table 7-16 and writing it to this field.

The Graphics Engine has a separate setting for this value, which may or may not be the same.

Table 7-16. Bits per Pixel Scanned Out

P2	P1	P0	Pixel Mode
0	0	0	pixel multiplexer disabled
0	0	1	4 bits per pixel
0	1	0	8 bits per pixel
0	1	1	do not use

Copyright 2007 Cirrus Logic

7

Table 7-16. Bits per Pixel Scanned Out (Continued)

P2	P1	P0	Pixel Mode
1	0	0	16 bits per pixel
1	0	1	do not use
1	1	0	24 bits per pixel packed
1	1	1	32 bits per pixel (24 bits per pixel unpacked)

ParIIfOut

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD							RD	DAT							

Address: 0x8003_0058

Default: 0x0000_0000

Definition: Parallel Interface Output/Control Register.

This register, if PIFEN = '1' in the VideoAttribs register, is used to access a Smart Panel. A Smart Panel has an integrated controller and frame buffer.

Bit Descriptions:

RSVD: Reserved - Unknown during read

RD: Read control bit - Write Only

Writing a '0' to this bit location will initiate a parallel interface write cycle; writing a '1' will initiate a parallel interface read cycle:

1 - Start Smart Panel write cycle

0 - Start Smart Panel read cycle

DAT: Data - Write Only

The value written to this field is output on the parallel interface pins during a write cycle. Writing PIFEN = '1' to the VideoAttribs register redefines the signals on these pins for Parallel Interface (Smart Panel) operation:

V_CSYNC --> D7 (Smart Panel)

HSYNC --> D6

BLANK --> D5

P17 --> D4

Copyright 2007 Cirrus Logic

P3 --> D3

P[2:0] --> D[2:0]

SPCLK --> E

Smart Panel R/W and RS signals must be implemented via GPIOs and controlled via software.

ParlIIfIn

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD								ESTRT				CNT			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								DAT							

Address: 0x8003_005C

Default: 0x0000_0000

Definition: Parallel Interface Output/Control Register

This register, if PIFEN = '1' in the VideoAttribs register, is used to access a Smart Panel. A Smart Panel has an integrated controller and frame buffer.

Bit Descriptions:

RSVD: Reserved - Unknown during read

ESTRT: Enable Start - Read/Write

The Enable Signal Start Value for the parallel interface down counter should be written to this field. When the parallel interface counter counts down to this value during a write cycle (see RD bit in the ParlIIfOut register for write cycle), the E enable signal on the E pin goes active.

The E enable signal becomes inactive just before the counter counts down to 0x0, although data remains driven on the D[7:0] pins throughout the 0x0 count. This allows data to be driven for one additional clock cycle, providing data hold time to the Smart Panel.

CNT: Count - Read/Write

The counter preload value that is written to this field gets loaded into the parallel interface down counter. When a write or read cycle is initiated by writing to the RD bit in the ParlIIfOut register, the counter begins to count down from this value.

Copyright 2007 Cirrus Logic

Smart Panel R/W and RS signals must be implemented via GPIOs and controlled via software. The difference between the CNT[3:0] value and the ESTRT[3:0] value is what guarantees set up time for these GPIO signals to the Smart Panel before the rising edge of the E enable signal on the E pin.

DAT: Data - Read Only

This parallel interface data is input to the EP93xx processor from the Smart Panel during a read cycle (see RD bit in the ParllIfOut register for read cycle). The D[7:0] bits from the Smart Panel are loaded into this DAT field, respectively, on the falling edge of the 'E' enable control signal on the E pin.

Writing PIFEN = '1' to the VideoAttribs register redefines the signals on these pins for Parallel Interface (Smart Panel) operation:

V_CSYNC --> D7 (Smart Panel)

HSYNC --> D6

BLANK --> D5

P17 --> D4

P3 --> D3

P[2:0] --> D[2:0]

SPCLK --> E

Smart Panel R/W and RS signals must be implemented via GPIOs and controlled via software.

Copyright 2007 Cirrus Logic

Blink Control Registers

BlinkRate

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								RATE							

Address: 0x8003_0040

Default: 0x0000_0000

Definition: Blink Rate Control register

Bit Descriptions:

RSVD: Reserved - Unknown during read

RATE: Rate - Read/Write

The blink rate value that is written to this field controls the number of video frames that occur before the LUT addresses assigned to 'blink' change between masked and unmasked (see "Blink Function" on page 7-10). The on/off blink cycle is controlled by this equation:

Blink Cycle = 2 x (1/VCLK) x HClkTotal x VLinesTotal x (255 - BlinkRate)

BlinkMask

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD								MASK							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MASK															

Address: 0x8003_0044

Default: 0x0000_0000

Definition: Blink Mask register

This register is used in conjunction with the BlinkPattrn register to determine which pixels that are fetched from SDRAM are blink pixels.

Copyright 2007 Cirrus Logic

Bit Descriptions:

RSVD: Reserved - Unknown during read

MASK: Mask - Read/Write

The Blink Mask value that is written to this field is logical ANDed, ORed, or XORed with the pixel data that addresses the LUT. The mask allows a blinking pixel to jump from the normal color definition location to a blink color definition location in the look-up-table.

The logical operator is selected by writing to the M field in the PixelMode register. The functions of the BlinkMask AND/OR/XOR operation can be viewed as:

ANDing modifies the LUT address by clearing bits

ORing modifies the LUT address by setting bits

XORing modifies the LUT address by inverting bits

BlinkPattrn

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD								PATRN							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PATRN															

Address: 0x8003_0048

Default: 0x0000_0000

Definition: Blink Pattern register

This register is used in conjunction with the BlinkMask register to determine which pixels that are fetched from SDRAM are blink pixels (see "BlinkPattrn Register" on page 7-33).

Bit Descriptions:

RSVD: Reserved - Unknown during read

PATRN: Pattern - Read/Write

The pixel value is first operated on by the Mask field in the BlinkMask register. The result is then compared to the blink pattern value that is written to this PATRN field. If the comparison results in a match, the pixel is validated as a blink pixel.

Copyright 2007 Cirrus Logic

PattrnMask

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD								PMASK							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PMASK															

Address: 0x8003_004C

Default: 0x0000_0000

Definition: Blink Pattern Mask register

Bit Descriptions:

RSVD: Reserved - Unknown during read

PMASK: Pattern Mask - Read/Write

The Blink Pattern Mask value that is written to this field defines which bits of the PATRN field in the BlinkPattrn register are used to validate a blink pixel:

0 - Bit used for comparison

1 - Bit not used for comparison

BkgrndOffset

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD								BGOFF							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BGOFF															

Address: 0x8003_0050

Default: 0x0000_0000

Definition: Blink Background Color / Blink Offset value register

Copyright 2007 Cirrus Logic

Bit Descriptions:

RSVD: Reserved - Unknown during read

BGOFF: Background Off - Read/Write

The function of Background Off value that is written to this field is defined by the selected blink mode.

When the value of the M field in the PixelMode is written to select 'blink to background' mode, the BGOFF field defines a 24-bit color for the background.

When the value of the M field in the PixelMode is written to select 'blink to offset' mode, the BGOFF field defines the mathematical offset value for the blink color. The format for the mathematical offset is based on the color display mode - that is, 888, 565, 555 (see "Types of Blinking" on page 7-33).

Hardware Cursor Registers

CursorAdrStart

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADR															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADR														NA	

Address: 0x8003_0060

Default: 0x0000_0000

Definition: Cursor Image Address Start register

Bit Descriptions:

ADR: Address - Read/Write

The Cursor Address Start value that is written to this field specifies the SDRAM location that contains the start of the cursor image. The cursor image is 2-bits per pixel, and is stored linearly. The amount of storage space is dependent on the width and height of the cursor.

NA: Not Assigned - Will return the written value

Copyright 2007 Cirrus Logic

CursorAdrReset

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADR															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADR														NA	

Address: 0x8003_0064

Default: 0x0000_0000

Definition: Cursor Image Address Reset register

Bit Descriptions:

ADR: Address - Read/Write

The Cursor Address Reset value that is written to this field specifies the SDRAM location of the part of the cursor that will be displayed next after reaching the last line of the cursor.

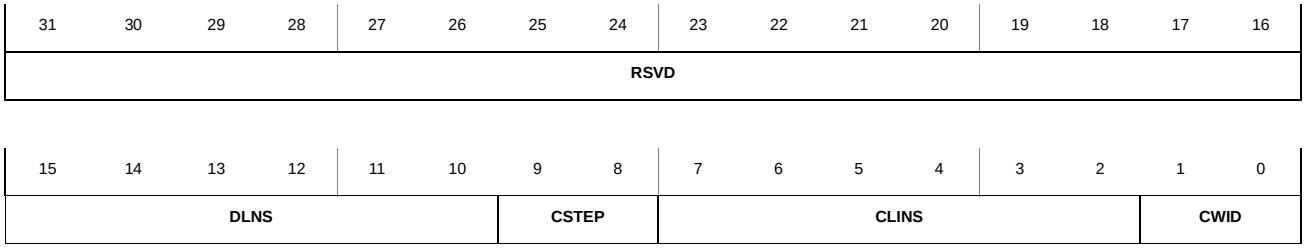
Both start and reset locations are necessary for Dual Scan display of cursor information. If the cursor is totally in the upper half or lower half of the screen, the Start and Reset locations will be the same. Otherwise the cursor will start being overlaid on the video information at the start address, and when the dual scan height counter generates a carry, will jump to the reset value. The cursor will then continue to be overlaid when the Y location is reached, and will jump to the start address value when the height counter for the upper half generates a carry.

Offsetting the reset value and changing the width of the cursor to be different from the cursor step value allows the right 48, 32, or 16 pixels of a larger cursor to be displayed only. Furthermore, offsetting the reset X location off of the left edge of the screen will allow pixel placement of the cursor off of the screen edge.

NA: Not Assigned - Will return the written value

Copyright 2007 Cirrus Logic

CursorSize



Address: 0x8003_0068

Default: 0x0000_0000

Definition: Cursor Height, Width, and Step Size register

Bit Descriptions:

RSVD: Reserved - Unknown during read

DLNS: Dual Scan Lower Half Lines - Read/Write

If DSCAN = '1' in the PixelMode register, the Dual Scan Lower Half Lines value that is written to this field specifies the number of cursor lines that are displayed in the lower half of the display.

CSTEP: Cursor Step Size - Read/Write

The Cursor Step Size value that is written to this field specifies the counter step size for the width of the cursor image:

00 - Step by 1 word or 16 pixels at a time

01 - Step by 2 words or 32 pixels at a time

10 - step by 3 words or 48 pixels at a time

11 - Step by 4 words or 64 pixels at a time

CLINS: Cursor Lines - Read/Write

The Cursor Lines value that is written to this field specifies the height in lines of the cursor image. The value should be set to 'number of cursor lines minus one'.

In dual scan mode this field should be set to the 'number of cursor lines minus one' to be displayed in the top half of the display.

CWID: Cursor Width - Read/Write

The Cursor Width value that is written to this field specifies the 'displayed word width minus one' of the cursor image:

Copyright 2007 Cirrus Logic

00 - Display 1 word (16 pixels)

01 - Display 2 words (32 pixels)

10 - Display 3 words (48 pixels)

11 - Display 4 words (64 pixels)

CursorColor1,

CursorColor2,

CursorBlinkColor1,

CursorBlinkColor2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD								COLOR							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COLOR															

Address: **CursorColor1 - 0x8003_006C**
CursorColor2 - 0x8003_0070
CursorBlinkColor1 - 0x8003_021C
CursorBlinkColor2 - 0x8003_0220

Default: 0x0000_0000

Definition: Cursor Color registers

Bit Descriptions:

RSVD: Reserved - Unknown during read

COLOR: Color - Read/Write

The Color value that is written to this field specifies the cursor image color that is inserted directly into the video pipeline. This color overlays all other colors when the cursor is enabled. This color does not go through the LUT.

The 2-bits-per-pixel cursor image is stored anywhere in SDRAM. When cursor pixels are fetched from SDRAM, they are decoded and displayed as:

00 - Transparent
01 - Invert video stream
10 - CursorColor1 during no blink; CursorBlinkColor1 during blink
11 - CursorColor2 during no blink; CursorBlinkColor2 during blink

Copyright 2007 Cirrus Logic

CursorXYLoc

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				YLOC											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEN	RSVD			XLOC											

Address: 0x8003_0074

Default: 0x0000_0000

Definition: Cursor X and Y Location register

Bit Descriptions:

RSVD: Reserved - Unknown during read

YLOC: Y Location - Read/Write

The Y Location value written to this field specifies the starting vertical Y location of the cursor image. The value is compared to the vertical line counter and it should be specified to be between the active start and active stop vertical line values.

The cursor hardware will clip the cursor at the bottom of the screen. To prevent cursor distortion, a new Y Location value will not be used until the next frame.

CEN: Cursor Enable - Read/Write

Writing a '1' to this bit enables the hardware to insert the defined cursor into the image output video stream. The cursor image fetched from an SDRAM location that is defined by the CursorAdrStart register is combined with the output video stream. Writing a '0' to this bit disables the cursor.

0 - Hardware cursor not enabled

1 - Hardware cursor enabled

When Dual Scan mode is enabled by writing DSCAN = '1' in the PixelMode register, this Cursor Enable bit specifies that some or all of the cursor is located in the upper half of the display.

XLOC: Y Location - Read/Write

Copyright 2007 Cirrus Logic

The X Location value written to this field specifies the starting horizontal X location of the cursor image. The value is compared to the horizontal pixel counter and it should be specified to be between the active start and active stop horizontal pixel values.

This X Location value is also used to specify the starting location for the cursor image in the upper half of the display when Dual Scan mode is enabled by writing DSCAN = '1' in the PixelMode register.

The cursor hardware will clip the cursor at the right edge of the screen. To prevent cursor distortion, a new X Location value will not be used until the next frame.

CursorDScanLHYLoc

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLHEN	RSVD							YLOC							

Address: 0x8003_0078

Default: 0x0000_0000

Definition: Cursor Y Location register

Bit Descriptions:

RSVD: Reserved - Unknown during read

CLHEN: Cursor Lower Half Enable - Read/Write

Writing a '1' to this bit specifies that some or all of the cursor image is located in the lower half of the display. Writing a '0' to this bit specifies the opposite.

0 - Hardware cursor not located in lower half of display

1 - Hardware cursor located in lower half of display

YLOC: Y Location - Read/Write

Copyright 2007 Cirrus Logic

When Dual Scan mode is enabled by writing DSCAN = '1' in the PixelMode register, the Y Location value written to this field specifies the starting vertical Y location (in the lower half of the display) of the cursor image. The value is compared to the vertical line counter and it should be specified to be between the active start and active stop vertical line values.

The cursor hardware will clip the cursor at the bottom of the display. To prevent cursor distortion, a new Y Location value will not be used until the next frame.

CursorBlinkRateCtrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD							EN	RATE							

Address: 0x8003_0224

Default: 0x0000_0000

Definition: Blink Rate Control register

Bit Descriptions:

RSVD: Reserved - Unknown during read

EN: Enable - Read/Write

Writing a '1' to this bit enables hardware cursor blinking and enables the blink rate counter. Writing a '0' to this bit disables hardware cursor blinking and disables the blink rate counter:

0 - Hardware cursor blinking not enabled

1 - Hardware cursor blinking enabled

When EN = '1' and the 2-bit cursor pixel fetched from SDRAM is '10', CursorColor2, is used for the 'on' part of the blink toggle and CursorColor1, is used for the 'off' part of the blink toggle.

When EN = '1' and the 2-bit cursor pixel fetched from SDRAM is '11', CursorBlinkColor1, is used for the 'on' part of the blink toggle and CursorColor1, is used for the 'off' part of the blink toggle.

Copyright 2007 Cirrus Logic

When EN = '0' and the 2-bit cursor pixel fetched from SDRAM is '10', CursorColor1, is used for the non-blinking cursor image.

When EN = '0' and the 2-bit cursor pixel fetched from SDRAM is '11', CursorColor1, is used for the non-blinking cursor image.

RATE: Rate - Read/Write

When EN = '1', the Rate value written to this field specifies the number of video frames that will occur before switching between CursorColor1 or CursorColor2, and CursorBlinkColor1 or CursorBlinkColor2, respectively.

An on/off cursor blink cycle is controlled by the equation:

Blink Cycle = 2 x (1/VCLK) x HClkTotal.Total x VLinesTotal.Total x (255 - RATE)

LUT Registers

GrySclLUTR,

GrySclLUTG,

GrySclLUTB

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD													FRAME	VERT	HORZ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D															

Address: **GrySclLUTR - 0x8003_0080 through 0x8003_00FC**
GrySclLUTG - 0x8003_0280 through 0x8003_02FC
GrySclLUTB - 0x8003_0300 through 0x8003_037C

Default: **0x0000_FFFF in offset locations 0x7, 0x15, 0x23, and 0x31**
0x0000_0000 in all other locations

Definition: Grayscale Look-Up-Tables

Bit Descriptions:

RSVD: Reserved - Unknown during read

FRAME: Frame Counter Selection - Read/Write

Copyright 2007 Cirrus Logic

Writing a Frame Counter Selection value to this bit selects which Frame Counter is used for the current 3-bit pixel value:

0 - use FRAME_CNT3
1 - use FRAME_CNT4

This bit is only defined for address locations GrySclLUTx Base + 0x000 to GrySclLUTx Base + 0x01C.

VERT: Vertical Counter Selection - Read/Write

Writing a Vertical Counter Selection value to this bit selects which Vertical Counter is used for the current 3-bit pixel value:

0 - use FRAME_CNT3
1 - use FRAME_CNT4

This bit is only defined for address locations GrySclLUTx Base + 0x000 to GrySclLUTx Base + 0x01C.

HORZ: Horizontal Counter Selection - Read/Write

Writing a Horizontal Counter Selection value to this bit selects which Horizontal Counter is used for the current 3-bit pixel value:

0 - use FRAME_CNT3
1 - use FRAME_CNT4

This bit is only defined for address locations GrySclLUTx Base + 0x000 to GrySclLUTx Base + 0x01C.

D: Matrix Position Enable - Read/Write

Copyright 2007 Cirrus Logic

Writing '1's to these Matrix Position Enable bits enables the control/dither of the monochrome data outputs according the to horizontal position, the vertical position, the frame, and the 3-bit incoming pixel value. Please reference Table 7-17 below to determine D bit positions in the matrix.

Table 7-17. Grayscale Look-Up-Table (LUT)

Frame Ctr	Vert Ctr	Horz Ctr	VCNT (lines)	11	11	11	11	10	10	10	10	01	01	01	01	00	00	00	00	GSLUT Address *4	
			HCNT (pixels)	11	10	01	00	11	10	01	00	11	10	01	00	11	10	01	00	Frame	Pixel Value
D18	D17	D16		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00	000
D18	D17	D16		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	00	001
D18	D17	D16		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	00	010
D18	D17	D16		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	00	011
D18	D17	D16		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	00	100
D18	D17	D16		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	00	101
D18	D17	D16		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	00	110
D18	D17	D16		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	00	111
X	X	X		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	01	000
X	X	X		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	01	001
X	X	X		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	01	010
X	X	X		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	01	011
X	X	X		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	01	100
X	X	X		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	01	101
X	X	X		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	01	110
X	X	X		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	01	111
X	X	X		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	000
X	X	X		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	10	001
X	X	X		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	10	010
X	X	X		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	10	011
X	X	X		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	10	100
X	X	X		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	10	101
X	X	X		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	10	110
X	X	X		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	10	111
X	X	X		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	000
X	X	X		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	11	001
X	X	X		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	11	010
X	X	X		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	11	011
X	X	X		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	11	100
X	X	X		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	11	101
X	X	X		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	11	110
X	X	X		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	11	111

Copyright 2007 Cirrus Logic

Where:
FRAME[1:0] = FRAME_CNT3 or FRAME_CNT4 as defined by FRAME at address Pixel_In
VCNT[1:0] = VERT_CNT3 or VERT_CNT4 as defined by VERT at address Pixel_In
HCNT[1:0] = HORZ_CNT3 or HORZ_CNT4 as defined by HORZ at address Pixel_In

LUTSwCtrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD														SSTAT	SWTCH

Address: 0x8003_0218

Default: 0x0000_0000

Definition: LUT Switching Control register

Bit Descriptions:

RSVD: Reserved - Unknown during read

SSTAT: Switch Status - Read Only

When SWTCH = '0', Switch Status = '1' means that RAM0 is in the video pipeline and RAM1 is accessible to the bus.

When SWTCH = '1', Switch Status = '1' means that RAM1 is in the video pipeline and RAM0 is accessible to the bus.

During active video, the switch does not occur until the beginning of the next frame. When the video state machine is disabled, the switch occurs almost immediately.

SWTCH: Switch - Read/Write

Writing a Switch value to this bit selects which of these conditions is present when SSTAT = '1':

0 - RAM0 in video pipeline, RAM1 is accessible from bus
1 - RAM1 in video pipeline, RAM0 is accessible from bus.

Copyright 2007 Cirrus Logic

ColorLUT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD								R							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G								B							

Address: 0x8003_0400 through 0x8003_07FC

Default: Unknown after power up

Definition: Color Look-Up-Table

Bit Descriptions:

Note: Triple 8-bit RGB is the most common way to use the LUT. However, The LUT may be organized differently depending on the needs of the display technology.

RSVD: Reserved - Unknown during read

R, G, B: Red, Green, Blue Color - Read/Write

Triple 8-bit Red, Green, and Blue Look-Up-Table (LUT) data is written to and read from these LUT locations. The position in the LUT where the RGB data is read/written is determined by the word address value ADR[9:2]. When the LUT is in the video pipeline, pixel data [23:0] is output from LUT word location ADR[9:2].

Video Signature Registers

VidSigRsltVal

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGVAL															

Address: 0x8003_0200

Default: 0x0000_0000

Definition: Video Output Signature Result Value register

Copyright 2007 Cirrus Logic

Bit Descriptions:

RSVD: Reserved - Unknown during read

SIGVAL: Signature Results Value - Read ONly

The Signature Results Value contained in this field is the 16-bit result of the video output signature calculation. This Signature Results Value is usually updated once per frame based on the SigClrStr location. During grayscale operation, the Signature Results Value is updated once every 12 frames.

VidSigCtrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN	RSVD	SPCLK	BRIGHT	CLKEN	BLANK	HSYNC	VSYNC	PEN							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEN															

Address: 0x8003_0204

Default: 0x0000_0000

Definition: Video Output Signature Control register

Bit Descriptions:

EN: Enable - Read/Write

Writing a '1' to this bit enables the Linear Feedback Shift Register (LFSR).

Writing a '0' to this bit disables the LFSR.

RSVD: Reserved - Unknown during read

SPCLK: Smart Panel/Pixel Clock - Read/Write

Writing a '1' to this bit enables the SPCLK output for calculation in the video signature.

Writing a '0' to this bit disables the SPCLK output for calculation in the video signature.

BRIGHT: Bright - Read/Write

Writing a '1' to this bit enables the Brightness control output for calculation in the video signature.

Writing a '0' to this bit disables the Brightness control output for calculation in the video signature.

Copyright 2007 Cirrus Logic

CLKEN: Clock Enable - Read/Write

Writing a '1' to this bit enables the CLKEN control for calculation in the video signature.

Writing a '0' to this bit disables the CLKEN control for calculation in the video signature.

BLANK: Blank - Read/Write

Writing a '1' to this bit enables the BLANK output for calculation in the video signature.

Writing a '0' to this bit disables the BLANK output for calculation in the video signature.

HSYNC: Horizontal Synchronization - Read/Write

Writing a '1' to this bit enables the HSYNC output for calculation in the video signature.

Writing a '0' to this bit disables the HSYNC output for calculation in the video signature.

VSYNC: Vertical Synchronization - Read/Write

Writing a '1' to this bit enables the VSYNC output for calculation in the video signature.

Writing a '0' to this bit disables the VSYNC output for calculation in the video signature.

PEN: Pixel Bits Enable - Read/Write

Writing '1's to these bits enables respective pixel bits for calculation in the video signature.

Writing '0's to these bits disables respective pixel bits for calculation in the video signature.

VSigStrtStop

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				STOP											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				STRT											

Address: 0x8003_0208

Default: 0x0000_0000

Definition: Vertical Signature Bounds Start/Stop register

Copyright 2007 Cirrus Logic

Bit Descriptions:

RSVD: Reserved - Unknown during read

STOP: Stop - Read/Write

The STOP value is the value of the Vertical down counter at which the VSIGEN signal becomes inactive (stops).This indicates the end of the signature calculation for the Vertical frame. VSIGEN is an internal block signal. The SIG_ENABLE control to the video signature analyzer is enabled by the logical AND of VSIGEN and HSIGEN.

STRT: Start - Read/Write

The STRT value is the value of the Vertical down counter at which the VSIGEN signal becomes active (starts).This indicates the start of the signature calculation for the Vertical frame. VSIGEN is an internal block signal. The SIG_ENABLE control to the video signature analyzer is enabled by the logical AND of VSIGEN and HSIGEN.

HSigStrtStop

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				STOP											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				STRT											

Address: 0x8003_020C

Default: 0x0000_0000

Definition: Horizontal Signature Bounds Start/Stop register

Bit Descriptions:

RSVD: Reserved - Unknown during read

STOP: Stop - Read/Write

The STOP value is the value of the horizontal down counter at which the HSIGEN signal becomes inactive (stops). This indicates the end of the signature calculation for a horizontal line. HSIGEN is an internal block signal. The SIG_ENABLE control to the video signature analyzer is enabled by the logical AND of VSIGEN and HSIGEN.

STRT: Start - Read/Write

Copyright 2007 Cirrus Logic

The STRT value is the value of the horizontal down counter at which the HSIGEN signal becomes active (starts). This indicates the start of the signature calculation for a horizontal line. HSIGEN is an internal block signal. The SIG_ENABLE control to the video signature analyzer is enabled by the logical AND of VSIGEN and HSIGEN.

SigClrStr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				VCLR											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				HCLR											

Address: 0x8003_0210

Default: 0x0000_0000

Definition: Signature Clear and Store Location register

Bit Descriptions:

RSVD: Reserved. Unknown during read.

VCLR: Vertical Clear - Read/Write

The VCLR value is the value of the Vertical down counter at which the VSIGCLR signal is active. This indicates the line for clearing the LFSR and storing the result value for the Vertical frame. VSIGCLR is an internal block signal. The SIG_CLR control to the video signature analyzer is generated by the logical AND of VSIGCLR and HSIGCLR. The SigClrStr control signal is also routed to an edge trigger capable interrupt on the interrupt controller for use as a programmable secondary raster engine interrupt output.

HCLR: Horizontal Clear - Read/Write

The HCLR value is the value of the Vertical down counter at which the HSIGCLR signal is active. This indicates the specific horizontal pixel clock for clearing the LFSR and storing the result value within a horizontal line. HSIGCLR is an internal block signal. The SIG_CLR control to the video signature analyzer is generated by the logical AND of VSIGCLR and HSIGCLR. The SigClrStr control signal is also routed to an edge trigger capable interrupt on the interrupt controller for use as a programmable secondary raster engine interrupt output.

Copyright 2007 Cirrus Logic

Copyright 2007 Cirrus Logic

Chapter 8
Graphics Accelerator 8

8.1 Overview

Note: The chapter applies only to the EP9307 and EP9315 procesors.

The hardware Graphics Accelerator improves graphic performance by handling block copy, block fill, and hardware line draw functions. The Graphics Accelerator is used to off-load graphics functions from the ARM Core. Pixel depths supported by the Graphics Accelerator are 4, 8, 16 or 24 bits per pixel. The 24 bits per pixel mode can be operated as packed (4 pixels every 3 words) or unpacked (1 pixel per word with the high byte unused.) The Block Copy function of the Graphics Accelerator is similar to a DMA (Direct Memory Access) transfer that understands:

1. Pixel organization
2. Block width
3. Transparency, and
4. Transformation from 1 bpp (bit per pixel) to higher 4, 8, 16 or 24 bpp.

The Line Draw functions allow for solid lines or dashed lines. The colors for line drawing can be either foreground color and background color or foreground color with the background being transparent. The Graphics Accelerator also has an interrupt to indicate completion, or termination due to error, of the current function.

8.2 Block Processing Modes

The block transfer modes allow transferring blocks of data from the source to the destination. Block transfers occur between two memory areas that are the same size or from a packed source to unpacked destination. It is not possible to copy from a large source to a smaller destination. Three data path options are provided during block transfers:

1. Transparency
2. Logical AND/OR/XOR Mask, and
3. Logical AND/OR/XOR Destination

Since the block transfer features are all in the data path, transfers may be performed with any combination of the previous functions enabled. When combining functions, the precedence is Mask logic first, destination logical combination second, and finally transparency.

Copyright 2007 Cirrus Logic

8

8.2.1 Copy

It is possible to copy data from the source memory to the destination memory using the copy form of block transfer. A copy is accomplished by not enabling any of the data path options, i.e. Transparency, Logical Mask, or Destination Mask.

8.2.1.1 Transparency

Transparency is used to preserve pixels in the destination memory. When a pixel in the source block is defined as a transparent pixel, the corresponding destination pixel will be left un-modified by the block transfer. A pixel is defined as transparent when it matches the pixel value that is loaded into the TRANSPATTRN register. Comparisons are made based on the bits per pixel mode of the transfer.

8.2.1.2 Logical Mask

Logical Mask is used to manipulate the pixel data as it is copied from the source location to the destination. The source data will not be modified unless the source is also the destination. All pixel data is manipulated based on the value of the "BLOCKMASK" register and the desired operation. The operations, using "C" syntax, of Logical Mask are:

AND - This operator is used to remove pattern attributes from a pixel.

Dest = BLOCKMASK & Src;

OR - This operator may be used to add regular pattern attributes to a pixel.

Dest = BLOCKMASK | Src;

XOR - This operator is used for pixel bit plane inversion.

Dest = BLOCKMASK ^ Src; (where ^ is an XOR operation)

8.2.1.3 Logical Destination

Logical Destination provides for the modification of the destination data based on the value of the source pixels. The operations, using "C" syntax, of Logical Destination are:

AND - This operator is used to remove pattern attributes from a pixel.

Dest = Dest & Src;

OR - This operator is used to add regular pattern attributes to a pixel.

Dest = Dest | Src;

XOR - This operator is used for pixel bit plane inversion.

Dest = Dest ^ Src; (where ^ is an XOR operation)

8.2.1.4 Operation Precedence

The order of precedence is:

1. Logical Mask

Copyright 2007 Cirrus Logic

2. Logical Destination
3. Transparency

8.2.2 Remapping

The Graphics Accelerator supports single bit pixel remapping with foreground/background or foreground/transparency to system color depth images (1 bpp mapped to 4, 8, 16 or 24 bpp expansion.) Images stored as a single bit plane can be expanded with a foreground color and either transparent or background color. Remapping can be used for fast transfer of text, single color patterns, and single color bit maps to video memory.

Note: The Graphics Accelerator only supports movement in the positive direction for X and Y. In other words, use the remapping function only from the display top to bottom and from the display left to right.

8.2.3 Block Fills

The Graphics Accelerator supports pixel addressed Block Fills and Block Copies with 4, 8, 16, or 24 bpp resolution. During Block Fills, rectangular blocks of pixels are replaced with the pixel value that is in the "BLOCKMASK" register. For unpacked 24 bpp fills, the high byte is set to 0x00.

8.2.4 Packed Memory Transfer

A packed source means that all bits in a word are used for the source image. The only exception is the last word, which is not required to be used if the image size does not require the storage. A packed source DOES NOT mean that all words are packed together. The lines may have none, or one or more word(s) between each line. A line is defined as a continuous block of words that contains pixel data.

In packed mode, the source can have a different layout than the destination. This is different from non-packed mode where the "BLKSRCWIDTH" and "BLKDESTHEIGHT" are the same.

To enable a Packed Source transfer set the PACKD bit in the "BLOCKCTRL" register.

8.3 Line Draws

The Graphics Engine supports two types of hardware accelerated lines draws:

1. Breshenham Line Draw, or
2. Pixel Step Line Draw

The only programming difference between the two line draw algorithms is how the line increment registers are set. The lines may be drawn using solid lines or patterned lines. Accelerated line draw makes it possible to draw a single pixel width line between any two points with sub pixel accuracy.

Copyright 2007 Cirrus Logic

8

8.3.1 Breshenham Line Draws

Based on Breshenham's algorithm, this is the fastest of the two lines draws. Patterned lines drawn are aligned to the major axis. Steps made in the major axis are made on a 4095/4096 pixel step per clock basis. This allows the algorithm to complete the line with the amount of pixel draws in the major axis. Steps in the minor axis are made in sub pixel increments.

Patterned lines drawn in this mode are aligned to the major axis. A pattern up to 16 bits long repeats on an interval up to 16 bits. This type of patterning is commonly used.

8.3.2 Pixel Step Line Draws

This is a sub-pixel accumulation line draw that will typically take longer to draw than a Breshenham line draw. The major advantage of the pixel step line draw is that it provides angularly corrected patterns. This means that the pattern of the line is applied along the line at the appropriate angle. The number of algorithm iterations is calculated based on the calculated pixel length of the line (Pythagorean theorem). A pattern up to 16 bits long repeats on an interval up to 16 bits. In this mode, visual correctness is emphasized over completeness. For higher definition patterns, details of the pattern may be lost.

Wide lines are not hardware accelerated, but may be generated by stepping and repeating single pixel width lines.

If speed is critical, horizontal un-patterned lines may be drawn by single pixel deep block fills.

Note: Line drawing in the negative X or Y directions is not supported by the hardware.

8.4 Memory Organization for Graphics Accelerator

Table 8-1 shows a hypothetical 8 x 6 pixel matrix as it would appear on a display.

P(x,y) is defined as a pixel at location x,y from the upper left corner of the screen.

Table 8-1. Screen Pixels

	X-Axis →							
Y-Axis ↓	P(0,0)	P(1,0)	P(2,0)	P(3,0)	P(4,0)	P(5,0)	P(6,0)	P(7,0)
	P(0,1)	P(1,1)	P(2,1)	P(3,1)	P(4,1)	P(5,1)	P(6,1)	P(7,1)
	P(0,2)	P(1,2)	P(2,2)	P(3,2)	P(4,2)	P(5,2)	P(6,2)	P(7,2)
	P(0,3)	P(1,3)	P(2,3)	P(3,3)	P(4,3)	P(5,3)	P(6,3)	P(7,3)
	P(0,4)	P(1,4)	P(2,4)	P(3,4)	P(4,4)	P(5,4)	P(6,4)	P(7,4)
	P(0,5)	P(1,5)	P(2,5)	P(3,5)	P(4,5)	P(5,5)	P(6,5)	P(7,5)

Copyright 2007 Cirrus Logic

8

8.4.1 Memory Organization for 1 Bit Per Pixel (bpp)

The 1 bpp storage format is for storing compressed image data for remapping only. This data cannot be displayed until it is remapped into a supported color depth. Table 8-2 shows how compressed 1 bpp images are stored in memory as 8 pixels per byte.

Table 8-2. bpp Memory Organization

	31 24	23 16	15 8	7 0
0x0000	P(7,3)..... .....P(0,3)	P(7,2)..... .....P(0,2)	P(7,1)..... .....P(0,1)	P(7,0)..... .....P(0,0)
0x0004	X X	X X	P(7,5)..... .....P(0,5)	P(7,4)..... .....P(0,4)

8.4.2 Memory Organization for 4-Bits Per Pixel

The 4 bpp storage format can be used to support monochrome, 8 levels of grayscale, and 8 or 16 color displays. The actual frame buffer can be organized as 2 pixels per byte or 1 pixel per byte. The Graphics Accelerator engine treats 4 bpp with 1 pixel per byte as 8 bpp mode. Table 8-3 shows how 4 bpp images are stored in memory as 2 pixels per byte.

Table 8-3. 4 bpp Memory Organization

	31 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0
0x0000	P(6,0)	P(7,0)	P(4,0)	P(5,0)	P(2,0)	P(3,0)	P(0,0)	P(1,0)
0x0004	P(6,1)	P(7,1)	P(4,1)	P(5,1)	P(2,1)	P(3,1)	P(0,1)	P(1,1)
0x0008	P(6,2)	P(7,2)	P(4,2)	P(5,2)	P(2,2)	P(3,2)	P(0,2)	P(1,2)
0x0000	P(6,3)	P(7,3)	P(4,3)	P(5,3)	P(2,3)	P(3,3)	P(0,3)	P(1,3)
0x0004	P(6,4)	P(7,4)	P(4,4)	P(5,4)	P(2,4)	P(3,4)	P(0,4)	P(1,4)
0x0008	P(6,5)	P(7,5)	P(4,5)	P(5,5)	P(2,5)	P(3,5)	P(0,5)	P(1,5)

8.4.3 Memory Organization for 8-Bits Per Pixel

The 8 bpp storage format can be used to support 8 level grayscale and color displays. For color displays, this mode would use a software changeable palette in the Raster Engine to map 256 color selections to 24-bit colors. Table 8-4 shows how 8 bpp images are stored in memory as 1 pixel per byte.

Copyright 2007 Cirrus Logic

8

Table 8-4. 8 bpp Memory Organization

	31 24	23 16	15 8	7 0
0x0000	P(3,0)	P(2,0)	P(1,0)	P(0,0)
0x0004	P(7,0)	P(6,0)	P(5,0)	P(4,0)
0x0008	P(3,1)	P(2,1)	P(1,1)	P(0,1)
0x000C	P(7,1)	P(6,1)	P(5,1)	P(4,1)
0x0010	P(3,2)	P(2,2)	P(1,2)	P(0,2)
0x0014	P(7,2)	P(6,2)	P(5,2)	P(4,2)
.....	.....	.....	.....	.....
0x0028	P(3,5)	P(2,5)	P(1,5)	P(0,5)
0x002C	P(7,5)	P(6,5)	P(5,5)	P(4,5)

8.4.4 Memory Organization for 16-Bits Per Pixel

The 16 bpp storage format can be used to support high color displays. This mode would typically be used to implement a 5-bit blue, 6-bit green, 5-bit red color scheme or a 5-bit blue, 5-bit green, 5-bit red color scheme. The least significant byte in 16 bpp mode could also be used in conjunction with the Raster Engine palette to map 256 color selections to 24 bit colors. With 256 color mapping, the most significant byte for each pixel would not be used for color information. Table 8-5 shows how 16 bpp images are stored in memory as 1 pixel for every two bytes.

Table 8-5. 16 bpp Memory Organization

	31 16	15 0
0x0000	P(1,0)	P(0,0)
0x0004	P(3,0)	P(2,0)
0x0008	P(5,0)	P(4,0)
0x000C	P(7,0)	P(6,0)
0x0010	P(1,1)	P(0,1)
0x0014	P(3,1)	P(2,1)
0x0018	P(5,1)	P(4,1)
0x001C	P(7,1)	P(6,1)
.....	.....	.....
0x0050	P(1,5)	P(0,5)
0x0054	P(3,5)	P(2,5)
0x0058	P(5,5)	P(4,5)
0x005C	P(7,5)	P(6,5)

Copyright 2007 Cirrus Logic

8.4.5 Memory Organization for 24-Bits Per Pixel

The 24 bpp packed or unpacked storage formats can be used to support higher color displays. These modes would typically be used to implement an 8-bit blue, 8-bit green, 8-bit red color scheme. Table 8-6 shows how 24 bpp packed images are stored in memory as 1 pixel for every three bytes. Table 8-7 shows how 24 bpp unpacked images are stored in memory

Table 8-6. 24 bpp Packed Memory Organization (4 pixel/ 3 words)

	31 24	23 16	15 8	7 0
0x0000	P(1,0)B	P(0,0)R	P(0,0)G	P(0,0)B
0x0004	P(2,0)G	P(2,0)B	P(1,0)R	P(1,0)G
0x0008	P(3,0)R	P(3,0)G	P(3,0)B	P(2,0)R
0x000C	P(5,0)B	P(4,0)R	P(4,0)G	P(4,0)B
0x0010	P(6,0)G	P(6,0)B	P(5,0)R	P(5,0)G
0x0014	P(7,0)R	P(7,0)G	P(7,0)B	P(6,0)R
.....	.....	.....	.....	.....
0x0078	P(1,5)B	P(0,5)R	P(0,5)G	P(0,5)B
0x007C	P(2,5)G	P(2,5)B	P(1,5)R	P(1,5)G
0x0080	P(3,5)R	P(3,5)G	P(3,5)B	P(2,5)R
0x0084	P(5,5)B	P(4,5)R	P(4,5)G	P(4,5)B
0x0088	P(6,5)G	P(6,5)B	P(5,5)R	P(5,5)G
0x008C	P(7,5)R	P(7,5)G	P(7,5)B	P(6,5)R

Table 8-7. 24 bpp Unpacked Memory Organization (1 pixel/ 1 word)

	31 24	23 16	15 8	7 0
0x0000	unused	P(0,0)R	P(0,0)G	P(0,0)B
0x0004	unused	P(1,0)R	P(1,0)G	P(1,0)B
0x0008	unused	P(2,0)R	P(2,0)G	P(2,0)B
0x000C	unused	P(3,0)R	P(3,0)G	P(3,0)B
0x0010	unused	P(4,0)R	P(4,0)G	P(4,0)B
0x0014	unused	P(5,0)R	P(5,0)G	P(5,0)B
0x0018	unused	P(6,0)R	P(6,0)G	P(6,0)B
0x001C	unused	P(7,0)R	P(7,0)G	P(7,0)B
0x0020	unused	P(0,1)R	P(0,1)G	P(0,1)B
....	.....	.....	.....	.....
0x00A0	unused	P(0,5)R	P(0,5)G	P(0,5)B
0x00A4	unused	P(1,5)R	P(1,5)G	P(1,5)B
0x00A8	unused	P(2,5)R	P(2,5)G	P(2,5)B
0x00AC	unused	P(3,5)R	P(3,5)G	P(3,5)B

Copyright 2007 Cirrus Logic

8

Table 8-7. 24 bpp Unpacked Memory Organization (1 pixel/ 1 word)

0x00B0	unused	P(4,5)R	P(4,5)G	P(4,5)B
0x00B4	unused	P(5,5)R	P(5,5)G	P(5,5)B
0x00B8	unused	P(6,5)R	P(6,5)G	P(6,5)B
0x00BC	unused	P(7,5)R	P(7,5)G	P(7,5)B

8.4.6 Memory Map Access

The Graphics Accelerator has access to the entire memory map. Therefore pixel block function processing is not limited to graphics and video memory. Font storage, bit map storage, etc. can be stored anywhere in the memory map. To alleviate page miss penalties for copies between SDRAM memory pages, the Graphics Accelerator uses a 32-entry copy buffer during block transfers.

8.5 Register Programming

Some of the registers used to operate the Graphics Accelerator need extra explanation for proper usage. There are two sets such registers. They specify Word Count and Pixel End/Start values.

8.5.1 Word Count

The "BLKSRCWIDTH" and "BLKDESTWIDTH" registers must be written with the 'number of 32-bit words minus 1' that are to be fetched from the SDRAM buffer. If any pixel bit is in a word. it must be counted as a full word.

8.5.1.1 Example: 8 BPP mode

If a Block Copy starts at pixel 0 and 7 pixels are to be copied, the "BLKSRCWIDTH" register would be loaded with a 0x1 (2 words - 1 word = 0x1) since the 7th pixel resides in word 1 and the 0th pixel resides in word 0. The pixels fetched are highlighted in Table 8-8.

Table 8-8. Transfer Example 1

Address	31			0	31			0	31			0	31			0
0x0000 - 0x000C	FF	EE	DD	CC	BB	AA	99	88	77	66	55	44	33	22	11	00

If a Block Copy starts at pixel 0 and 2 pixels are to be copied, the "BLKSRCWIDTH" register would be loaded with 0x0 (1 word - 1 word = 0x0). The pixels fetched are highlighted in Table 8-9.

Copyright 2007 Cirrus Logic

8

Table 8-9. Transfer Example 2

Address	31				0	31				0	31				0	31				0
0x0000 - 0x000C	FF	EE	DD	CC		BB	AA	99	88		77	66	55	44		33	22	11	00	

If a Block Copy starts at pixel 3 and 10 pixels are to be copied, the "BLKSRCWIDTH" register would be loaded with 0x3 (4 words - 1 word = 0x3). The pixels fetched are highlighted in Table 8-10.

Table 8-10. Transfer Example 3

Address	31				0	31				0	31				0	31				0
0x0000 - 0x000C	FF	EE	DD	CC		BB	AA	99	88		77	66	55	44		33	22	11	00	

8.5.1.2 Example: 24 BPP (packed) mode

If a Block Copy starts at pixel 0 and copies 5 pixels, the "BLKSRCWIDTH" register would be filled with 0x3. This is because the first four pixels consume 3 words and the 5th pixel consumes part of 1 word. This is a total of 4 words. So, the word width is 4 words - 1 word = 0x3. The pixels fetched are highlighted in Table 8-11.

Table 8-11. Transfer Example 4

Address	31				0	31				0	31				0	31				0
0x0000 - 0x000C	55	44	44	44		33	33	33	22		22	22	11	11		11	00	00	00	

If a Block Copy starts at pixel 2 and copies 6 pixels, the "BLKSRCWIDTH" register would be filled with 0x4. This is because the 1st pixel consumes part of the 1st word and the 4 remaining pixels consume the next 4 words. So, the word width is 5 words - 1 word = 0x4. The pixels fetched are highlighted in Table 8-12.

Table 8-12. Transfer Example 5

Address	31				0	31				0	31				0	31				0
0x0000 - 0x000C	55	44	44	44		33	33	33	22		22	22	11	11		11	00	00	00	
0x0010 - 0x001C	AA	AA	99	99		99	88	88	88		77	77	77	66		66	66	55	55	

8.5.2 Pixel End and Start

Two registers are used to control where in a word the first and last pixels reside. This is required since in all color depths more than 1 pixel can reside in a word of memory. This fact requires that the programmer provide the hardware with the exact information of where in a 32-bit word a pixel starts or ends. One register, "SRCPIXELSTRT", is used for the source

Copyright 2007 Cirrus Logic

8

memory and the other register, "DESTPIXELSTRT", is used for the destination memory. All start and stop values described below apply for source and destination values.

The two registers operate in an identical fashion for source and destination. To see how they operate requires looking at several tables that show the memory layout for pixels in the various color modes.

8.5.2.1 4 BPP Word Layout

This 4 BPP mode example is somewhat difficult because the pixels are not in sequential order. For a Block Copy where 8 pixels are transferred per scan line, let the starting SDRAM address of the source image be 0x0000. Table 8-13 shows that Pixel 0 starts at bit 4, Pixel 1 starts at bit 0, etc. The start pixel, P0, is in the word at address 0x0000 and has a beginning bit position of 4. This makes 4 = 0x4 the value that is used for the SPEL field in the "SRCPIXELSTRT" register.

Table 8-13. 4 BPP Memory Layout for Source Image

Address	31 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0
0x0000	P6	P7	P4	P5	P2	P3	P0	P1

Let the starting SDRAM address of the destination image be 0x0020. Table 8-14 shows that Pixel 0 starts at bit 20, Pixel 1 starts at bit 16, etc. The start pixel, P0, is in the word at address 0x0020 and has a beginning bit position of 20. This makes 20 = 0x14 the value that is used for the SPEL field in the "DESTPIXELSTRT" register.

The end pixel, P7, is in the word at address 0x0024 and has a beginning bit position of 8. This makes 8 = 0x8 the value that is used for the EPEL field in the "DESTPIXELSTRT" register.

Note: The word count for this example would be: 2 - 1 = 1 words, since P7 ends in the 2nd word. So, WIDTH = 0x1 would be written to the "BLKDESTWIDTH" register.

Table 8-14. 4 BPP Memory Layout for Destination Image

Address	31 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0
0x0020	P2	P3	P0	P1				
0x0024					P6	P7	P4	P5

Copyright 2007 Cirrus Logic

8

8.5.2.2 8 BPP Word Layout

For a Block Copy where 4 pixels are transferred per scan line, let the starting SDRAM address of the source image be 0x0000. Table 8-15 shows that Pixel 2 starts at bit 16, Pixel 3 starts at bit 24, etc. The start pixel, P2, is in the word at address 0x0000 and has a beginning bit position of 16. This makes 16 = 0x10 the value that is used for the SPEL field in the "SRCPIXELSTRT" register.

Table 8-15. 8 BPP Memory Layout for Source Image

Address	31 24	23 16	15 8	7 0
0x0000	P3	P2	P1	P0
0x0004	P7	P6	P5	P4

Let the starting SDRAM address of the destination image be 0x0030. Table 8-16 shows that Pixel 2 starts at bit 16, Pixel 3 starts at bit 34, etc. The start pixel, P2, is in the word at address 0x0030 and has a beginning bit position of 16. This makes 16 = 0x10 the value that is used for the SPEL field in the "DESTPIXELSTRT" register.

The end pixel, P5, is in the word at address 0x0034 and has a beginning bit position of 8. This makes 8 = 0x8 the value that is used for the EPEL field in the "DESTPIXELSTRT" register.

Note: The word count for this example would be: 2 - 1 = 1 words, since P5 ends in the 2nd word. So, WIDTH = 0x1 would be written to the "BLKDESTWIDTH" register.

Table 8-16. 8 BPP Memory Layout for Destination Image

Address	31 24	23 16	15 8	7 0
0x0030	P3	P2	P1	P0
0x0034	P7	P6	P5	P4

8.5.2.3 16 BPP WORD Layout

For a Block Copy where 8 pixels are transferred per scan line, let the starting SDRAM address of the source image be 0x0000. Table 8-17 shows that Pixel 0 starts at bit 0, Pixel 1 starts at bit 16, etc. The start pixel, P0, is in the word at address 0x0000 and has a beginning bit position of 0. This makes 0 = 0x0 the value that is used for the SPEL field in the "SRCPIXELSTRT" register.

Table 8-17. 16 BPP Memory Layout for Source Image

Address	31 16	15 0
0x0000	P1	P0
0x0004	P3	P2
0x0008	P5	P4
0x000C	P7	P6

Copyright 2007 Cirrus Logic

8

Let the starting SDRAM address of the destination image be 0x0044. Table 8-18 shows that Pixel 0 starts at bit 16. The start pixel, P0, is in the word at address 0x0044 and has a beginning bit position of 16. This makes 16 = 0x10 the value that is used for the SPEL field in the "DESTPIXELSTRT" register.

The end pixel, P7, is in the word at address 0x0054 and has a beginning bit position of 0. This makes 0 = 0x0 the value that is used for the EPEL field in the "DESTPIXELSTRT" register.

Note: The word count for this example would be: 5 - 1 = 4 words, since P7 ends in the 5th word. So, WIDTH = 0x4 would be written to the "BLKDESTWIDTH" register.

Table 8-18. 16 BPP Memory Layout for Destination Image

Address	31 16	15 0
0x0044	P0	
0x0048	P2	P1
0x004C	P4	P3
0x0050	P6	P5
0x0054		P7

8.5.2.4 24 BPP mode

This 24 BPP mode is the most difficult because, unlike the other modes, pixels will span words. For a Block Copy where 6 pixels are transferred per scan line, let the starting SDRAM address of the source image be 0x0000. Table 8-19 shows that Pixel 1 starts at bit 24. The start pixel, P1, is in the word at address 0x0000 and has a beginning bit position of 24. This makes 24 = 0x18 the value that is used for the SPEL field in the "SRCPIXELSTRT" register.

Table 8-19. 24 BPP Memory Layout for Source Image

Address	31 24	23 16	15 8	7 0
0x0000	P1	P0	P0	P0
0x0004	P2	P2	P1	P1
0x0008	P3	P3	P3	P2
0x000C	P5	P4	P4	P4
0x0010	P6	P6	P5	P5
0x0014	P7	P7	P7	P6

Let the starting SDRAM address of the destination image be 0x0058. Table 8-20 shows that Pixel 1 starts at bit 24. The start pixel, P1, is in the word at address 0x0058 and has a beginning bit position of 24. This makes 24 = 0x18 the value that is used for the SPEL field in the "DESTPIXELSTRT" register.

The end pixel, P6, is in the word at address 0x006C and has a beginning bit position of 0. This makes 0 = 0x0 the value that is used for the EPEL field in the "DESTPIXELSTRT" register.

Copyright 2007 Cirrus Logic

Note: The word count for this example would be: 6 - 1 = 5 words, since P6 ends in the 6th word. The word count takes into account the whole pixel, not just the starting location. So, WIDTH = 0x5 would be written to the "BLKDESTWIDTH" register.

Table 8-20. 24 BPP Memory Layout for Destination Image

Address	31 24	23 16	15 8	7 0
0x0058	P1	P0	P0	P0
0x005C	P2	P2	P1	P1
0x0060	P3	P3	P3	P2
0x0064	P5	P4	P4	P4
0x0068	P6	P6	P5	P5
0x006C	P7	P7	P7	P6

8.6 Register Usage

Since some registers have different meanings based on the type of transfer being performed, the next section will give the use and meaning of the register during the various graphics transfers.

8.6.1 Breshenham's Algorithm Line Draw

The following sequence describes how to set up the registers that are used for a Breshenham algorithm line draw:

1. **Setup LINEINIT Register**

   Write YINIT = 0x800 (2048) and XINIT = 0x800 in the "LINEINIT" register.

2. **Setup LINEPATTERN Register**

   A. Write desired values to the Pattern (PTRN) and Count (CNT) fields to create solid or patterned lines. The "LINEPATTRN" register contains a 4-bit pattern Count (CNT) value and a 16-bit Pattern (PTRN) that defines 16 pixel on/off patterns for line functions. CNT specifies the position of the last bit used in the PTRN field starting at bit 0 of the PTRN field.

   B. For a solid line, write CNT = 0xF and PTRN = 0xFFFF to the "LINEPATTRN" register. The solid line will have the color value that is written to the MASK field in the "BLOCKMASK" register.

   C. For a pattern of 8 'on' pixels and 8 'off' pixels, write CNT = 0xF and PTRN = 0x00FF to the "LINEPATTRN" register. The 8 'on' pixels would have the color value that is written to the MASK field in the "BLOCKMASK" register. The 8 'off' pixels would either be transparent as specified by BG = '0' in the "BLOCKCTRL" register or have the color value written to the "BACKGROUND" register as specified by BG = '1' in the "BLOCKCTRL" register. Using DX/DY line draw, the pattern will be more consistent for any line regardless of angle.

Copyright 2007 Cirrus Logic

3. **Setup DESTLINELENGTH Register**

   A. Determine how many pixels occupy a 32-bit word. For example, four 8-bit pixels can occupy a 32-bit word.

   B. Determine the width of the display in pixels. For example, a 640x480 display has a width of 640 pixels.

   C. The line length is determined by the 'stride' of the display, that is, how many 32-bit words are needed to populate the width of the display with pixels. From steps A and B, the stride for this example is 640 pixels divided by 4, where 4 is the number of 8-bit pixels that occupy a word. So, for this example, destination line length is 640 divided by 4 = 160 = 0xA0.

   D. Write 0x0000_00A0 to the "DESTLINELENGTH" register.

4. **Setup DESTPIXELSTRT Register**

   Write desired values to the EPEL and SPEL fields in the "DESTPIXELSTRT" register. See "Pixel End and Start" on page 8-9 for details.

5. **Setup BLKDESTSTRT Register**

   Write the SDRAM address for the starting pixel of the 1st line to the ADR field in the "BLKDESTSTRT" register.

6. **Setup BACKGROUND Register**

   If BG = '1' in the "BLOCKCTRL" register, write the desired background color value to the BG field in the "BACKGROUND" register; if BG = '0' in the "BLOCKCTRL" register, the color value written to the "BACKGROUND" register is ignored. The 'off' pattern bits, if any, will be displayed using the background color.

7. **Setup BLOCKMASK Register**

   Write the desired foreground color value to the MASK field in the "BLOCKMASK" register.

8. **Setup LINEINC Register**

   Write the values determined below to the YINC and XINC fields in the "LINEINC" register:

```
if abs(x2 - x1) == abs(y2 -y1)
Write YINC = 0xFFF (4095)
Write XINC = 0xFFF (4095)
if abs(x2 -x1) < abs(y2 - y1)
Write YINC = 0xFFF (4095)
Write XINC = (abs(x2 - x1) / abs(y2 -y1)) * 4095). Round up to the nearest
whole integer value.
if abs(x2 -x1) > abs(y2 - y1)
Write YINC = (abs(y2 - y1) / abs(x2 - x1)) * 4095). Round to the nearest whole
integer value.
Write XINC = 0xFFF (4095)
```


Copyright 2007 Cirrus Logic

9. Setup BLKDESTWIDTH Register

Write 'abs(X2 -X1) modulo 4096, minus 1' to the WIDTH field in the "BLKDESTWIDTH" register.

10. Setup BLKDESTHEIGHT Register

Write 'abs(Y2 - Y1) / 4096, minus 1' to the HEIGHT field in the "BLKDESTHEIGHT" register.

11. Setup BLOCKCTRL Register

A. Clear the "BLOCKCTRL" register by writing 0x0000_0000 to it.

B. Set the LINE bit to '1'

C. If X2 > X1, set the DXDIR bit to '1', else set the DXDIR bit to '0'

D. If Y2 > Y1, set the DYDIR bit to '1', else set the DYDIR bit to '0'

E. Either set the BG bit to '1' to use the background color specified in "BACKGROUND" register or set the BG bit to '0' for transparent background.

F. Set the P bits to the value for the desired BPP color depth

G. If interrupts are desired, set the INTEN bit to '1'

H. Set the EN bit to '1'

The final step is to wait for an interrupt or poll for EN = '0' in the BLOCKCTRL register. When the EN bit becomes cleared to '0', the line draw function is complete.

8.6.2 Example of Breshenham's Algorithm Line Draw

To achieve the following display and pattern, follow Steps 1 to 14 in this section.

- Display size is 640 x 480 x 16-bits per pixel
- Display memory starts at physical location 0x0000_0000
- Pattern is 8 transparent pixels and 8 white pixels
- X2 = 20, X1 = 101
- Y2 = 20, Y1 = 301

The following sequence describes how to set up those registers that are used for a Breshenham's algorithm line draw.

1. Write XINIT = 0x800 (2048) and YINIT = 0x800 to the "LINEINIT" register
2. Write PTTN = 0x00FF and CNT = 0xF to the "LINEPATTRN" register
3. Write LEN = 0x140 to the "DESTLINELENGTH" register, where LEN = 640 (pixels) x 1/2 (1 / # of 16-bit pixels in word) = 640 x 1/2 = 320 = 0x140
4. Write SPEL = 0x8 and EPEL = 0x0 to the "DESTPIXELSTRT" register, where:

Copyright 2007 Cirrus Logic

SPEL = [X2% 2 (pixel depth / 8-bit byte)] x 8 = [101% 2 (16-bits / 8-bits)] x 8-bits = [101% 2] x 8 = 1 x 8 = 8 = 0x8, and

EPEL = [X1% 2 (pixel depth / 8-bit byte)] x 8 = [20% 2 (16-bits / 8-bits)] x 8-bits = [20% 2] x 8 = 0 x 0 = 0 = 0x0

5. Write the word-aligned value of the SDRAM address 'for the beginning of the line draw' to the "BLKDESTSTRT" register.
6. Write the desired background color value to the BG field in the "BACKGROUND" register. The 'off' pattern bits of the line will be displayed using the background color.
7. Write the desired foreground color value to the MASK field in the "BLOCKMASK" register. The 'on' pattern bits of the line will be displayed using the foreground color.
8. Write YINC = 0xFFF and XINC = 0x49C to the "LINEINC" register, where

   YINC = 4095 = 0xFFF

   XINC = [abs(X2 - X1) / abs(Y2 - Y1)] x 4095 = [abs(20 - 101) / abs(20-301)] x 4095 = (81 / 281) x 4095 = 1180.409, which rounds to 1180 = 0x49C
9. Write WIDTH = 0x50 to the "BLKDESTWIDTH":register, where

   WIDTH = abs(X2 - X1)% 4096 - 1 = abs(20 - 101)% 4096 - 1 = 81% 4096 - 1 = 81 - 1 = 80 = 0x50
10. Write HEIGHT = 0x0 to the "BLKDESTHEIGHT" register, where

    HEIGHT = [abs(Y2 - Y1) - 1] / 4096 = [abs(20 - 301) - 1]/ 4096 = (281 - 1) / 4096 = 0.0686 = 0x0
11. Clear the "BLOCKCTRL" register by writing 0x0000_0000 to it
12. Write Line = '1', DXDIR = '0', DYDIR = '0', BG = '0', P = 0x4, and INTEN = '1' to the "BLOCKCTRL" register
13. Write EN = '1' to the "BLOCKCTRL" register
14. Wait for an interrupt or poll for EN = '0' in the "BLOCKCTRL" register. When the EN bit becomes cleared to '0', the Breshenham's Algorithm line draw function is complete.

8.6.3 Block Fill Function

The following sequence describes how to carry out a Block Fill function:

1. Setup BLOCKMASK Register

Write the desired pixel-fill value to the MASK field in the "BLOCKMASK" register. The pixel-fill value is dependant on the color depth.

2. Setup DESTPIXELSTRT Register

Write the desired values to the SPEL field and the EPEL field in the "DESTPIXELSTRT" register.

Copyright 2007 Cirrus Logic

SPEL is the starting pixel position within the word that the pixel-fill will begin with. EPEL is the ending pixel position within the word that the pixel-fill will end with. See Section 8.5.2. Pixel End And Start. Use the DESTPIXELSTRT calculation in the block copy example shown in Section 8.6.4.1.

3. Setup DESTLINELENGTH Register

Write the line length value to the LEN field in the "DESTLINELENGTH" register, where LEN is determined by:

A. Find how many pixels occupy a 32-bit word. For example, four 8-bit pixels can occupy a 32-bit word.

B. Find the width of the display in pixels. For example, a 640x480 display has a width of 640 pixels.

C. The line length, LEN, is determined by the stride of the display, that is, how many 32-bit words are needed to populate the width of the display with pixels. From steps 1 and 2, the stride for this example is 640 pixels divided by 4, where 4 is the number of 8-bit pixels that occupy a word. So, for this example, line length is 640 divided by 4 = 160 = 0xA0.

Usually the same LEN value is used in both the "DESTLINELENGTH" register and the "SRCLINELENGTH" register.

4. Setup BLKDESTWIDTH Register

Write the value of 'Stride minus 1' to the WIDTH field in the "BLKDESTWIDTH" register, where WIDTH is determined by:

A. Find how many pixels occupy a 32-bit word. For example, four 8-bit pixels can occupy a 32-bit word.

B. Find the width of the image in pixels. For example, a 20 x 10 image has a width of 20 pixels.

C. The stride of the image is how many 32-bit words are needed to populate the width of the image with pixels. From steps 1 and 2, the stride for this example is 20 pixels divided by 4, where 4 is the number of 8-bit pixels that occupy a word. So, the stride is 20 divided by 4 = 5. However, the value of WIDTH is defined as the value of stride less 1. So, WIDTH = 5 - 1 = 4 = 0x004.

5. Setup BLKDESTHEIGHT Register

Write the desired value to the HEIGHT field in the "BLKDESTHEIGHT" register, where HEIGHT = the height in lines of the fill area minus 1.

For example, a 20-pixels x 10-lines image has a height of 10 lines. So, HEIGHT = 10 - 1 = 9 = 0x9.

6. Setup BLKDESTSTRT Register

Write the word-aligned value of the SDRAM address 'for the beginning of the block fill' to the "BLKDESTSTRT" register.

Copyright 2007 Cirrus Logic

7. Setup BLOCKCTRL Register

For (example) 16-bit pixels and Mask AND Mode:

A. Clear the "BLOCKCTRL" register by writing 0x0000_0000 to it

B. Write Fill = '1', BG = '0', M = 0x1, P = 0x4, and INTEN = '1' to the "BLOCKCTRL" register

C. Write EN = '1' to the "BLOCKCTRL" register

8. Wait for an Interrupt or Poll for EN = '0' in the BLOCKCTRL Register.

When the EN bit becomes cleared to '0', the Block Fill Algorithm function is complete.

8.6.4 Block Copy Function

The following sequence describes how to set up the registers used for a Block Copy function:

1. Setup Source Memory

A. Write the desired values to the SPEL field and the EPEL field in the "SRCPIXELSTRT" register.

SPEL is the starting pixel position within the word that the pixel-copy will begin with. EPEL is the ending pixel position within the word that the pixel-copy will end with. See Section 8.5.2.

For example, if the image to be copied is at position (51, 75) and the pixel depth is 16-bits, the value for SPEL is (51 x 16)% 32 = 16 = 0x10 and the value for EPEL is (75 x 16)% 32 = 16 = 0x10

B. Write the word-aligned value of the SDRAM address 'for the beginning of the image that is to be copied' to the "BLKDESTSTRT" register.

C. Write the line length value to the LEN field in the "SRCLINELENGTH" register, where LEN is determined by:

(1). Find how many pixels occupy a 32-bit word. For example, four 8-bit pixels can occupy a 32-bit word.

(2). Find the width of the display in pixels. For example, a 640x480 display has a width of 640 pixels.

(3). The line length, LEN, is determined by the stride of the display, that is, how many 32-bit words are needed to populate the width of the display with pixels. From steps 1 and 2, the stride for this example is 640 pixels divided by 4, where 4 is the number of 8-bit pixels that occupy a word. So, for this example, line length is 640 divided by 4 = 160 = 0xA0.

Usually the same LEN value is used in both the "SRCLINELENGTH" register and the "DESTLINELENGTH" register.

D. Write the value of the WIDTH field to the "BLKSRCWIDTH" register, where WIDTH is the number of 32-bit words, minus 1, that are needed to contain the pixels that

Copyright 2007 Cirrus Logic

comprise the first scan line of the source image.

For example, Table 8-21 shows that six 32-bit words are needed to contain six 24-bit pixels. So, LEN = 6 - 1 = 5 = 0x5.

Table 8-21. Words Needed for Six 24-Bit Pixels

Address	31 24	23 16	15 8	7 0
0x0000	P1	P0	P0	P0
0x0004	P2	P2	P1	P1
0x0008	P3	P3	P3	P2
0x000C	P5	P4	P4	P4
0x0010	P6	P6	P5	P5
0x0014	P7	P7	P7	P6

2. Setup Destination Memory

   A. Write the desired values to the SPEL field and the EPEL field in the "DESTPIXELSTRT" register.

      SPEL is the starting pixel position within the word that the pixel-copy will begin with. EPEL is the ending pixel position within the word that the pixel-copy will end with. See Section 8.5.2.

      For example, if the image is to be copied to position (81, 105) and the pixel depth is 16-bits, the value for SPEL is (81 x 16)% 32 = 16 = 0x10 and the value for EPEL is (105 x 16)% 32 = 16 = 0x10

   B. Write the word-aligned value of the SDRAM address 'for the beginning of the copy destination' to the "BLKDESTSTRT" register.

   C. Write the line length value to the LEN field in the "DESTLINELENGTH" register, where LEN is determined by:

      (1). Find how many pixels occupy a 32-bit word. For example, four 8-bit pixels can occupy a 32-bit word.

      (2). Find the width of the display in pixels. For example, a 640x480 display has a width of 640 pixels.

      (3). The line length, LEN, is determined by the stride of the display, that is, how many 32-bit words are needed to populate the width of the display with pixels. From steps 1 and 2, the stride for this example is 640 pixels divided by 4, where 4 is the number of 8-bit pixels that occupy a word. So, for this example, line length is 640 divided by 4 = 160 = 0xA0.

      Usually the same LEN value is used in both the "DESTLINELENGTH" register and the "SRCLINELENGTH" register.

   D. Write the value of the WIDTH field to the "BLKDESTWIDTH" register, where WIDTH specifies the number of 32-bit words, minus 1, that are needed to contain the pixels that comprise the 1st scan line of the destination image. For an example, please

Copyright 2007 Cirrus Logic

refer to the note in Section 8.5.2.4 on page 8-12.

E. Write the desired value to the HEIGHT field in the "BLKDESTHEIGHT" register, where HEIGHT = the height in lines of the image that is to be copied minus 1.

   For example, a 20-pixels x 10-lines image has a height of 10 lines. So, HEIGHT = 10 - 1 = 9 = 0x9.

F. The "BLOCKCTRL" register must be cleared to 0x0. This action clears out the previous graphics instruction. The EOI bit field must be cleared to '0' regardless of the interrupt enable status.

   The PACKD bit must be configured to indicate if the image to be copied has the same size for the source and the destination. Setting the PACKD bit allows transfers from images that are packed into whole word areas.

   The P bits must be configured for the BPP depth of the image to be copied.

   When using the AND/OR/XOR mask function, the M bits must be configured for the appropriate function.

   When using the AND/OR/XOR destination function, the D bits must be configured for the appropriate function.

   When using transparency, the TRANS bit must be enabled to '1'. This allows data from the source to be compared with the transparency pixel pattern to determine if the destination pixel is to be modified before it is written. Without this bit enabled, a direct block copy would occur.

   The SYDIR, SXDIR and DYDIR, DXDIR direction bits must be configured. These bits control the direction for the line accumulator, Y, and the word/pixel counter, X. In a left to right and top to bottom transfer:

   (1). if the destination is not exactly the same as the source, or

   (2). if the destination partially overlaps the source and has a destination starting word address greater than the source starting word address, then the source information may be corrupted before being read. For this condition, the direction bits for the transfer must be changed from left to right and top to bottom to right to left and bottom to top.

Note: Setting the source direction bits different from the destination direction bits is illegal and will have unpredictable results.

G. The INTEN bit must configured to enable or disable an interrupt signal to the ARM Core that occurs upon completion of the acceleration function.

H. After Step G is complete, write EN = '1' to start the Block Copy function.

I. Wait for an interrupt or poll for EN = '0'. When the EN bit is cleared to '0', the Block Copy function sequence is done.

Copyright 2007 Cirrus Logic

8

8.6.4.1 Example of Block Copy

To achieve the following display and pattern, follow Steps A to I in this section.

- Screen Size is 640x480x16-bits/pixel
- Screen memory starts at physical address 0x0000_0000
- Image to be copied is at physical address 0x0000_0960
- Image to be copied is at position (51, 75)
- Image destination is at position (300, 115)
- Source and destination width is 30 pixels

A. SRCPIXELSTRT = (51 * 16)% 32 = 16

B. BLKSRCSTRT = 0x960

C. SRCLINELENGTH = 640 / 2 pixels per word = 320 = 0x140

D. DESTPIXELSTRT:

SPEL = [(640 * 115) + 300] * 16% 32 = 0 = 0x0

EPEL = {[640 * (115 + 20)] + 300 +20} * 16% 32 = 0 = 0x0

E. BLKDESTSTRT =[(640 * 115) + 300] * 2 = 147800 = 0x24158

F. BLKDESTWIDTH = (30 / 2) - 1 = 14 = 0xE

G. BLKDESTHEIGHT = 20

H. BLOCKCTRL:

Write 0x0000_0000 to the BLOCKCTRL register to clear it.

Write PACKD = '0' to specify that the size of the source and destination images are the same.

Write P = 0x4 to specify 16-bits/pixel.

Write SXDIR = '0', SYDIR = '0', DXDIR = '0', DYDIR = '0' to specify that pixels are placed into the destination image left to right and top to bottom.

Write FILL = '1' to enable the block copy or Block Copy function.

Write INTEN = '1' to enable the Graphics Accelerator interrupt.

Write EN = '1' to initiate graphics processing

I. The final step is to wait for an interrupt or poll for EN = '0'. When the EN bit becomes cleared to '0', the Block Copy function is complete.

Copyright 2007 Cirrus Logic

8.7 Registers

Table 8-22. Graphics Accelerator Registers

Address	Name	SW locked	Type	Size	Description
0x8004_0000	"SRCPIXELSTRT"	No	Read/Write	5 bits	Source Pixel Start Register
0x8004_0004	"DESTPIXELSTRT"	No	Read/Write	5 + 5 bits	Destination Pixel Start/End Register
0x8004_0008	"BLKSRCSTRT"	No	Read/Write	32 bits	Block Source Word Address Start Register
0x8004_000C	"BLKDESTSTRT"	No	Read/Write	32 bits	Block Destination Word Address Start Register
0x8004_0010	"BLKSRCWIDTH"	No	Read/Write	12 bits	Block Function Source Width Register
0x8004_0014	SRCLINELENGTH"SRCLINELENGTH"	No	Read/Write	12 bits	Block Source Line Length Register
0x8004_0018	"BLKDESTWIDTH"	No	Read/Write	12 bits	Block Function Destination Width Register
0x8004_001C	"BLKDESTHEIGHT"	No	Read/Write	11 bits	Block Function Destination Height Register
0x8004_0020	"DESTLINELENGTH"	No	Read/Write	12 bits	Destination Line Length Register
0x8004_0024	"BLOCKCTRL"	No	Read/Write	16 bits	Block Function Control Register
0x8004_0028	"TRANSPATTRN"	No	Read/Write	24 bits	Block Function Transparency Register
0x8004_002C	"BLOCKMASK"	No	Read/Write	24 bits	Block Function Mask Register
0x8004_0030	"BACKGROUND"	No	Read/Write	24 bits	Block Function Background Register
0x8004_0034	"LINEINC"	No	Read/Write	12 + 12 bits	Line Draw Increment Register
0x8004_0038	"LINEINIT"	No	Read/Write	12 + 12 bits	Line Draw Initialization Register
0x8004_003C	"LINEPATTRN"	No	Read/Write	20 bits	Line Pattern Register

Note: Graphics Accelerator registers are intended to be word accessed only. Since the least significant bytes of the address bus are not decoded, byte and half word accesses are illegal and may yield unpredictable results.

Copyright 2007 Cirrus Logic

Register Descriptions

SRCPIXELSTRT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												PEL			

Address: **0x8004_0000 - Read/Write**

Default: **0x0000_0000**

Mask: **0x0000_001F**

Definition: **Source Pixel Start register**

Bit Descriptions:

RSVD: Reserved - Unknown during read

PEL: Source Pixel Location - Read/Write

For the starting pixel (at the starting X-Y coordinate of the 1st scan line) of the source image for a block copy, the value in this field specifies where the beginning bit of the pixel is located in a 32-bit word. For example, if the beginning bit of a 16-bit pixel is located at bit 16 of a 32-bit word, PEL = 0x10.

The PEL field and the ADR field in the "BLKSRCSTRT" register together define the starting pixel's address in the SDRAM frame buffer. In REMAP mode, the starting location written to the PEL field can be defined with bit-level granularity. For all other modes, the granularity must be a multiple of the pixel size: e.g. in 8 bpp mode, acceptable PEL values are 0x00, 0x08, 0x10, and 0x18.

DESTPIXELSTRT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD												EPEL			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												SPEL			

Address: **0x8004_0004 - Read/Write**

Copyright 2007 Cirrus Logic

8

Default: **0x0000_0000**

Mask: **0x001F_001F**

Definition: **Destination Pixel Start/End register**

Bit Descriptions:

RSVD: Reserved - Unknown during read

EPEL: Destination Pixel Location - Read/Write

For the ending pixel (at the ending X-Y coordinate of the 1st scan line) of the destination image for a block copy, the value in this field specifies where the beginning bit of the ending pixel is located in a 32-bit word. For example, if the beginning bit of an 8-bit pixel is located at bit 24 of a 32-bit word, EPEL = 0x18.

The EPEL field and the ADR field in the "BLKDESTSTRT" register together define the destination ending pixel's address in the SDRAM frame buffer. Granularity must be a multiple of the pixel size in all video display modes. For example,.acceptable values in 8 bpp mode are 0x00, 0x08, 0x10, and 0x18.

SPEL: Source Pixel Location - Read/Write

For the starting pixel (at the starting X-Y coordinate of the 1st scan line) of the destination image for a block copy, the value in this field specifies where the beginning bit of the pixel is located in a 32-bit word. For example, if the beginning bit of a 16-bit pixel is located at bit 16 of a 32-bit word, PEL = 0x10.

The SPEL field and the ADR field in the "BLKDESTSTRT" register together define the destination starting pixel's address in the SDRAM frame buffer. Granularity must be a multiple of the pixel size in all video display modes. For example,.acceptable values in 8 bpp mode are 0x00, 0x08, 0x10, and 0x18.

BLKSRCSTRT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADR															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADR														NA	

Address: **0x8004_0008 - Read/Write**

Copyright 2007 Cirrus Logic

Default: **0x0000_0000**

Mask: **0xFFFF_FFFC**

Definition: **Block Source Word Address Start register**

Bit Descriptions:

ADR: Address - Read/Write

The value in this field specifies the word address of the SDRAM frame buffer location that contains the starting pixel (of the first scan line) of the source image.

The ADR field and the PEL field in the "SRCPIXELSTRT" register together define the starting pixel's address in the SDRAM frame buffer of the source image.

NA: Not Assigned - Not used, returns written value

BLKDESTSTRT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADR															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADR															NA

Address: **0x8004_000C - Read/Write**

Default: **0x0000_0000**

Mask: **0xFFFF_FFFC**

Definition: **Block Destination Word Address Start register**

Bit Descriptions:

ADR: Address - Read/Write

The value in this field specifies the word address of the SDRAM frame buffer location that contains the starting pixel (of the first scan line) of the destination image.

The ADR field and the SPEL field in the "DESTPIXELSTRT" register together define the starting pixel's address in the SDRAM frame buffer of the destination image.

NA: Not Assigned - Not used, returns written value

Copyright 2007 Cirrus Logic

8

BLKSRCWIDTH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				WIDTH											

Address: **0x8004_0010 - Read/Write**

Default: **0x0000_0000**

Mask: **0x0000_0FFF**

Definition: **Block Function Source Width Register**

Bit Descriptions:

RSVD: Reserved. Unknown during read.

WIDTH: Width - Read/Write

For a Block Copy function, the value in this field specifies the number of 32-bit words, minus 1, that are needed to contain the pixels in the 1st scan line of the source image. For an example, please refer to Table 8-18 on page 8-12. Six 32-bit words are needed to contain six 24-bit pixels. So, WIDTH = 6 - 1 = 5 = 0x5.

The maximum value for the field is 0xFFE = 4095 words.

SRCLINELENGTH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				LEN											

Address: **0x8004_0014 - Read/Write**

Default: **0x0000_0000**

Mask: **0x0000_0FFF**

Definition: **Block Source Line Length Register**

Bit Descriptions:

RSVD: Reserved - Unknown during read

LEN: Length - Read/Write

Copyright 2007 Cirrus Logic

The value in this field specifies the number of 32 bit words, minus 1, that are needed to contain all of the pixels that comprise width of the display. The value of LEN is determined by:

1) Find how many pixels occupy a 32-bit word. For example, four 8-bit pixels can occupy a 32-bit word.

2) Find the width of the display in pixels. For example, a 640x480 display has a width of 640 pixels.

3) The line length, LEN, is determined by the stride of the display, that is, how many 32-bit words are needed to populate the width of the display with pixels. From steps 1 and 2, the stride for this example is 640 pixels divided by 4, where 4 is the number of 8-bit pixels that occupy a word. So, for this example, LEN = 640 / 4 = 160 = 0xA0.

Usually the same LEN value is used in both the SRCLINELENGTH register and the DESTLINELENGTH register.

BLKDESTWIDTH

Address: **0x8004_0018 - Read/Write**

Default: **0x0000_0000**

Mask: **0x0000_0FFF**

Definition: **Block Function Destination Width Register.**

Bit Descriptions:

RSVD: Reserved - Unknown during read

WIDTH: Width - Read/ Write

For Block Fill and Block Copy functions, the value in this field specifies the number of 32-bit words, minus 1, that are needed to contain the pixels in the 1st scan line of the destination image. For example, please refer to the note in Section 8.5.2.4 on page 8-12. The maximum value for the field is 0xFFE = 4095 words.

Copyright 2007 Cirrus Logic

8

For Line Draw functions, the method to determine the value of WIDTH is dependent on the line draw algorithm. For the Burnishing algorithm, please refer to BLKDESTWIDTH in Section 8.6.1 on page 8-13. For the DX/DY algorithm, please refer to BLKDESTWIDTH in Section 8.6.3 on page 8-16. The value of WIDTH is multiplied by the value of HEIGHT in the BLKDESTHEIGHT register to determine the number of line draw iterations.

BLKDESTHEIGHT

Address: **0x8004_001C - Read/Write**

Default: **0x0000_0000**

Mask: **0x0000_07FF**

Definition: **Block Function Destination Height Register**

Bit Descriptions:

RSVD: Reserved - Unknown during read

HEIGHT: Height - Read/Write

For Block Fill or Block Copy functions, the value in this field specifies the height in 'lines minus 1' of the destination image. Since there is no BLKSRCHEIGHT register, the source image must have the same height as the destination image.

For Line Draw functions, the value in this field specifies the distance in 'lines minus 1' between Y_dest_end and Y_dest_start. The method to determine the value of HEIGHT is dependent on the line draw algorithm. For the Breshenham algorithm, please refer to BLKDESTHEIGHT in Section 8.6.1 on page 8-13. The value of HEIGHT is multiplied by the value of WIDTH in the BLKDESTWIDTH register to determine the number of line draw iterations.

Copyright 2007 Cirrus Logic

DESTLINELENGTH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				LEN											

Address: **0x8004_0020 - Read/Write**

Default: **0x0000_0000**

Mask: **0x0000_0FFF**

Definition: **Block Destination Line Length Register**

Bit Descriptions:

RSVD: Reserved - Unknown during read

LEN: Length - Read/Write

The value in this field specifies the number of 32 bit words, minus 1, that are needed to contain all of the pixels that comprise width of the display. The value of LEN is determined by:

1) Find how many pixels occupy a 32-bit word. For example, four 8-bit pixels can occupy a 32-bit word.

2) Find the width of the display in pixels. For example, a 640x480 display has a width of 640 pixels.

3) The line length, LEN, is determined by the stride of the display, that is, how many 32-bit words are needed to populate the width of the display with pixels. From steps 1 and 2, the stride for this example is 640 pixels divided by 4, where 4 is the number of 8-bit pixels that occupy a word. So, for this example, LEN = 640 / 4 = 160 = 0xA0.

Usually the same LEN value is used in both the DESTLINELENGTH register and the SRCLINELENGTH register.

Copyright 2007 Cirrus Logic

BLOCKCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD											PACKD	P			ERROR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTEOI	BG	REMAP	D1	D0	M1	M0	SYDIR	SXDIR	DYDIR	DXDIR	LINE	FILL	TRANS	INTEN	EN

Address: **0x8004_0024 - Read/Write**

Default: **0x0000_0000**

Mask: **0x001F_FFFF**

Definition: **Block Function Control Register**

Bit Descriptions:

RSVD: Reserved - Unknown during read

PACKD: Packed Image Bit - Read/Write

This bit is normally '0' to indicate that the source and destination images during a Block Copy function are the same size.

When this bit is '1', the a block transfer image source is stored in packed format. Packed format indicates that the source image is not the same dimensions as the destination image, and that source information transfers are whole words with the possible exceptions of the beginning and ending words. This allows images to be packed into any square configuration of whole words, including a serial stream.

P: Bits Per Pixel - Read/Write

The value of this field, as shown in Table 8-23, specifies the pixel mode (depth) that is used for Graphics Accelerator functions. The Raster Engine has a similar pixel depth field, but it's value is independent from this P value and may be either different or the same.

Table 8-23. Pixel Mode Encoding

P2	P1	P0	Pixel Mode
0	0	0	not defined
0	0	1	4 bit per pixel
0	1	0	8 bits per pixel

Copyright 2007 Cirrus Logic

Table 8-23. Pixel Mode Encoding

P2	P1	P0	Pixel Mode
0	1	1	not defined
1	0	0	16 bits per pixel
1	0	1	not defined
1	1	0	24 bits per pixel packed
1	1	1	32 bits per pixel (24 bpp unpacked)

ERROR: Error Indicator - Read/Write

1 - Bus error has occurred

0 - No error.

INTEOI: Interrupt / End of Interrupt - Read/Write

Reading this bit returns the status of the Block Fill or Block Copy function interrupt (active high):

'1' - Interrupt request. Indicates Block Fill or Block Copy function has completed.

'0' - No interrupt request. Indicates Block Fill or Block Copy function has not completed.

Writing '0' to this bit will clear the interrupt request; writing '1' to this bit will generate an interrupt request.

This bit may be used to cancel a 'broken' graphics function that never completes. Masking the interrupt by writing INTEN = '0', and writing INTEOI = '1' will halt the current Graphics Accelerator function.

BG: Background - Read/Write

When this bit is '0' during remap (REMAP = '0'), source image pixels that have a value of '0' are unaffected (transparent) when they are copied to the destination image.

When this bit is '1' during remap (REMAP = '1'), source image pixels that have a value of '0' are copied to the destination image with the color value in the BG field of the BACKGROUND register.

Reading this bit returns a valid value only when EN = '1'.

REMAP: Pixel Expansion Mapping Function Enable - Read/Write

The value of REMAP enables or disables the Pixel Expansion Mapping Function:

Copyright 2007 Cirrus Logic

'1' - Pixel Expansion Mapping Function enabled

'0' - Pixel Expansion Mapping Function disabled

The Pixel Expansion Mapping Function converts single bit pixels in the source image to defined pixel-depth (see Table 8-23) pixels in the destination image.

When BG = '0', source image pixels are unaffected (transparent) when they are copied to the destination image. When BG = '1', source image pixels that have a value of '0' are copied to the destination image with the color value in the BG field of the BACKGROUND register and source image pixels that have a value of '1' are copied to the destination image with the color value in the MASK field of the BLOCKMASK register.

D: Destination Mode - Read/Write

The value in the this field specifies the destination mode:
'00' - Disabled
'01' - Destination AND Mode
'10' - Destination OR Mode
'11' - Destination XOR Mode

M: Mask Mode - Read/Write

The value in the this field specifies the mask mode:
'00' - Disabled
'01' - Mask AND Mode
'10' - Mask OR Mode
'11' - Mask XOR Mode

SYDIR, SXDIR: Counter/Accumulator Direction - Read/Write

Write the values of the DYDIR and DXDIR bits to the SYDIR and DXDIR bits, respectively.

DYDIR, DXDIR: Counter/Accumulator and Line Direction - Read/Write

The value of these bits specifies the general direction that the current Graphics Acceleration function places pixels on the display:

For a Block Fill or Block Copy function:

DXDIR = '1' - Left in X

DXDIR = '0' - Right in X

DYDIR = '1' - Up in Y

Copyright 2007 Cirrus Logic

DYDIR = '0' - Down in Y

For a Line Draw function:

DXDIR = '1' - If X2 > X1

DXDIR = '0' - If X2 <= X1

DYDIR = '1' - If Y2 > Y1

DYDIR = '0' - If Y2 <= Y1

LINE: Line Draw Function Enable - Read/Write

'0' - Line draw disabled
'1' - Line draw enabled

Reading this bit returns a valid value only when EN = '1'.

FILL: FILL Function Enable - Read/Write

'0' - Fill disabled
'1' - Fill (with mask value) enabled

Reading this bit returns a valid value only when EN = '1'.

TRANS: Transparency Enable - Read/Write

'0' - Transparency disabled
'1' - Transparency enabled

Reading this bit returns a valid value only when EN = '1'.

INTEN: Graphics Accelerator Interrupt Enable - Read/Write

'0' - Interrupt disabled
'1' - Interrupt enabled

EN: Initiate Graphics Acceleration Function - Read/Write

Read:

'0' - Graphics processing completed
'1' - Graphics processing in progress

Write:

'0' - Terminate current graphics processing function
'1' - Initiate graphics processing function

Copyright 2007 Cirrus Logic

8

TRANSPATTRN

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD								PATRN							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PATRN															

Address: **0x8004_0028 - Read/Write**

Default: **0x0000_0000**

Mask: **0x00FF_FFFF**

Definition: **Block Function Transparency Pattern Register**

Bit Descriptions:

RSVD: Reserved - Unknown during read

PATRN: Transparent Bit Pattern - Read/Write

The value in this field specifies a transparent bit pattern. Transparent pixel transfers are not written. The transparent pixel definition is located in the least significant BPP part of the field for modes less than 24 bpp. Bits 0-23 are used for 24 bpp mode, bits 0-15 are used for 16 bpp mode, bits 0-7 are used for 8 bpp mode, and bits 0-3 are used for 4 bpp mode.

BLOCKMASK

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD								MASK							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MASK															

Address: **0x8004_002C - Read/Write**

Default: **0x0000_0000**

Mask: **0x00FF_FFFF**

Definition: **Block Mask Register**

Bit Descriptions:

RSVD: Reserved - Unknown during read

Copyright 2007 Cirrus Logic

MASK: Mask - Read/Write

For a Block Copy function, the value in this field specifies the logical mask, if used. If BG = '1', M = '00', and REMAP = '1' in the BLOCKCTRL register, the value specifies the destination foreground color for source image pixels = '1'.

For Block Fill and Line Draw functions, the value in this field specifies the pixel color for the destination image.

The mask or color value is located in the least significant BPP part of the register for modes less than 24 bpp. Bits 0-23 are used for 24 bpp mode, bits 0-15 are used for 16 bpp mode, bits 0-7 are used for 8 bpp mode, and bits 0-3 are used for 4 bpp mode.

BACKGROUND

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD								BG							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BG															

Address: **0x8004_0030 - Read/Write**

Default: **0x0000_0000**

Mask: **0x00FF_FFFF**

Definition: **Block Function Background Register**

Bit Descriptions:

RSVD: Reserved - Unknown during read

BG: Background - Read/Write

When performing remap operations without transparency (REMAP = '1' and BG = '1' in the BLOCKCTRL register), the value in this field specifies the destination background pixel color for source pixels that have a value of '0'.

Bits that are '1' in this field can be used with Line Draw functions to specify a 'blank space' in the drawn line.

The pixel color value is located in the least significant BPP part of the field for modes less than 24 bpp.

Copyright 2007 Cirrus Logic

LINEINC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				YINC											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				XINC											

Address: **0x8004_00343 - Read/Write**

Default: **0x0000_0000**

Mask: **0x0FFF_0FFF**

Definition: **Line Draw Increment Register**

Bit Descriptions:

RSVD: Reserved - Unknown during read

YINC: Y Increment - Read/Write

The value in this field specifies a 12-bit binary fraction of a pixel to be accumulated in the vertical (Y) direction during a Line Draw function. The maximum value is 4095/4096 and the minimum value is 1/4096.

XINC: X Increment - Read/Write

The value in this field specifies a 12 bit binary fraction of a pixel to be accumulated in the horizontal (X) direction during a Line Draw function. The maximum value is 4095/4096 and the minimum value is 1/4096.

LINEINIT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				YINIT											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				XINIT											

Address: **0x8004_0038 - Read/Write**

Default: **0x0000_0000**

Mask: **0x0FFF_0FFF**

Definition: **Line Draw Initialization Register**

Copyright 2007 Cirrus Logic

Bit Descriptions:

RSVD: Reserved - Unknown during read

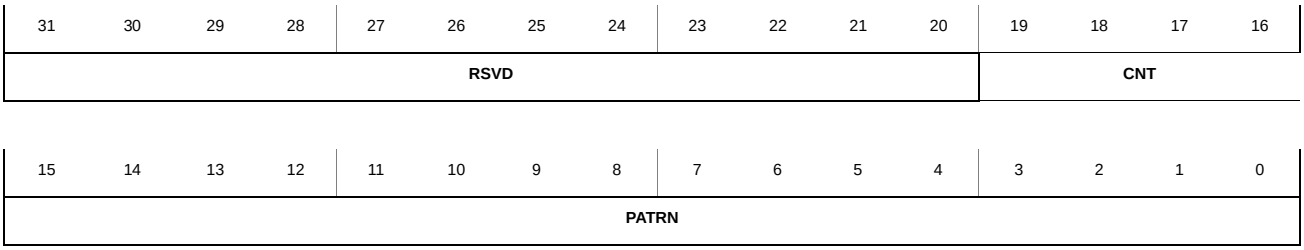
YINIT: Y Initialization - Read/Write

The value in this field specifies a 12 bit binary fraction of a pixel that provides sub-pixel precision to the algorithm. The minimum fractional value is 1/4096. This field can also be initialized to account for truncation errors in the drawing algorithm.

XINIT: X Initialization - Read/Write

The value in this field specifies a 12 bit binary fraction of a pixel that provides sub-pixel precision to the algorithm. The minimum fractional value is 1/4096. This field can also be initialized to account for truncation errors in the drawing algorithm.

LINEPATTRN



Address: **0x8004_003C - Read/Write**

Default: **0x000F_FFFF**

Mask: **0x000F_FFFF**

Definition: **Line Pattern Register**

Bit Descriptions:

RSVD: Reserved - Unknown during read

CNT: The value in this field specifies the pixel position in the PATRN field that defines the end of the pattern. It is used as the repeat interval for the pattern counter.

PATRN: The bit values in this field specify an 'on' and 'off' pattern that is to be used during a Line Draw function. The pattern will repeat based on the CNT value.

A '1' causes a pixel fill from the BLOCKMASK register.

Copyright 2007 Cirrus Logic

If BG = '1' in the BLOCKCTRL register, a '0' causes a pixel fill from the BACKGROUND register. If BG = '0' in the BLOCKCTRL register, a '0' is transparent.

When drawing solid lines, write LINEPATTERN = 0x000F_FFFF.

Copyright 2007 Cirrus Logic

Chapter 9
1/10/100 Mbps Ethernet LAN Controller

9

9.1 Introduction

The Ethernet LAN Controller incorporates all the logic needed to interface directly to the AHB and to the Media Independent Interface (MII). It includes local memory and DMA control, and supports full duplex operation with flow control support. Figure 9-1 shows a simplified block diagram.

This block was designed with a RAM of 544 words, each word containing 33 bits. These RAMs are used for packet buffering and controller data storage. One RAM is dedicated to the receiver, and one dedicated to the transmitter. These RAMs are mapped into the register space and are accessible via the AHB.

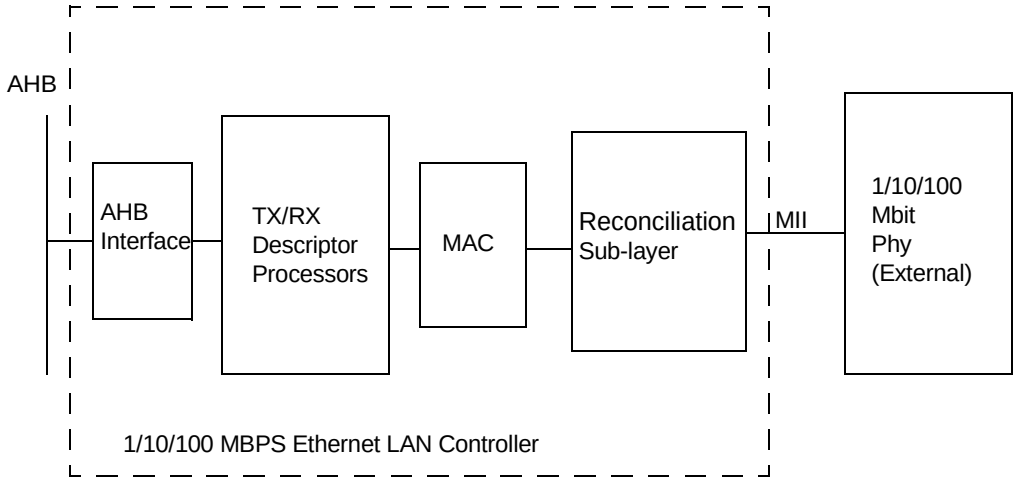


Figure 9-1. 1/10/100 Mbps Ethernet LAN Controller Block Diagram

9.1.1 Detailed Description

9.1.1.1 Host Interface and Descriptor Processor

The Host Interface can be functionally decomposed into the AHB Interface Controller and the Descriptor Processor. The AHB Interface Controller implements the actual connection to the AHB. The controller responds as a AHB bus slave for register programming, and acts as an AHB bus master for data transfers.

Copyright 2007 Cirrus Logic

The Descriptor Processor implements the Hardware Adapter Interface Algorithm and generates transfer requests to the AHB Interface Controller. The back-end interfaces to the MAC controllers and services MAC requests to run accesses to the FIFO and update queue status. The Descriptor Processor also generates internal requests for descriptor fetches. A priority arbiter arbitrates among the various requests and generates transfer requests to the AHB Interface Controller. There are 6 queues that require service in system memory:

- RxData: Write received frame data to host memory.
- RxStatus: Write received frame status to host memory.
- TxData: Read frame data from host memory.
- TxStatus: Write transmitted frame status to host memory.
- RxDescriptor: Read descriptors from host memory.
- TxDescriptor: Read descriptors from host memory.

Each queue generates a hard request (for urgent service) and a soft request (not urgent, but queue can run transfers). The priority assigned to the queues varies depending on the state of the system, but hard requests are prioritized over soft requests, and AHB write requests are prioritized over AHB read requests to allow faster back-to-back transfers.

9.1.1.2 Reset and Initialization

The Ethernet LAN Controller has three reset sources: the AHB reset, software reset from the SelfCtl register, and individual channel resets via the BMCtl register. The PHY is reset with the PHYRES function in compliance with the 802.3 specifications and has no effect on the MAC layer and up.

AHB reset initializes the entire controller, except for the receive MAC. The receive MAC is initialized by a SOFT_RESET. Upon AHB reset the AHB Interface and Descriptor Processor is put into a quiescent state.

Software Reset generates a SOFT_RESET which resets the Descriptor Processor, FIFO, and MAC. SOFT_RESET occurring in the middle of a frame transmission will result in the transmitted frame being truncated on the line. SOFT_RESET occurring in the middle of a received frame will result in the reset of the frame being dropped. The configuration registers remain intact during a soft reset. A SOFT_RESET should be issued following a power-on to ensure the receive MAC is fully initialized.

9.1.1.3 Power-down Modes

The only power-down option is to stop the TXCLK and RXCLK by disabling the PHY.

9.1.1.4 Address Space

The Address space is mapped as:

MACBase + 0x0000 - MACBase + 0x00FF: MAC setup registers.
MACBase + 0x0100 - MACBase + 0x011F: MAC configuration registers, only first 4 words used.

Copyright 2007 Cirrus Logic

The RAM blocks are interleaved in the AHB address space. AHB address bits 0 and 1 are byte selects and must be zero for direct access. AHB address bit 2 selects the left or right RAM array, which is the Transmit or Receive array. AHB address bits 3,4, and 5 perform a 1-of-8 column select. Address bit 6 selects the even or odd row address. Address bits 7, 8, 9, and 10 decode the rows. Thus from an AHB addressing perspective, the MAC FIFOs are one large RAM array.

Table 9-1 defines the FIFO RAM address map as it appears in the address space. Address are in byte units. All data transfers to the FIFO RAM are restricted to words.

Caution: Accessing the FIFO RAM while the MAC is operating will likely cause a malfunction.

There is no arbitration logic between direct AHB access and MAC Descriptor Processor access.

Table 9-1. FIFO RAM Address Map

FIFO RAM Address Map	Usage
0x8001_4000 to 0x8001_47FF	Rx Data
0x8001_4800 to 0x8001_4FFF	Tx Data
0x8001_5000 to 0x8001_503F	Rx Status
0x8001_5040 to 0x8001_507F	Tx Status
0x8001_5080 to 0x8001_50BF	Rx Descriptor
0x8001_50C0 to 0x8001_50FF	Tx Descriptor

The MAC configurations registers and FIFO RAMs are only word accessible

9.1.2 MAC Engine

The MAC engine is compliant with the requirements of ISO/IEC 8802-3 (1993), Sections 3 and 4.

9.1.2.1 Data Encapsulation

In transmission, the MAC automatically prepends the preamble, and computes and appends the FCS. The data after the SFD and before the FCS is supplied by the host as the transmitted data. FCS generation by the MAC may be disabled by setting InhibitCRC bit in the Transmit Frame Descriptor. Refer to Figure 9-2.

Copyright 2007 Cirrus Logic

9

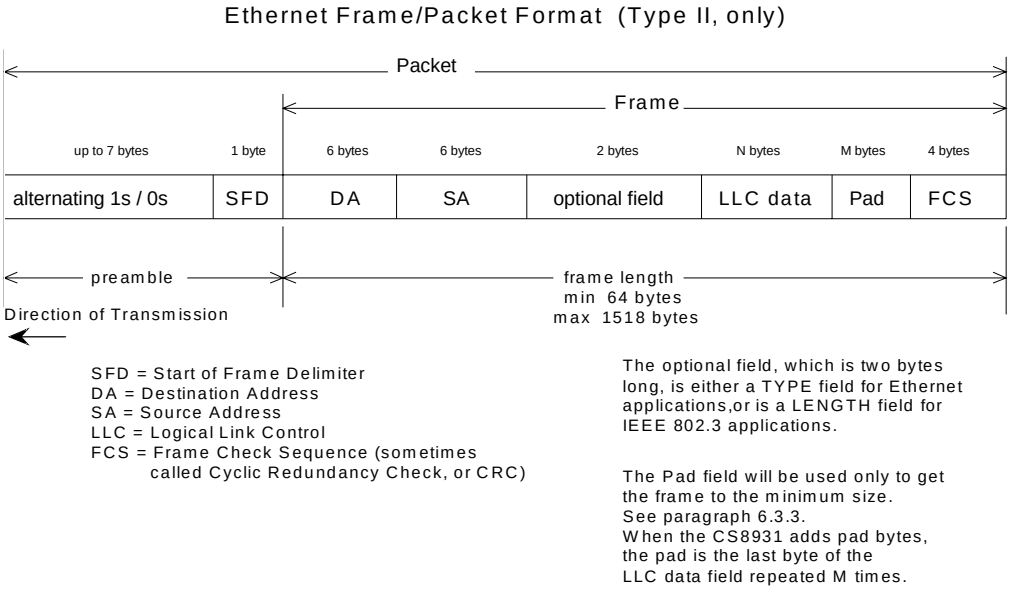


Figure 9-2. Ethernet Frame / Packet Format (Type II only)

In the receiver, the MAC detects the preamble and locks onto the embedded clock. The MAC performs destination address filtering (individual, group, broadcast, promiscuous) on the DA. The MAC engine computes the correct FCS, and reports if the received FCS is "good" or "bad". The data after the SFD and before the FCS is supplied to the host as the received data. The received FCS may also be passed to the host by setting RXCtl.BCRC.

Copyright 2007 Cirrus Logic

9

9.1.3 Packet Transmission Process

This section explains the complete packet transmission process as seen on the Ethernet line. This process includes: carrier deference, back-off, packet transmission, transmission of EOF, and SQE test. Refer to Figure 9-3.

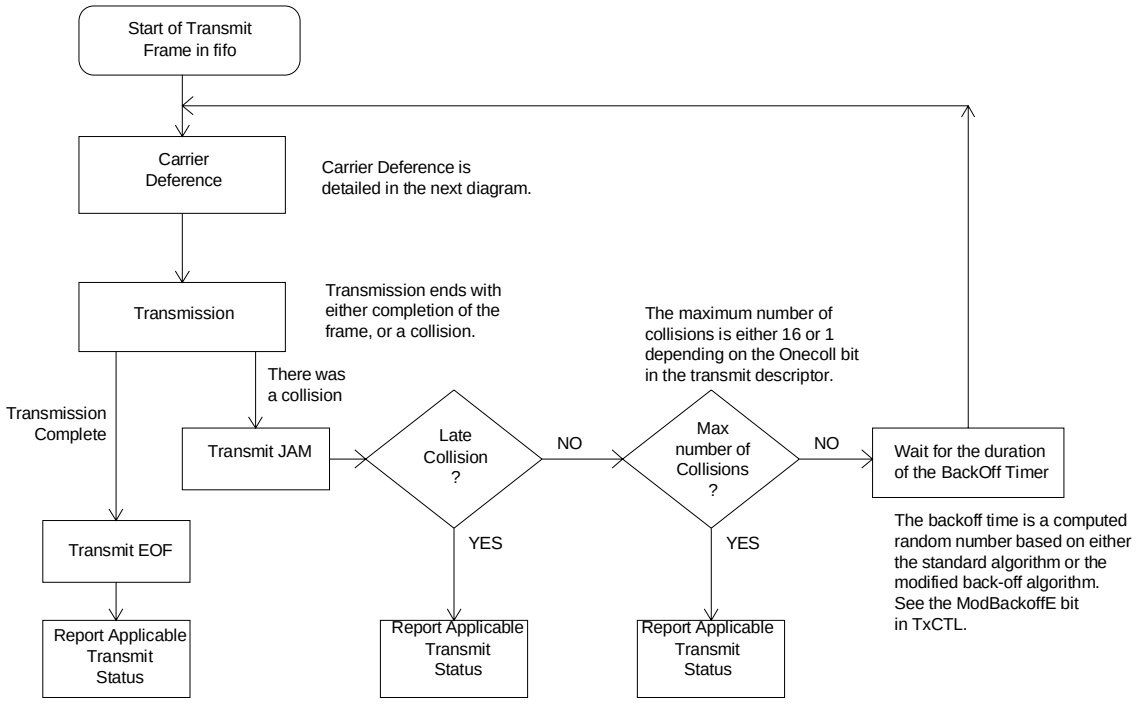


Figure 9-3. Packet Transmission Process

The Ethernet/ISO/IEC 8802-3 topology is a single shared medium with several stations. Only one station can transmit at a time. The access method is called Carrier Sense Multiple Access with Collision Detection (CSMA/CD). This method is a "listen before talk" mechanism that has an added feature to end transmissions when two, or more, stations start transmissions at nearly the same time.

The CSMA portion of this method provides collision avoidance. Each station monitors its receiver for carrier activity. When activity is detected, the medium is busy, and the MAC defers (waits) until the medium no longer has a carrier.

9.1.3.1 Carrier Deference

Refer to Figure 9-4. Once sufficient bytes have been written to the transmit FIFO, the MAC layer immediately moves to the Carrier Deference State Diagram. The Carrier Deference state is independent of entry into the state diagram. The MAC layer may enter the state diagram in any of its five states. The MAC layer exits the Carrier Deference only from the IFG

Copyright 2007 Cirrus Logic

9

Complete state. Thus, the Carrier Deference state may be entered and exited immediately, or there may be a delay depending on the state when entered.

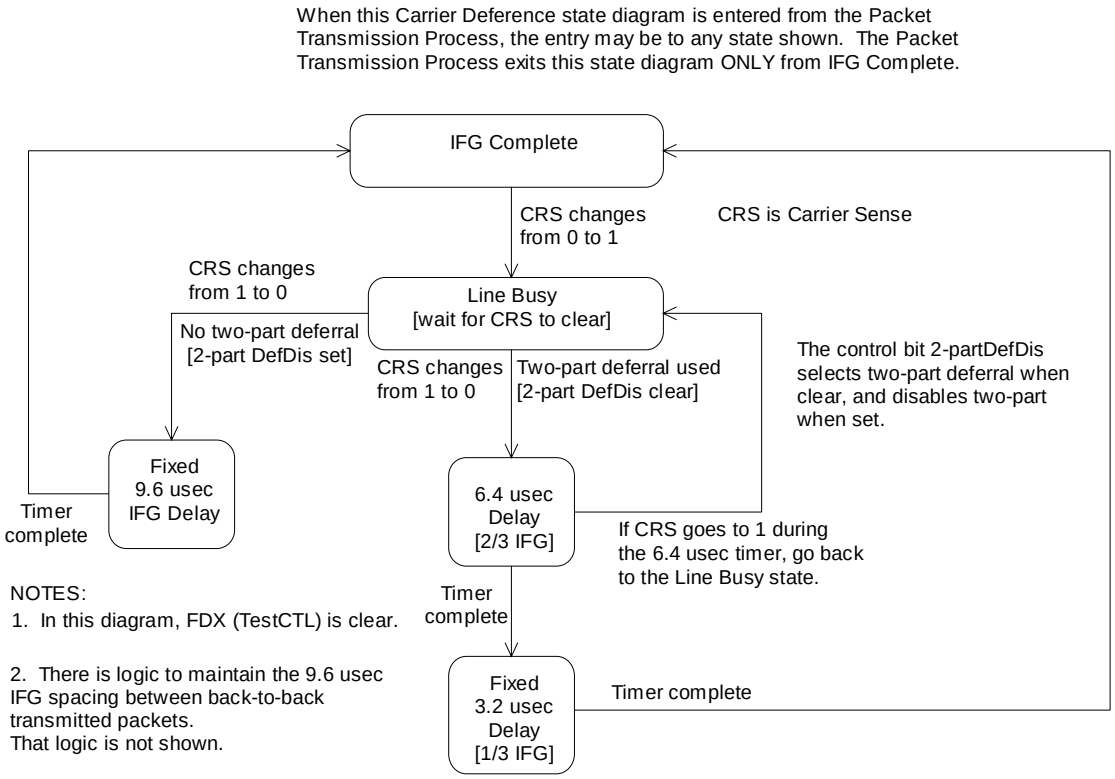


Figure 9-4. Carrier Deference State Diagram

When CRS becomes active, the Line Busy state is entered. This state is held until CRS returns to clear which starts the IFG timer. The time-out process after CRS clears is called Carrier Deference. In the MAC, Carrier Deference has two options as selected by the bit 2-part DefDis (TXCtl). If 2-part DefDis is clear, the two part deferral is used which meets the requirements of ISO/IEC 8802-3 paragraph 4.2.3.2.1. As shown in the diagram, if CRS becomes active during the first 2/3 (6.4 µsec) of the IFG, the MAC restarts the IFG timer. If CRS becomes active during the last 1/3 of the IFG, the timer is not restarted to ensure fair access to the medium.

If 2-part DefDis is set, the two part deferral is disabled. In this option, the IFG timer is allowed to complete even if CRS becomes active after the timer has started.

The 2-part deferral has an advantage for AUI connections to either 10BASE-2 or 10BASE-5. If the deferral process simply allowed the IFG timer to complete, then it is possible for a short Inter Frame Gap to be generated. The 2-part deferral prevents short IFGs. The disadvantage of the 2-part deferral is longer deferrals. In 10BASE-T systems, either deferral method should operate about the same.

Copyright 2007 Cirrus Logic

9.1.4 Transmit Back-Off

Refer to Figure 9-3. Once transmission is started, either the transmission is completed, or there is a collision. There are two kinds of collision: normal collision (one that occurs within the first 512 bits of the packet) and late collision (one that occurs after the first 512 bits). In either collision type, the MAC engine always sends a 32 bit jam sequence, and stops transmission.

After a normal collision and the jam, transmission is stopped, or "backed-off". The MAC attempts transmission again according to one of two algorithms. The ISO/IEC standard algorithm or a modified back-off algorithm may be used, and the host chooses which algorithm through the ModBackoffE control bit (TXCtl). The standard algorithm from ISO/IEC paragraph 4.2.3.2.5 is called the "truncated binary exponential backoff" and is shown below:

$$0 <= r <= 2^k$$

where r is a random integer for the number of slot times the MAC waits before attempting another transmission, and a slot time is time of 512 bits (51.2 µsec), k = minimum (n,10), and n is the nth retransmission attempt. The modified back-off algorithm uses delays longer than the ISO/IEC standard after each of the first three transmit collisions as shown below:

$$0 <= r <= 2^k$$

where k = minimum (n,10), but not less than 3, and n is the nth retransmission attempt

The advantage of the modified algorithm over the standard algorithm is that the modification reduces the possibility of multiple collisions on any transmission attempt. The disadvantage is that the modification extends the maximum time needed to acquire access to the medium.

The host may choose to disable the back-off algorithm altogether. This is done through the control bit DisableBackoff (TestCtl). When set, the MAC transmitter waits for the Inter Frame Gap time before starting transmission. There is no back-off algorithm employed. When clear, the MAC uses either the standard or the modified algorithm.

9.1.4.1 Transmission

After the transmission has passed the time for a normal collision (512 bits), then transmission is either completed, or aborted due to a late collision. For a late collision, the transmitter sends the 32 bit jam sequence, but does not back-off and try again. When a late collision occurs, Out-of-wdw collision (XStatQ) is set. A late collision is not retried, because the first 64 bytes of the FIFO are freed after the normal collision window, and will likely be refilled by a following packet. Driver intervention is needed to reconstruct the FIFO data.

9.1.4.2 The FCS Field

If InhibitCRC (Transmit Descriptor) is clear, the MAC automatically appends the standard 32 bit FCS to the end of the frame. The MAC tests the last 32 bits received against the standard CRC computation. If received in error, CRCerror (RStatQ) is set. If CRCerroriE (Interrupt Enable) is set, there is an interrupt associated with CRCerror. The standard CRC conforms to ISO/IEC 8802-3 section 3.2.8. The polynomial for the CRC is:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Copyright 2007 Cirrus Logic

The resultant 32 bit field is transmitted on the line with bit X^{31} first through X^{0} last.

9

9.1.4.3 Bit Order

In compliance with ISO/IEC 8802-3 section 3.3, each byte is transmitted low order bit first, except for the CRC, as noted in Section 9.1.4.2 on page 9--7.

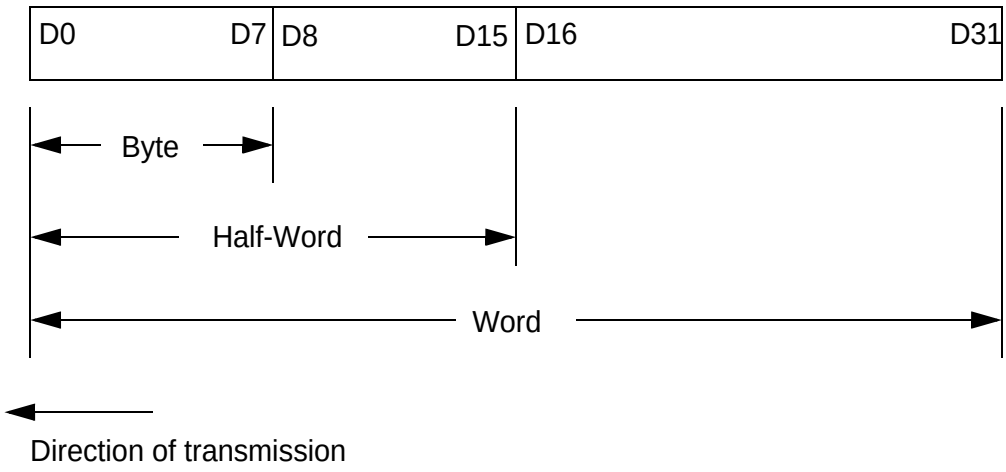


Figure 9-5. Data Bit Transmission Order

9.1.4.4 Destination Address (DA) Filter

There are two forms of destination address filtering performed by the MAC, perfect filtering, where the address is checked for an exact match, and hashing, where the address is checked for inclusion in a group. In addition there is a mode to accept all destination addresses which is enabled via the RXCtl.PA bit.

9.1.4.5 Perfect Address Filtering

The MAC includes four programmable perfect address filters, as well as the all ones filter for broadcast frames. The RXCtl register is used to control whether a particular filter is used. The filters themselves share the same address space and the value in the Address Filter Pointer register determines which filter is being accessed at any time. The filters are arranged such that the first is the normal MAC address for the interface, which is also used in the detection of remote wake-up frames, and may be optionally used to detect pause (flow control) frames. The primary purpose of the second filter is for the recognition of pause frames. This would normally be programmed to correspond to the multicast address used for MAC control frames. The third and fourth filters, provide extra optional address match capabilities, which can provide the capability of adding extra individual addresses or of providing two multicast address filters.

Copyright 2007 Cirrus Logic

9

9.1.4.6 Hash Filter

The 64 bit Logical Address Filter provides DA filtering hashed by the CRC logic. The Logical Address Filter is sometimes referred to as the multicast address filter.

Referring to Figure 9-6, notice that the CRC computation starts at the first bit of the frame, which is also the first bit of the DA. (Recall that a "frame" is a "packet" without the preamble.)

The CRC Logic can be viewed as a 32 bit shift register with specific Exclusive-OR feedback taps. After the entire DA has been shifted into the CRC Logic, the signal HashLat latches the 6 most significant bits of the CRC Logic (x^{26} through x^{31}) into the 6-bit hash register (HR). The contents of HR are passed through the 6-bit to 64-bit Decoder. Only one of the 64 Decoder outputs is asserted at a time. That asserted output is compared with a corresponding bit in the Logical Address Filter. The filter output, Hashed, is used to determine if the received DA passed the hash filter. When set, the Hashed event bit shows that the received DA passed the hash filter. When clear, Hashed shows the failure of the DA to pass the hash filter.

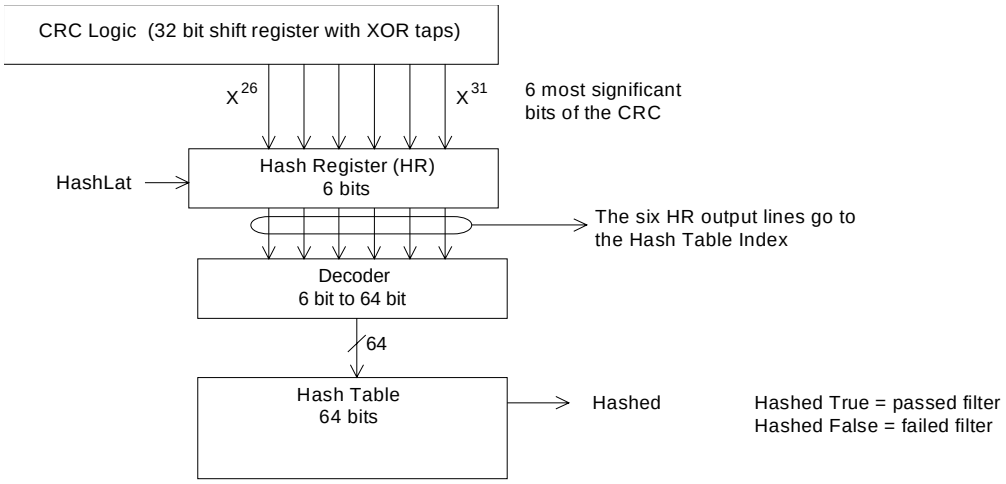


Figure 9-6. CRC Logic

Whenever the hashed filter is passed on good frames, the output of the HR is presented on the Hash Table Index (RStatQ). A received good frame is determined to be one without CRC error and which is the correct length (64 ≤ length ≤ 1518).

If RXCtl.MA is set, then any received multicast frame passing the hash filter is accepted. A multicast frame is one which has RXCtl.IA[0] = 1.

If RXCtl.IAHA[0] is set, then a frame with any individual address frame AND passing the hash filter is accepted. An individual address frame is one which has RXCtl.IA[0] = 0. For a frame to pass RXCtl.IAHA[0] it must have RXCtl.IA[0] = 0 and pass the hash.

Copyright 2007 Cirrus Logic

The relationship of RXCtl.MA and RXCtl.IAHA is shown below.

Table 9-2. RXCtl.MA and RXCtl.IAHA[0] Relationships

RXCtl.MA	RXCtl.IAHA[0]	Hash Filter Acceptance Results
0	0	Hash filter not used in acceptance criteria.
1	0	All multicast frames (first bit of DA = 1) passing the hash are accepted.
0	1	All individual address frames (first bit of DA = 0) passing the hash are accepted.
1	1	All frames that pass the hash are accepted.

9.1.4.7 Flow Control

The MAC provides special support for flow control by the transmission and reception of pause frames. A pause frame is a specific format of a MAC control frame that defines an amount of time for a transmitter to stop sending frames. Sending pause frames thereby reduces the amount of data sent by the remote station.

9.1.4.8 Receive Flow Control

The MAC can detect receive pause frames and automatically stop the transmitter for the appropriate period of time. To be interpreted as a pause frame the following conditions must be met:

- Destination address accepted by one of the first two individual address filters, with the appropriate RXCtl.RxFCE bit set.
- The Type field must match that programmed in the Flow Control Format register.
- The next two bytes of the frame (MAC Control Opcode) must equal 0x0001.
- The frame must be of legal length with a good CRC.

If accepted as a pause frame, the pause time field is transferred to the Flow Control Timer register. The pause frame may be optionally passed on to the Host or discarded by the MAC. Once the Flow Control Timer is set to a non-zero value, no new transmit frames are started, until the count reaches zero. The counter is decremented once every slot time while no frame is being transmitted.

9.1.4.9 Transmit Flow Control

When receive congestion is detected, the driver may want to transmit a pause frame to the remote station to create time for the local receiver to free resources. As there may be many frames queued in the transmitter, and there is a chance that the local transmitter is itself being paused, an alternative method is provided to allow a pause frame to be transmitted. Setting the Send Pause bit in the Transmit Control register causes a pause frame to be transmitted at the earliest opportunity. This occurs either immediately, or following the completion of the current transmit frame. If the local transmitter is paused, the pause frame will still be sent, and the pause timer will still be decremented during the frame transmission.

Copyright 2007 Cirrus Logic

To comply with the standard, pause frames should only be sent on full duplex links. The MAC does not enforce this, it is left to the driver. If a pause frame is sent on a half duplex link, it is subject to the normal half duplex collisions rules and retry attempts.

The format of a transmit pause frame is:

Bytes 1-6 - Destination address - this is the last Individual address (Address Filter Pointer = 6)

Bytes 7-12 - Source address - this is the first Individual address (Address Filter Pointer = 0)

Bytes 13-14 - Type field - this is defined in the Flow Control Format register

Bytes 15-16 - Opcode - set to 0x0001

Bytes 17-18 - Pause time - this is defined in the Flow Control Format register

Once the Host sets the Send Pause bit in TXCtl, it will remain set until the pause frame starts transmission. Then the Send Pause clears and the Pause Busy bit is set and remains set until the transmission is complete. No end of frame status is generated for pause frames.

9.1.4.10 Rx Missed and Tx Collision Counters

There are three counters that help the software in recording events, transmit collisions, receive missed frames, and receive runt frames. All three counters operate in similar ways. When the appropriate events occur the counters are incremented. They are cleared following a read of the count value. If a count is incremented such that the MSB is set, the corresponding status bit in the Interrupt Status Register is set. An interrupt is generated at this time if the corresponding enable bit is set in the Interrupt Enable Register. Once the count is incremented to an all ones condition it will not be incremented further, it will remain in this state until reset by a read operation.

9.1.4.11 Accessing the MII

This section describes the proper method to access the MII. It includes how to read/write PHY registers, how to have the PHY perform auto-negotiation, and how to startup the PHY.

The bits MDCDIV in register SelfCtl are used to control the PHY's clock divisor. The default value is 0x07, so the MDC clock frequency is HCLK divided by 8. This default value is correct for most PHYs. However, to be safe, check the PHY's data sheet to make sure that this clock frequency is correct.

The bit PSPRS in register SelfCtl is used to disable/enable Preamble Suppress for data passed from the MAC to the PHY through the MDIO. If bit PSPRS is set, the preamble is suppressed. In this case, the MAC won't prepend 32 bits of “1” to the data written to the PHY. Since the MAC automatically prepends the preamble to data when in transmission mode, bit PSPRS must be set while the MAC is transmitting frames. Otherwise, two preambles will be prepended and cause a transmission failure. The default value of “1” is appropriate for transmitting frames.

The MAC won't automatically prepend a preamble when not in transmission mode. Therefore, if the MAC wants to read/write PHY registers, bit PSPRS may be cleared since

Copyright 2007 Cirrus Logic

most PHYs require a preamble for access to the PHY's registers. However, to be safe, check PHY's data sheet to determine if a preamble is needed to read/write PHY registers.

9.1.4.11.1 Steps for Reading From the PHY Registers.

1. Read the value from the SelfCtl Register.
2. Since most PHYs need a Preamble for the MAC to read/write the PHY registers, you may need to clear the PreambleSuppress bit.
3. Ensure that the PHY is not busy by polling the MIIStatus_Busy Bit in MIIStatus register.
4. Issue the command to read the register within the PHY.
5. Wait until the read command is completed. Determine this by polling the MIIStatus_Busy bit in MIIStatus register.
6. Get the PHY data from the MII Data register.
7. Restore the old value to SelfCtl register.

Note: Steps 1, 2, and 7 are not required if the PHY doesn't need a preamble for access to the PHY's registers.

9.1.4.11.2 Steps for Writing To the PHY Registers.

1. Read the value from SelfCtl register.
2. Since most PHYs need a Preamble for the MAC to read/write the PHY registers, you may need to clear the PreambleSuppress bit.
3. Ensure that the PHY is not busy by polling the MIIStatus_Busy bit in MIIStatus register.
4. Put the PHY data into the PHY Data register
5. Issue the write command to write data to the register within the PHY
6. Wait until the write command is completed. Determine this by polling the MIIStatus_Busy Bit in MIIStatus Register.
7. Restore the old value to SelfCtl register.

Note: Steps 1, 2, and 7 are not required if the PHY doesn't need a preamble for access to the PHY's registers.

9.1.4.11.3 Steps for PHY Auto-negotiation

1. Write to the Auto-Negotiation Advertisement register (0x04). Set it in accordance with IEEE_802.3 standard, and advertise 100/10M full/half duplex available.
2. Write to Basic Mode Control Register (0x00), to enable and restart Auto-Negotiation.
3. Poll bit Auto_Neg_Complete in the BMSR register in the PHY until the Auto-Negotiation is complete.

Copyright 2007 Cirrus Logic

9.1.4.11.4 Steps for PHY Startup

1. Set the MDC ClockDivisor and the PreambleSuppress for the PHY in the SelfCtl register. The default value 0x0000_0F10 is appropriate for most PHYs in transmission mode.
2. Have the PHY perform auto-negotiation.
3. Read the Auto-Negotiation_Link_Partner_Ability register to check the PHY's configuration.
4. If the link is Full Duplex, then set MAC for Full Duplex.

9.2 Descriptor Processor

The MAC operates as a bus master to transfer all receive and transmit, data and status, across the AHB bus. The transfers are managed by two sets of queues for each direction, a descriptor queue and a status queue. The following section details the operation of these queues.

9.2.1 Receive Descriptor Processor Queues

The Receive Descriptor Processor uses two circular queues in Host memory to manage the transfer of receive data frames. The receive descriptor queue is used to pass descriptors of free data buffers from the Host to the MAC. The receive status queue is used to pass information on the MAC's use of the data buffers back to the Host. Keeping these queues separate enables the use of burst transfers to and from the queues, reducing the overall amount of bus traffic and avoiding some potential latency problems.

9.2.2 Receive Descriptor Queue

The receive descriptors are passed from the Host to the MAC via the receive descriptor queue. The receive descriptor queue is a circular queue occupying a contiguous area of memory. The location and size of the queue are set at initialization writing to the Receive Descriptor Queue Base Address Register, the Receive descriptor current address, and the Receive Descriptor Queue Base Length. The base address must point to a word-aligned memory location. The Current Address must be set to point to the first descriptor to be used. This would normally be the first entry (same value as the base address). The Receive Descriptor Queue Base Length is set to the length (in bytes) of the queue. The number of descriptors should be an integral power-of-two (2, 4, 8, 16, etc.). Otherwise the Receive Descriptor Processor may not work properly and the MAC/Ethernet may stop receiving frames.

Each descriptor entry defines one receive data buffer, and consists of two words. The first word contains the address of the data buffer, which must be word aligned. The second word contains three fields: buffer length, buffer index and a Not Start Of Frame bit. The buffer length field specifies the maximum number of bytes to be used in the buffer and should be an integral number of words. If the buffer length is set to zero, the descriptor will be ignored, and no status will be posted for the buffer. The buffer index can be used by the Host to keep track

Copyright 2007 Cirrus Logic

9

of buffers as they are exchanged with the MAC. When the MAC reads a descriptor, it keeps a copy of the index, which it includes in any status entry associated with that buffer. The Not Start Of Frame bit may be set by the Host on any buffer in which it does not want a new frame to be started. This buffer would then only be used for chaining of frame fragments. This mode may be used to align frames on boundaries coarser than descriptors, such as when multiple physical address descriptors are used to describe one virtual address buffer.

In normal operation, the Host does not need to access the RXDQBAdd, RXDQBLen, RXDCurAdd registers following initialization. Control of the use of the descriptors is handled using the Receive Descriptor Enqueue register (RXDEnq). The term enqueue refers to the action of adding descriptors to the end of an existing queue. To enqueue receive descriptors, the Host writes the number of descriptors to the RXDEnq register. The number is automatically added to the existing value. When the MAC consumes descriptors by reading them into its on local storage (internal MAC buffer), the number read is subtracted from the total. The Host can read the total number of unread valid descriptors left in the queue from the RXDEnq. There is a restriction that no more than 255 descriptors may be added to the queue in one write operation. To add more than this number requires multiple write operations. See Figure 9-7.

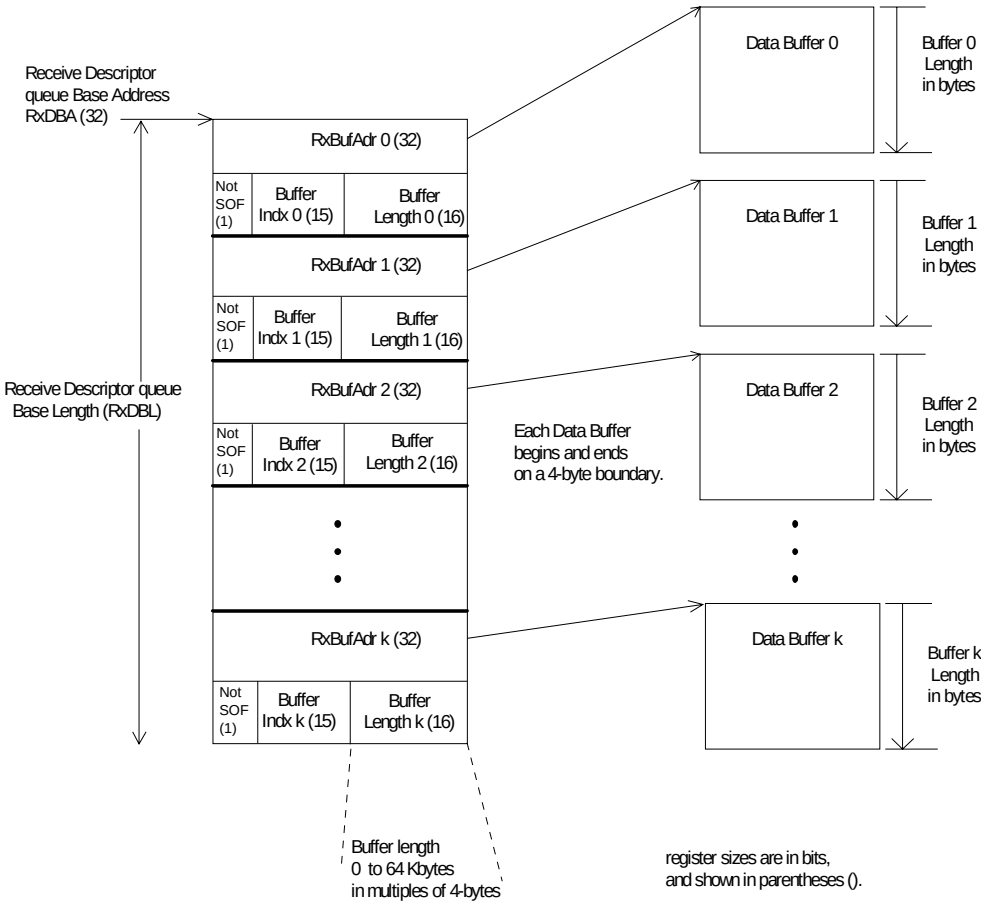


Figure 9-7. Receive Descriptor Format and Data Fragments

Copyright 2007 Cirrus Logic

Receive Descriptor Format - First Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BA															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BA															

Definition:

Receive Descriptor, first word. Contains the base address to the data buffer.

Bit Descriptions:

BA: Buffer Address. This location holds the 32 bit address pointer to the data buffer, this must point to a word aligned location.

Receive Descriptor Format - Second Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSOF	BI														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BL															

Definition:

Receive Descriptor, second word. Contains control, index and length for the descriptor.

Bit Descriptions:

NSOF: Not Start of Frame. When the Not Start Of Frame bit is set in a descriptor, the associated buffer will only be used for a frame being continued from another buffer. If there is not a frame to be continued (that is, start of a new frame), the buffer will be discarded. When a buffer is discarded in this manner, there is no status posted.

BI: Buffer Index. The buffer index is provided for Host software purposes. The MAC keeps an internal copy of the index and includes it with any status writes associated with a receive buffer.

BL: Buffer Length. The Buffer Length contains the number of bytes available to be used in the receive buffer. This should be an integral number of words. If the length is set to zero, the descriptor will be ignored and no status will be posted for the buffer.

Copyright 2007 Cirrus Logic

9.2.3 Receive Status Queue

The receive status queue is used to pass receive status from the MAC to the Host. In operation, the receive status queue is similar to the receive descriptor queue. It is a circular queue in contiguous memory space. The location and size of the queue are set at initialization by writing to the Receive Status Queue Base Address and the Receive Status Queue Base Length registers. The base address must point to a word aligned memory location. The length is set to the actual status queue length (in bytes) and should not exceed 64 Kbytes total. The number of status entries should be an integral power-of-two (2, 4, 8, 16, etc.), or the Receive Descriptor Processor may not work properly, and the MAC/Ethernet may stop receiving frames. The Current Address must be set to point to the first status entry to be used. This would normally be the first entry (same value as the base address).

When the receive status queue initialization is complete, the Receive Status Enqueue register is used by the Host to pass free status locations to the MAC. To simplify this process the Host writes the number of additional free status locations available to the enqueue register. The MAC adds the additional count to the previously available location to determine the total number of available receive status entries. When the MAC writes status to the queue, it subtracts the number written from this total. The current value of the total receive status entries is available by reading the enqueue register.

No more than 255 status entries may be added in one write. If a number greater than this needs to be written, the write should be broken up into more than one operation (that is, to add 520 status entries: write 255, then write 255, finally write 10).

Copyright 2007 Cirrus Logic

9

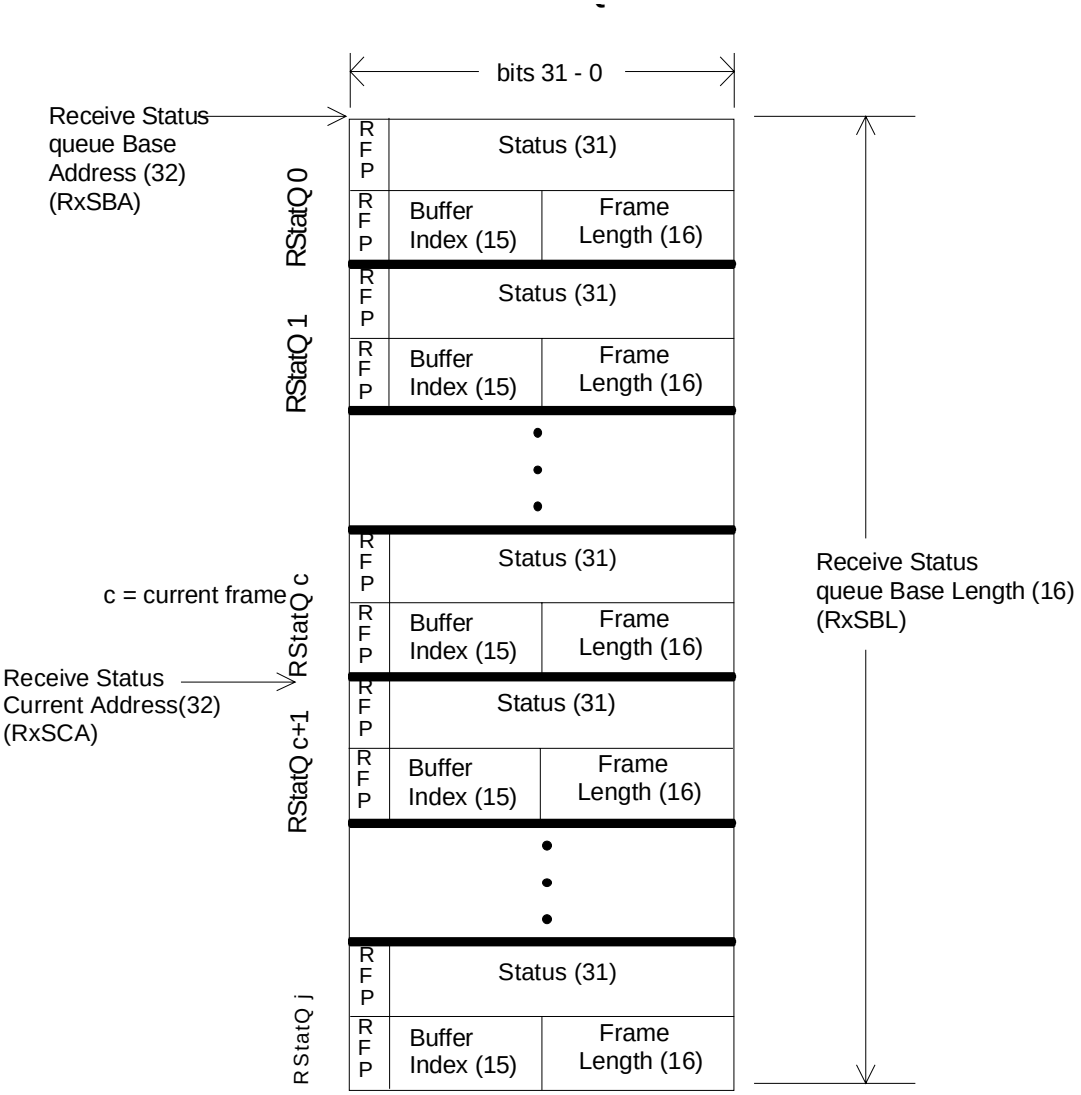


Figure 9-8. Receive Status Queue

Receive status entries are written to the status queue following one of three possible events, end of header, end of buffer, or end of frame. The status event is always written after the appropriate data transfer has been made. For example the end of frame status is written after the last byte of data has been written to the data buffer, not before. The EOF and EOB bits in the status entry can be used to determine the cause of a status entry.

Copyright 2007 Cirrus Logic

9

If both EOF and EOB bits are zero, the entry was made for a receive header threshold. This indicates that there have been at least as many bytes transferred as specified in Receive Header Length 1 or 2. These registers may be set to any threshold to provide an early indication to the Host that a receive frame is in progress. The status will contain valid data in the address match and hash table fields, but as the status is provided before end of frame is reached, it will always indicate received without error.

If the EOF bit is zero and the EOB bit is set, the status indicates that the end of a receive buffer has been reached before the end of the receive frame. If the receive buffers are much smaller than the frame size, there may be many such statuses per frame.

When the EOF and EOB bits are both set, the status indicates the end of frame has been transferred. The EOB is always set at this time to indicate that the MAC has finished transferring to the buffer. The buffer is not necessarily full.

When a status event causes an interrupt, the interrupt pin will be activated after the status has been transferred to the status queue.

9.2.3.1 Receive Status Format

Receive Status - First Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RFP	**RWE**	**EOF**	**EOB**	**RSVD**				**AM**		**RX_Err**	**OE**	**FE**	**Runt**	**EData**	**CRCE**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCI	**RSVD**	**HTI**						**RSVD**							

Definition:

Receive Status, first word. Contains status information for the receiver operation.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RFP: Receive Frame Processed. The Receive Frame Processed bit is always written as a "1" by the MAC when the status is ready and it may be used by the Host to mark its progress through the status queue. The Host may alternatively use the RXStsQCurAdd to determine how much of the status queue to process.

RWE: Received Without Error. The Received Without Error bit indicates that the frame was received without any of the following error conditions: CRCerror, ExtraData, Runt, or Receive Overrun.

Copyright 2007 Cirrus Logic

EOF:	End Of Frame. When this bit is set, the associated buffer contains the last data associated with this frame. In the case of an extra data or overrun error, the buffer may not contain the actual end of frame data. For a receive header status the EOF and EOB bits will both be clear.
EOB:	End Of Buffer. When this bit is set, no more data will be transferred to the associated data buffer. This may be due to an end of frame transfer or to reaching the actual end of the buffer. For a receive header status the EOF and EOB bits will both be clear.
AM:	Address Match: 00 - Individual Address match 01 - Global Address match 10 - Hashed Individual Address 11 - Hashed Multicast Address
RX_Err:	RX Error. The RX_Err is set for any receive frame for which the RX_ERR (MII pin) was activated.
OE:	Overrun Error. The receive overrun bit is set on any frame which could not be completely transferred to system memory. This could be as a result of insufficient buffer space, or an excessive bus arbitration time.
FE:	Framing Error. This bit is set for any frame not having an integral number of bytes, and received with a bad CRC value.
Runt:	Runt Frame. The Runt bit is set for any receive frame, including CRC, that is shorter than 64 bytes.
EData:	Extra Data. The ExtraData bit indicates that the length of the incoming frame was equal or greater than the value programmed in the Max Frame Len register. The receive frame will be terminated at this maximum length to conserve system buffer space.
CRCE:	CRC Error. This indicates the frame was received with a bad CRC.
CRCI:	CRC Included. This bit is set to one when the CRC has been included in the Receive data buffer. Including or excluding the CRC is controlled by the BufferCRC bit in the RXCtl register.

Copyright 2007 Cirrus Logic

HTI: Hash Table Index. If the frame was accepted as a result of a hash table match, these bits contain the hash table index, otherwise they are written as zero. If the frame was received as a result of Promiscuous Accept, this field will be zero. If the frame was accepted as a result of an Individual Address Match then the field indicates which address was matched, as follows:

000001 - Frame matched Individual Address 0
000010 - Frame matched Individual Address 1
000100 - Frame matched Individual Address 2
001000 - Frame matched Individual Address 3

Receive Status - Second Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RFP	BI														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FL															

Definition:

Receive Status, second word. Contains status information for the receiver operation.

Bit Descriptions:

RFP: Receive Frame Processed. The Receive Frame Processed bit is always written as a 1 by the MAC when the status is ready and it may be used by the Host to mark its progress through the status queue.

BI: Buffer Index. This field contains the buffer index field from the descriptor table for the data buffer associated with this status entry.

FL: Frame Length. The frame length field contains the total number of bytes transferred for this frame. For an intermediate status (not end of frame) this is the total number of bytes transferred up through the current data buffer.

Copyright 2007 Cirrus Logic

9.2.3.2 Receive Flow

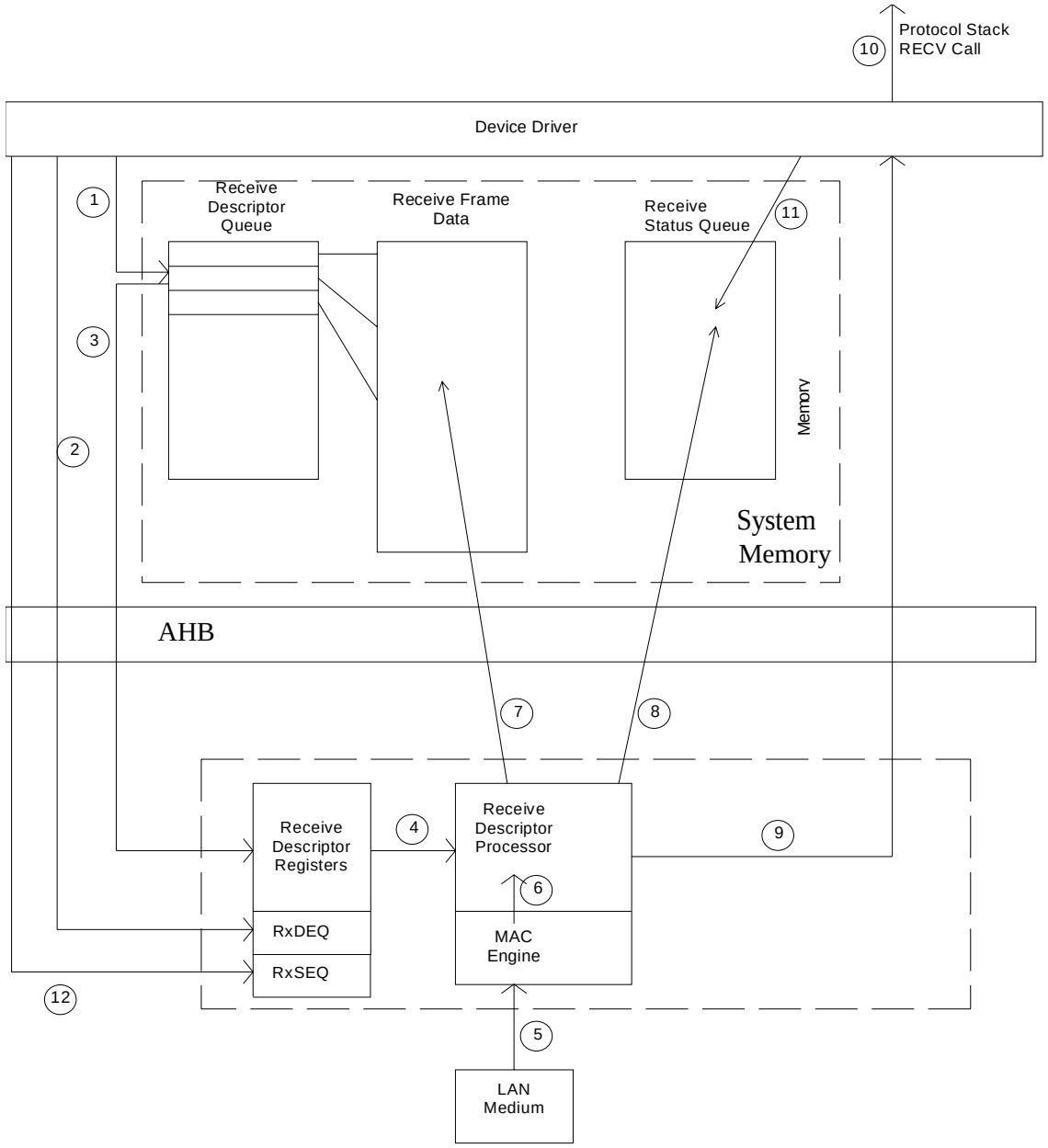


Figure 9-9. Receive Flow Diagram

Copyright 2007 Cirrus Logic

Refer to the circled numbers in Figure 9-9. The detailed receive flow is:

1. Driver initializes some number of receive descriptors.
2. Driver writes RXDEnq register with the additional number of receive descriptors.
3. On-chip Descriptor Processor fetches descriptors into internal FIFO decrements RXDEnq appropriately.
4. The address of the next receive data buffer is loaded into the Receive Buffer Current Address.
5. A frame is received from the LAN medium.
6. The MAC Engine passes the frame data to the Receive Data FIFO.
7. The Receive Descriptor Processor stores the frame data into system memory.

Note: Steps 5, 6, and 7 can overlap.

8. End of frame status is written to the Receive Status Queue the RXStsEnq value reduced by one.
9. Driver interrupted if interrupt conditions met.
10. Received frame passed to the protocol stack.
11. Driver clears the Receive Frame Processed bit in Status Queue.
12. Driver writes number of entries processed in the status queue, freeing them for future use by the MAC.
13. After the driver gets the used receive buffers back from the stack, the driver may repeat step 2.

Note: Steps 1, 11, and 13 are transparent to the MAC. Steps 2 through 10 and 12 directly involve the MAC.

9.2.3.3 Receive Errors

Receive error conditions are broken into two categories: hard errors and soft errors. A hard error is generally considered a reliability problem. This includes AHB bus access problems. A soft error indicates that the frame was not successfully received. The error may be expected or rare. A soft error needs a graceful recovery by the host driver. Soft errors include: CRC errors, receiver over-run, frames too long, or frames too short. Hard errors are parity errors (when enabled), system errors, and master or target aborts, these errors will stop receive DMA activity, and require host intervention for recovery. Recovery may be achieved by performing a RxChRes (Bus Master Control) and reinitializing.

Copyright 2007 Cirrus Logic

9.2.3.4 Receive Descriptor Data/Status Flow

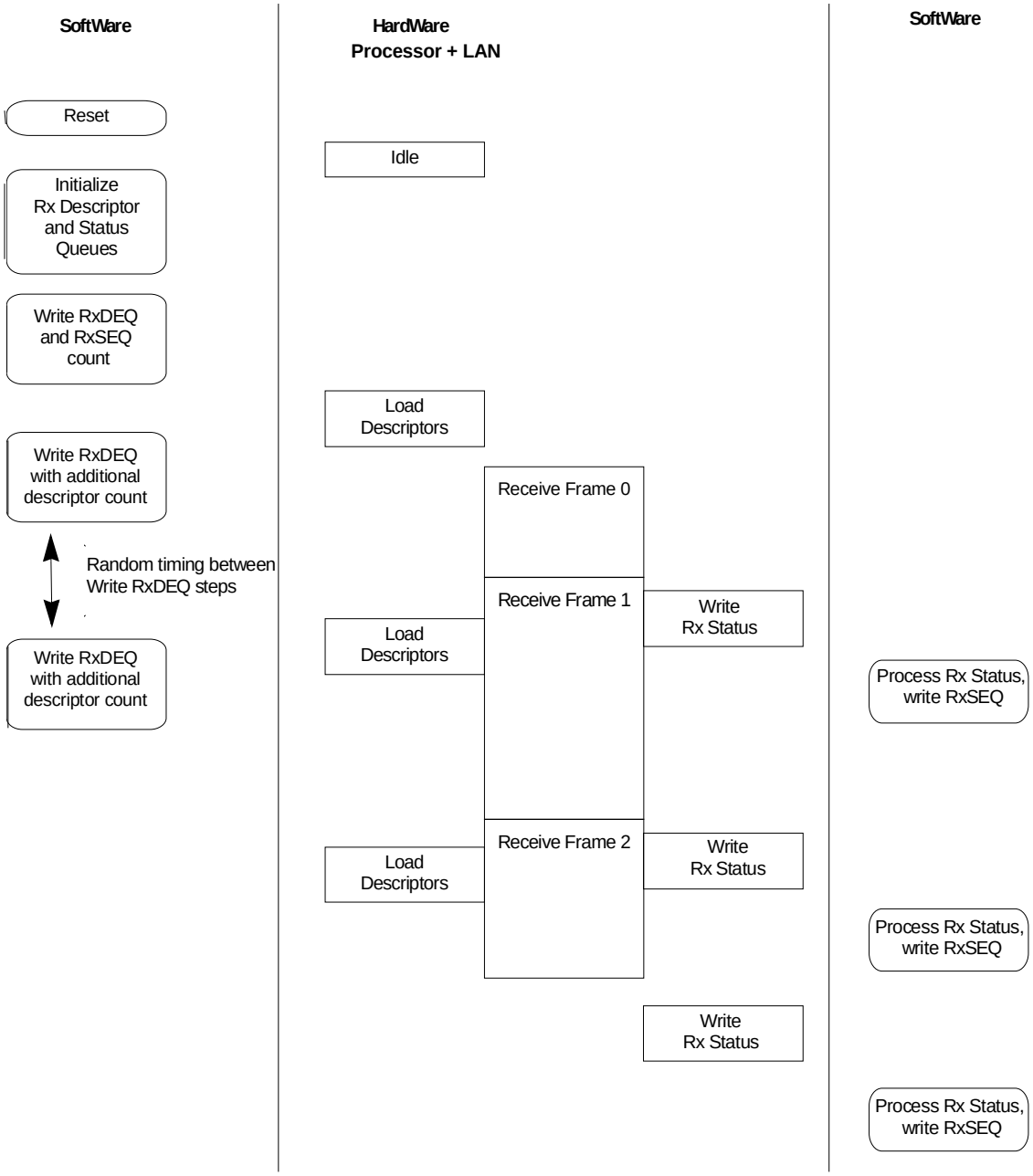


Figure 9-10. Receive Descriptor Data/Status Flow

Copyright 2007 Cirrus Logic

9.2.3.5 Receive Descriptor Example

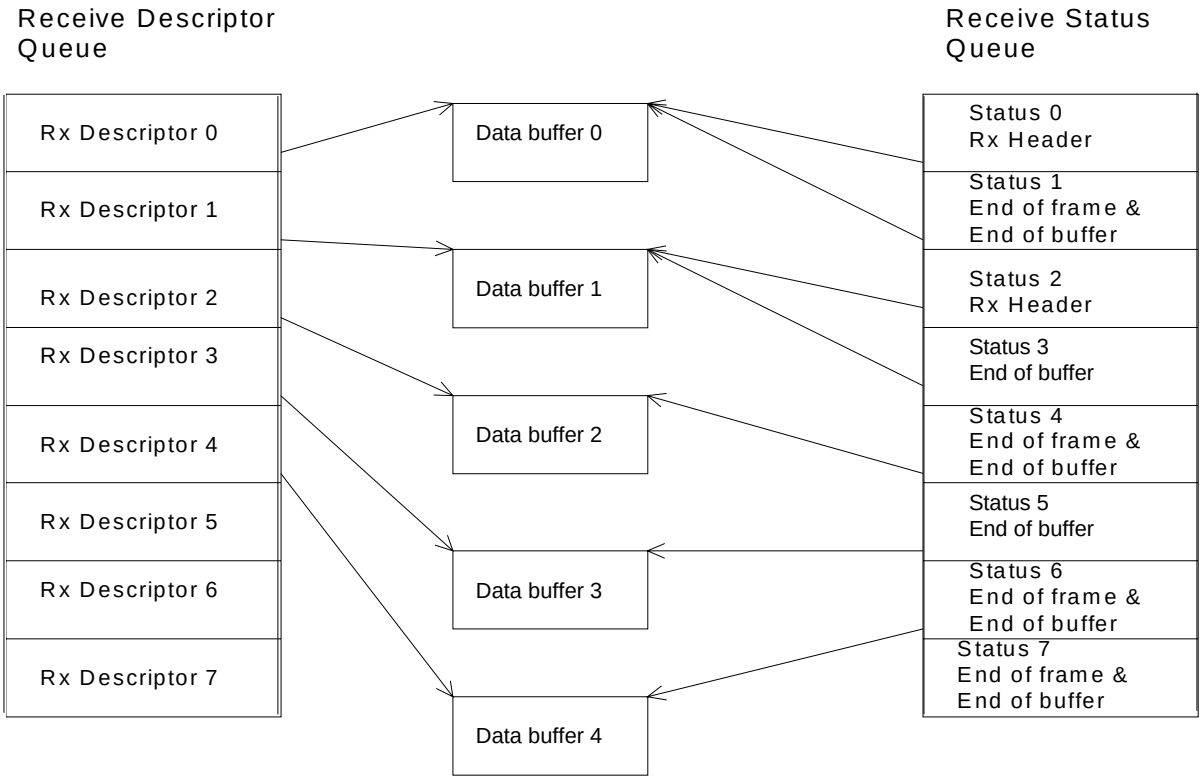


Figure 9-11. Receive Descriptor Example

Figure 9-11 shows the state of the receive queues following the reception of four frames. The first frame uses Data buffer 0 only and there are two status entries associated with it. The first status (status 0) is for the reception of a receive header and the second (status 1) is for the end of frame/buffer, both status entries point to the beginning of data buffer 0. The second frame occupies two buffers (data buffers 1 and 2), and three status entries (2, 3, and 4). Status 2 is for the receive header, status 3 for the end of buffer 1 (frame size larger than buffer size), and status 4 for end of frame/buffer. The next two frames both occupy one data buffer each and one status each. This could be the case for short frames that do not exceed the header size or the buffer size. The result of this is that the status queue may be used at a different rate to the descriptor queue, based on the type of traffic and the options selected.

Copyright 2007 Cirrus Logic

9

9.2.3.6 Receive Frame Pre-Processing

The MAC pre-processes all incoming receive frames. First the frame is either passed on to the next level or discarded according to the destination address filter. The next decision is whether to accept the frame. A frame is accepted when the frame data are brought into MAC through internal memory. The final step in frame pre-processing is the decision on causing an interrupt. These pre-processing steps are detailed in Figure 9-12.

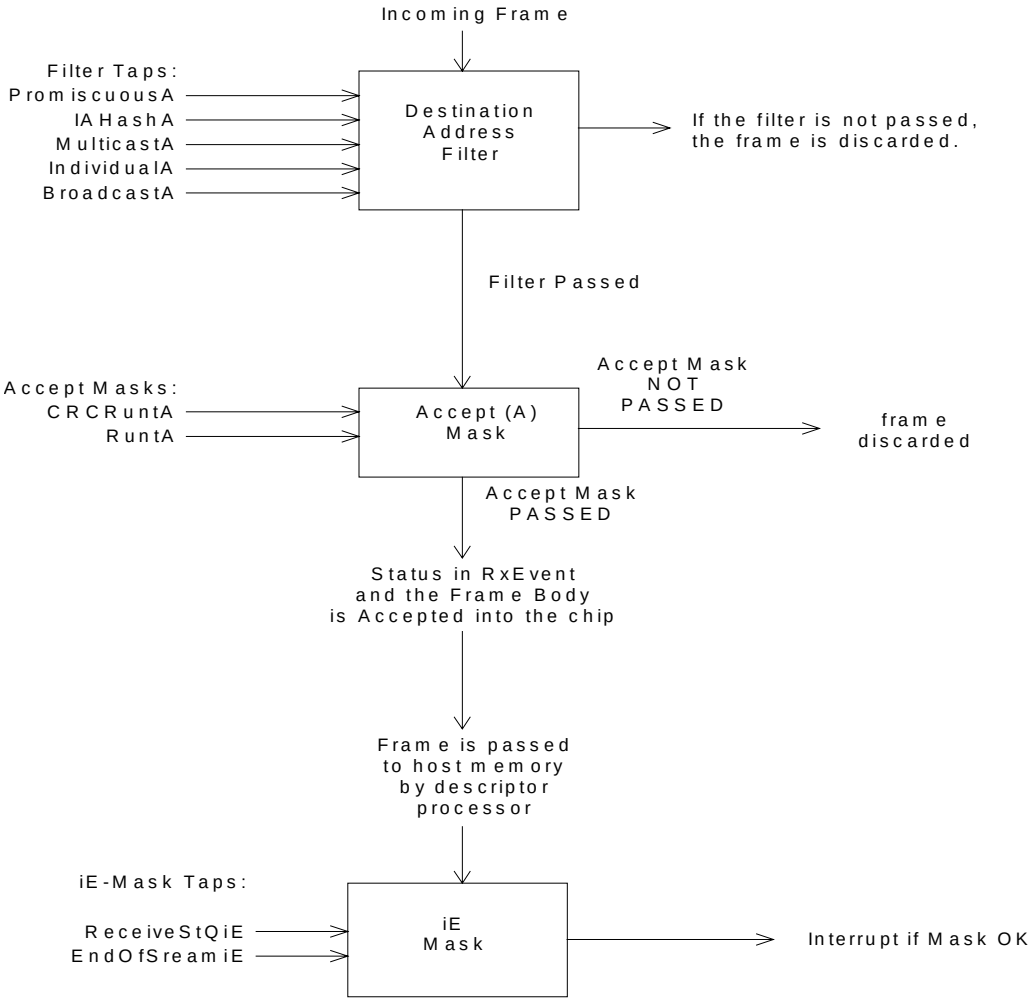


Figure 9-12. Receive Frame Pre-processing

Copyright 2007 Cirrus Logic

9.2.3.7 Transmit Descriptor Processor Queues

The transmit descriptor processor uses two circular queues in Host memory to manage the transfer of transmit data frame. The transmit descriptor queue is used to pass descriptors of user's data buffers from the Host to the MAC. The transmit status queue is used to pass information on the MAC's use of the data buffer back to the Host. Keeping these queues separate enables the use of burst transfers to and from the queues, reducing the overall amount of bus traffic and avoiding some potential latency problem.

9.2.3.8 Transmit Descriptor Queue

The Transmit descriptors are passed from the Host to the MAC via the Transmit descriptor queue. The Transmit descriptor queue is a circular queue occupying a contiguous area of memory. The location and size of the queue are set at initialization by the Host writing to the Transmit Descriptor Queue Base Address Register and the Transmit Descriptor Queue Base Length. The base address must point to a word aligned memory location. The Transmit Descriptor Queue Base Length is set to the length in bytes of the queue. The length should be an integral number of descriptors and must not exceed 64 Kbytes total. The Transmit descriptor current address must also be set at initialization to point to the first descriptor to be used. This would normally be the first entry (same value as the base address).

Following initialization, the MAC will start to use descriptors from the Current Descriptor Address, wrapping back to the base pointer whenever the end of the queue is reached. In normal operation the Host should not need to access these registers after the initialization. The management of the descriptors is handled via the Transmit Descriptor Enqueue register.

Enqueueing descriptors is the process of adding descriptors to an existing queue. This is achieved in transmit by writing the number of additional descriptors to the Transmit Descriptor Enqueue register. The written value will be added to the previous value to keep a running total, as descriptors are read by the MAC, the total is adjusted. The running total is available by reading the enqueue register. One frame may be described by more than one descriptor, but the final descriptor will contain the EOF bit. Not all the descriptors for a frame need to be supplied at once.

No more than 255 descriptors may be added in one write. If a number greater than this needs to be written. the write should be broken up into more than one operation (that is, to add 300 descriptors - first write 255, then write 45).

Copyright 2007 Cirrus Logic

9

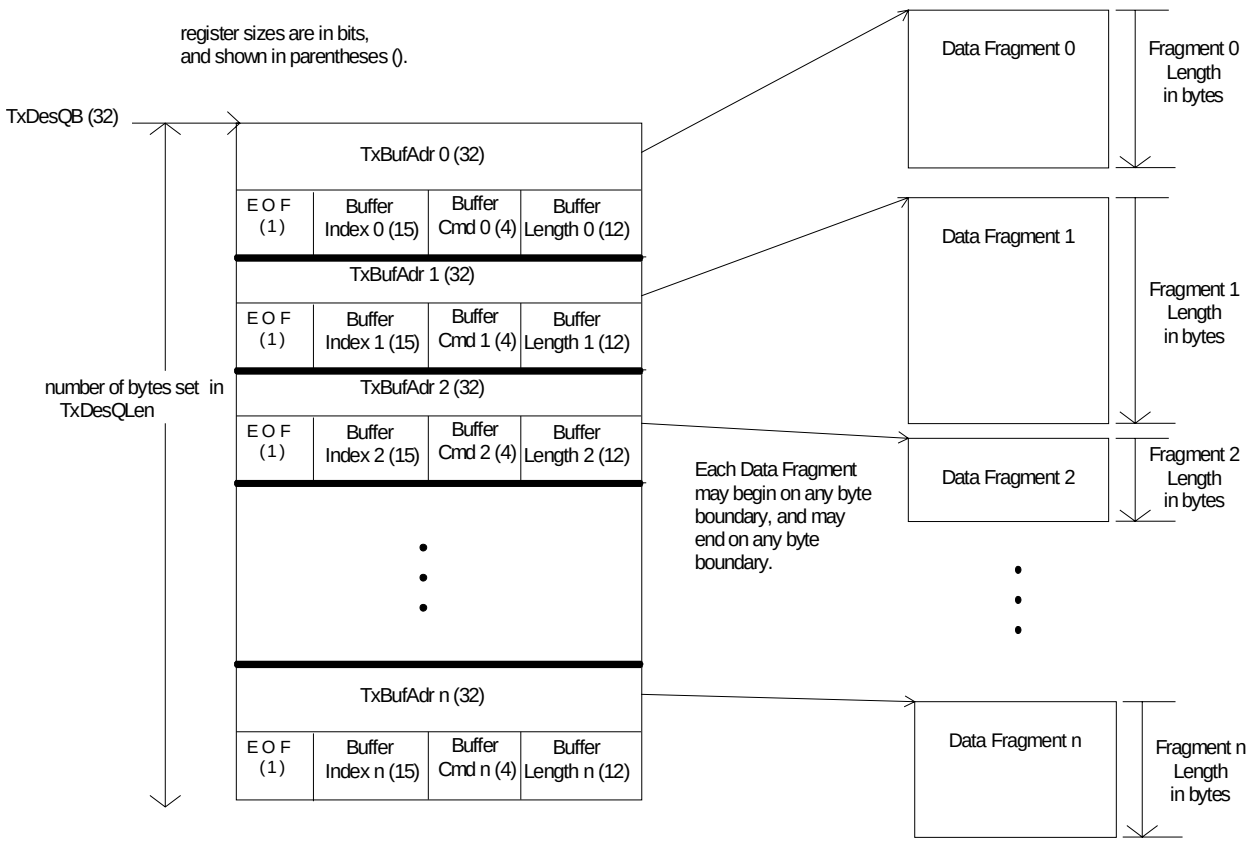


Figure 9-13. Transmit Descriptor Format and Data Fragments

Copyright 2007 Cirrus Logic

9

Example: Fragments 0, 1, 2 make-up one complete frame.

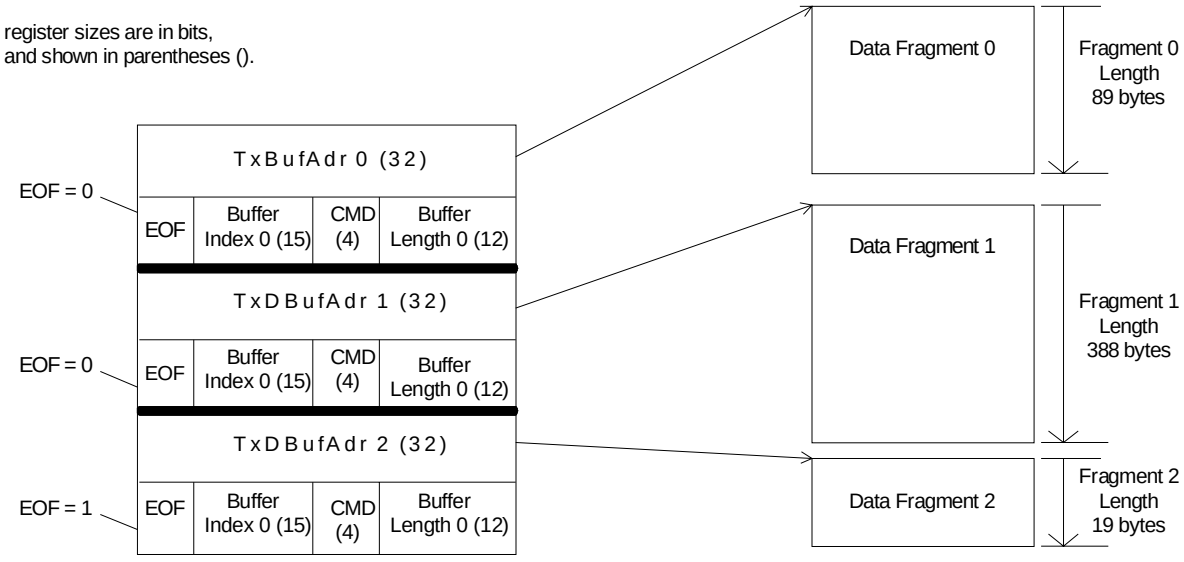


Figure 9-14. Multiple Fragments Per Transmit Frame

In the example shown in Figure 9-14, one frame is transmitted from three fragments. The MAC starts the frame by acquiring the medium and transmitting the preamble. Then, the fragments 0, 1, 2 are transmitted in order for a total of 446 bytes (39 + 388 + 19). Since the CRC bit in the first frame fragment is clear, the HW appends the 4 byte CRC. Thus, 4 more bytes are added to the frame for the CRC making the total frame length 450 bytes. Finally, the MAC sends the end-of-frame.

The CMD field is 4 bits. Only the AF bit is valid. The other fields are reserved.

9.2.3.9 Transmit Descriptor Format

Transmit Descriptor Format - First Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TBA															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBA															

Definition:

Transmit Descriptor, first word. Contains the base address of the data buffer.

Copyright 2007 Cirrus Logic

Bit Descriptions:

TBA: Transmit Buffer Address. The transmit buffer address contains the 32 bit address pointer to the transmit buffer. The base address of the data buffer must be word-aligned (32-bit aligned).

Transmit Descriptor Format - Second Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EOF	TBI														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AF	RSVD			TBL											

Definition:

Transmit Descriptor, second word. Contains control, index and length for the descriptor.

Bit Descriptions:

EOF: End of Frame. When this bit is set, the descriptor terminates a transmit frame. When clear, the descriptor is not the end of frame and a future descriptor will provide the EOF.

TBI: Transmit Buffer Index. The transmit buffer index is provided to help the Host software keep track of the transmit buffers. A copy of the index for the first buffer of a frame is kept in the MAC, and is included in any status written for the particular frame.

AF: Abort Frame. When the Abort Frame and EOF bits are set in a descriptor, the transmit frame will be terminated with a bad CRC. A bad CRC is applied even when the InhibitCRC bit (TXCtl) is set. The Abort Frame bit is ignored in a descriptor which does not have the EOF bit set. The abort feature is useful in a forwarding environment, where the integrity of the incoming frame is not known before the outgoing frame is started. If the incoming frame is received with error, the outgoing frame can be then invalidated. The AF bit is the only valid bit in the CMD field.

RSVD: Reserved. Unknown During Read.

Copyright 2007 Cirrus Logic

TBL: Transmit Buffer Length. This field contains the byte count of the number of bytes in the transmit buffer. There are no restrictions on the actual buffer size. If the length is set to zero, the descriptor will be ignored. A frame may not be terminated with a zero length buffer.

9.2.3.10 Transmit Status Queue

The Transmit Status queue is used to pass transmit status from the MAC to the Host. In operation the status queue is similar to the transmit descriptor queue, it is a circular queue in contiguous memory space. The location and size of the queue are set at initialization by the Host writing to the Transmit Status Queue Base Address, and the Transmit Status Queue Base Length registers. The base address must point to a word aligned memory location. The length is set to the actual status queue length in bytes. This should be an integral number of status entries and should not exceed 64 Kbytes total. The Current Address must be set to point to the first status entry to be used. This would normally be the first entry in the queue (same value as the base address).

The Host needs to ensure that in operation there is always room in the status queue for any transmit frame which is enqueued in the transmit descriptor queue.

Copyright 2007 Cirrus Logic

9

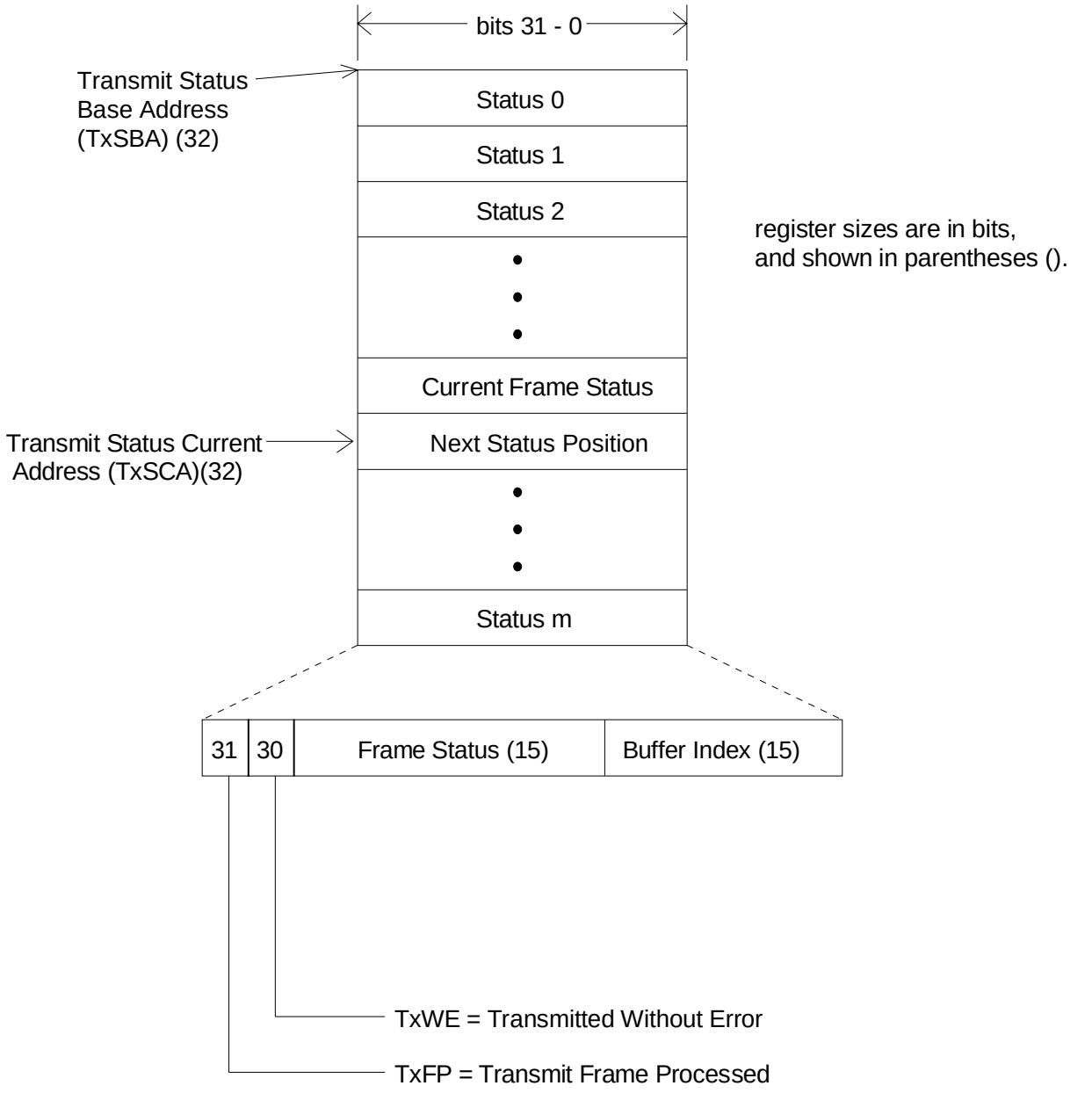


Figure 9-15. Transmit Status Queue

Copyright 2007 Cirrus Logic

9.2.3.11 Transmit Status Format

Only one Transmit Status entry is posted for each transmit frame, regardless of the number of transmit descriptors that are used to describe the frame.

Transmit Status

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TxFP	TxWE	FA	LCRS	RSVD	OW	TxU	EColl	RSVD				NColl			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	TBI														

Definition:

Transmit Status. Contains the status information for the transmitter operation.

Bit Descriptions:

TxFP: Transmit Frame Processed. The Transmit Frame Processed bit is always written as a 1 by the MAC when the status is ready and it may be used by the Host to mark its progress through the status queue.

TxWE: Transmitted Without Error. The transmitted Without Error bit is set when a frame is successfully transmitted without errors.

FA: Frame Abort. When a frame has been terminated by the Host with an Abort Frame command, in the transmit descriptor, the Frame Abort status bit is set.

LCRS: Loss of CRS. The Loss of CRS bit is set when a frame is transmitted and the MII CRS signal is not asserted at the end of preamble.

RSVD: Reserved. Unknown During Read.

OW: Out of Window. The Out of Window bit indicates that a collision was detected after the transmission of more than 60 bytes (from the start of preamble).

TxU: Transmit Underrun. TxUnderrun is set when a frame fails to be transmitted because of an excessive bus latency starving the transmitter.

EColl: Excess Collisions. The excessive collision bit is set when the frame failed to transmit due to excessive collisions. This may either be due to one or sixteen collisions dependent on the OneColl bit in the transmit descriptor.

Copyright 2007 Cirrus Logic

NColl: Number of Collisions. This field contains the number of collisions that were experienced in transmitting this frame.

TBI: Transmit Buffer Index. The transmit buffer index is a copy of the transmit buffer index from the first descriptor of a transmit frame. This is provided as an aid to the Host software in keeping track of the transmit buffers.

Copyright 2007 Cirrus Logic

9.2.3.12 Transmit Flow

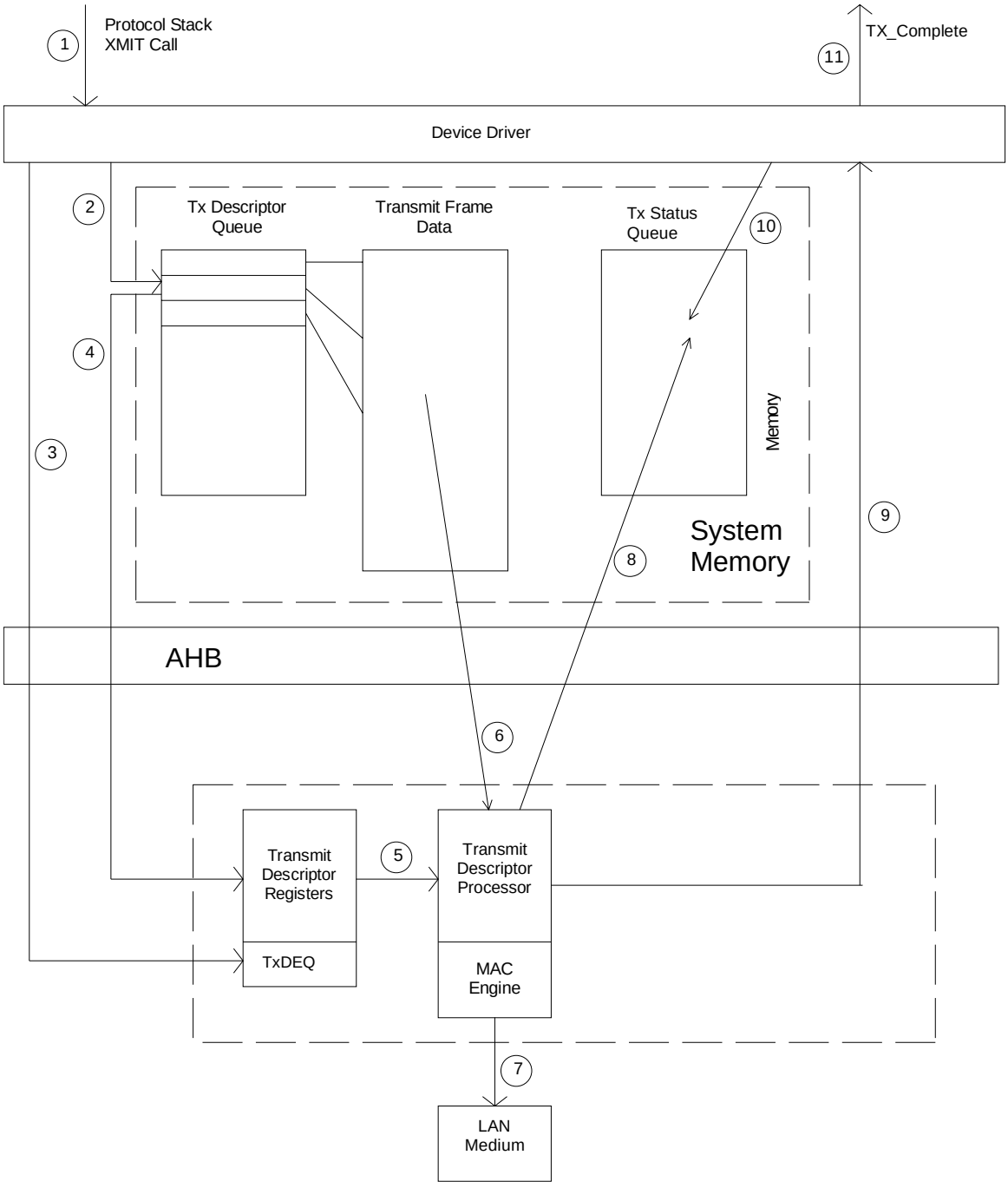


Figure 9-16. Transmit Flow Diagram

Copyright 2007 Cirrus Logic

Refer to Figure 9-16. The detailed transmit flow is:

1. Protocol stack initiates a transmit frame.
2. Driver parses protocol stack buffer into Transmit Descriptor Queue.
3. Driver writes number of additional entries to the Transmit Enqueue register.
4. On-chip Descriptor Processor fetches descriptor information.
5. On-chip Descriptor Processor initiates data move.
6. Frame data fetched from system memory into the transmit FIFO.
7. Frame transmitted onto LAN medium. Steps 6 and 7 can overlap.
8. End of frame status written to status queue
9. Driver interrupted if interrupt conditions met.
10. Driver processes the transmit status
11. Driver informs the protocol stack that transmit is complete.

Note: Steps 1, 2, 10, and 11 are transparent to the MAC block. Steps 3 through 9, inclusive, directly involve the MAC.

9.2.3.13 Transmit Errors

Transmit error conditions are broken into two categories: hard errors and soft errors. A hard error is generally considered a reliability problem. This includes AHB bus access problems. A soft error indicates that the frame was not successfully transmitted. The error may be expected or rare. A soft error needs a graceful recovery by the host driver. Soft errors include: excessive collisions, SQE error (if connected to a MAU). Hard errors are parity errors (if enabled), system errors, master and target aborts. These will stop further transmit DMA activity and require host intervention for recovery.

Hard errors cause the Descriptor Processor to halt operation. This allows the Host to determine the cause of error and reinitialize and restart the bus master operations.

Most soft errors do not cause the frame processing operations to halt. The Descriptor Processor simply flags the error and continues on to the next frame. The exception is on a transmit underrun. By halting the transmit frame processing, the Host has the ability to re-initialize the transmit Descriptor Processor registers to point to the start of the failed frame and re-initialize. This will cause the MAC to reattempt the failed frame and allows the order of frame transmission to be maintained.

Copyright 2007 Cirrus Logic

9.2.3.14 Transmit Descriptor Data/Status Flow

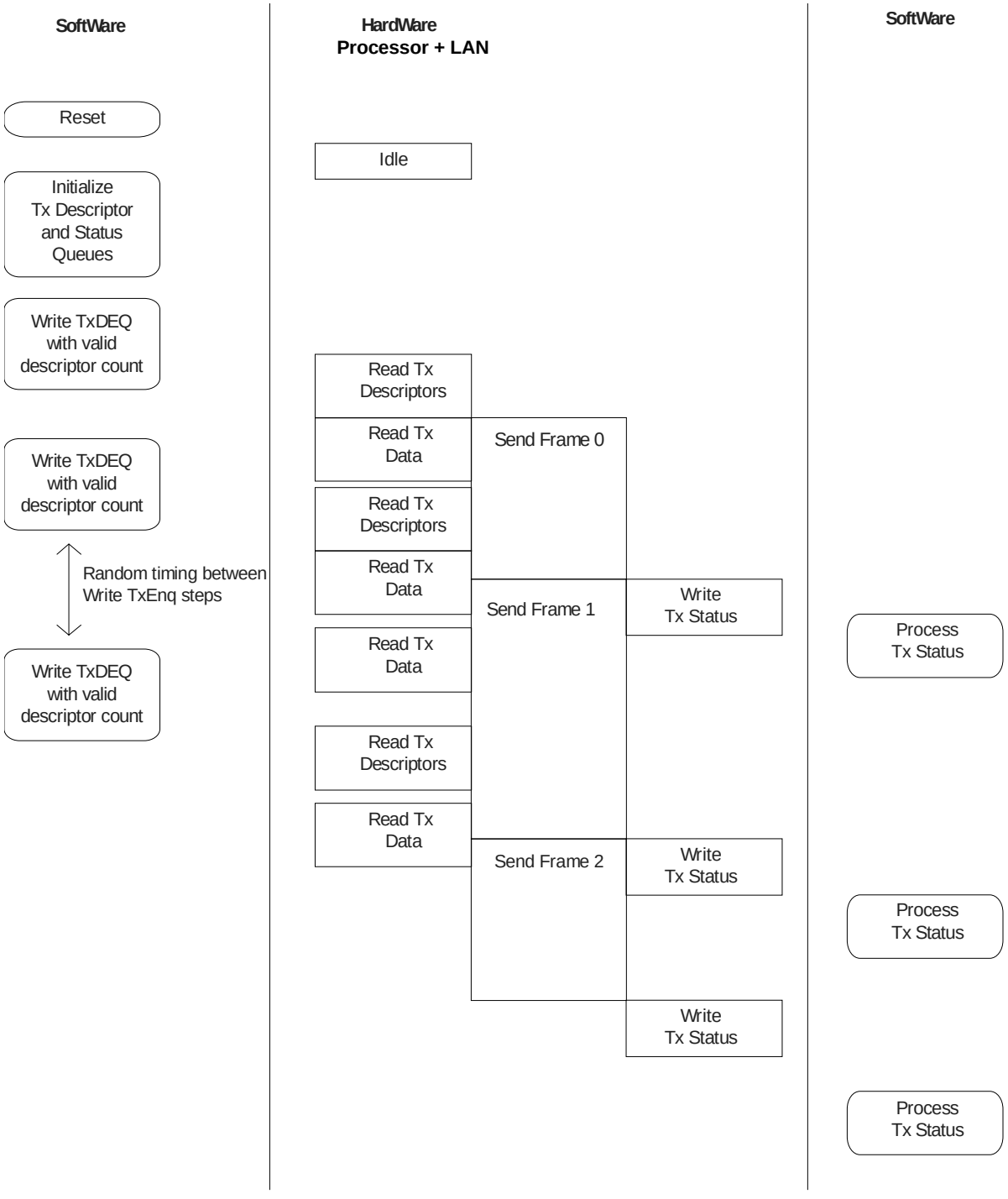


Figure 9-17. Transmit Descriptor Data/Status Flow

Copyright 2007 Cirrus Logic

9.2.4 Interrupts

9.2.4.1 Interrupt Processing

Interrupts can be associated with on chip status or with off-chip status. (Off-chip status is status that has been transferred to either the transmit or receive status queue.) The status for any outstanding interrupt event is available via two different register addresses: IntStsP (Interrupt Status Preserve) and IntStsC (Interrupt Status Clear).

Reading the IntStsP register has no effect on the bits set in the register. They may be explicitly cleared by writing a "1" back to any of the bit positions. This allows the Host to process interrupt events across multiple routines, only clearing the bits for which it has processed the corresponding events.

The IntStsC register will clear the status for all outstanding events when it is read. This provides a quick mechanism for the Host to accept all the outstanding events in one read and not incur the additional IO cycles required in specifically clearing the events.

9.2.5 Initialization

The following is the suggested hardware initialization sequence for a driver:

1. Determine what PHYs are available (poll PHYs via the management interface via MICmd, MIIData, and MIISts registers.
2. Enable auto negotiation to determine the mode of operation 10/100 Mbit, FDX/HDX. This may be needed to determine the amount of buffering to use.
3. Set RXDQBAdd and RXDCurAdd to point to the start of the receive descriptor queue
4. Set RXDQBLen to the length of the receive descriptor queue.
5. Set RXStsQBAdd and RXStsQCurAdd to point at the start of the receive status queue.
6. Set RXStsQBLen to the length of the status queue.
7. Set BMCtl.RxEn which clears the RXDEnq/RXStsEnq registers and initializes internal pointers to the queues. No bus master activity is triggered by the enable, because the enqueue registers are zero.
8. Set TXDQBAdd and TXDQCurAdd to point to the start of the transmit descriptor queue.
9. Set TXDQBLen to the length of the transmit descriptor queue.
10. Set TXStsQBAdd and TXStsQCurAdd to point to the start of the transmit status queue.
11. Set TXStsQBLen to the length of the status queue.
12. Set BMCtl.TxEn which clears the TXDEnq and initializes internal pointers to the queues. No bus master activity is triggered by the enable because the enqueue register is zero.
13. Set required interrupt mask and global interrupt mask (IntEn, GlIntMsk).

Copyright 2007 Cirrus Logic

14. Wait for RxAct (BMSts) to be set, and then enqueue the receive descriptors and status. This will trigger bus master activity for the descriptor reads.
15. Set the required values for Individual Address and Hash Table.
16. Set the required options in RXCtl and TXCtl, enabling SRxON, and STxON.
17. Set any required options in the PHY, and activate.
18. Enqueue transmit descriptors as required.

9.2.5.1 Interrupt Processing

This is the suggested method for processing an interrupt:

1. Interrupt received from the LAN Controller. This may be determined directly by vectoring to the interrupt service routine, or in a shared environment by polling the interrupt status register.
2. Read the Interrupt Status Clear register. Based on the result of the low byte, one or more of three processes need to run - receive queue processing, transmit queue processing, or other processing.

9.2.5.2 Receive Queue Processing

1. Read the RXStsQCurAdd. This is the point to which the Host needs to process the status queue.
2. Read status entries up to the value of RXStsQCurAdd.
3. For each status entry, process the receive data. Set the respective status entry to 0 after the data has been processed
4. Write the number of statuses processed to the RXStsEnq.
5. Write the number of descriptors returned to the RXDEnq. Writing once to each enqueue register is more economical on bus cycles than writing once for every descriptor or status entry. Writing once also avoids any possible delays that may otherwise occur when the controller has to process multiple accesses to the same descriptor.

9.2.5.3 Transmit Queue Processing

1. Read TXStsQCurAdd. This is the point to which the Host needs to process the status queue.
2. Read status entries up to the value of the TXStsQCurAdd.
3. For each status entry, free the data buffer.

9.2.5.4 Other Processing

The upper three bytes of the Interrupt Status register provide the specific information related to the "Other" bit in the LSB. There are a number of bits that relate to the descriptor queues.

Copyright 2007 Cirrus Logic

1. RxMiss - This bit indicates that the receive frames have been missed which may be the result of insufficient bus bandwidth being available, or of a lack of receive descriptors, or free receive status locations.
2. RxBuffers - This bit is a warning that the last free receive descriptor has been read by the controller, and RXDEnq is now zero.In a system with a dynamic number of receive buffers, this may be use as a trigger to allocate more buffers.
3. End of Chain - This bit is set when the last transmit descriptor has been read into the controller (TXDEnq equal to zero). The controller may still be transmitting at this time due to the local descriptor and data storage. This bit may be used as a signal to add more transmit descriptors, if available.
4. TxLenErr - This signifies that the controller has processed a transmit frame that exceeds the maximum allowable length. This may be caused by an internal error in the controller, a data corruption in the transmit descriptors, or a Host programming error in the descriptor queue. The error will cause the Transmit Descriptor Processor to halt. The Host should perform the Transmit Restart Process detailed in Section 9.2.5.5.
5. TxUnderrun Halt - When the Halt on Underrun (BMCtl) is set and an underrun occurs, the Transmit Descriptor Processor will halt. The underrun may be the result of insufficient bus bandwidth available, or the lack of the next transmit descriptor. The Host should perform the Transmit Restart Process detailed in Section 9.2.5.5.

9.2.5.5 Transmit Restart Process

Following a halt of the Transmit Descriptor Processor from a Halt on Underrun, TxLength Error, or setting the TxDis (BMCtl), processing may be restarted from the same point in the queues or from a different point. To start from the same point, the Host only needs to set BMCtl.TxEn. To start from a different point the following steps should be taken:

1. Process any transmit status entries in the transmit status queue (up to TXStsQCurAdd).
2. Set TxChRes in BMCtl and wait for the bit to clear. This ensures that the reset is complete.
3. Set the TXDQBAdd to the start of the descriptor queue.
4. Set TXDQBLen to the length of the descriptor queue.
5. Determine the point in the transmit descriptor where the controller should start processing, and set the TXDQCurAdd to this address. This point may be from the frame which caused the initial problem.
6. Set the TXStsQBAdd to the start of the status queue.
7. Set the TXStsQBLen to the length of the status queue.
8. Determine the point at which the controller should start writing status entries, and set the TXStsQCurAdd to this address. This can be the start of the status queue, as all existing status entries have been processed.
9. Set TxEn in BMCtl. This will cause the Transmit Descriptor Processor to reinitialize.

Copyright 2007 Cirrus Logic

10. Wait for TxAct in BMSts to be set and then write the appropriate number of descriptors remaining in the queue to TXDEnq.

9

9.3 Registers

Table 9-3. Ethernet Register List

Address	Name	Description
0x8001_0000	RXCtl	MAC Receiver Control Register
0x8001_0004	TXCtl	MAC Transmitter Control Register
0x8001_0008	TestCtl	MAC Test Control Register
0x8001_0010	MIICmd	MAC MII Command Register
0x8001_0014	MIIData	MAC MII Data Register
0x8001_0018	MIISts	MAC MII Status Register
0x8001_0020	SelfCtl	MAC Self Control Register
0x8001_0024	IntEn	MAC Interrupt Enable Register
0x8001_0028	IntStsP	MAC Interrupt Status Preserve Register
0x8001_002C	IntStsC	MAC Interrupt Status Clear Register
0x8001_0030 - 0x8001_0034		Reserved
0x8001_0038	DiagAd	MAC Diagnostic Address Register
0x8001_003C	DiagDa	MAC Diagnostic Data Register
0x8001_0040	GT	MAC General Timer Register
0x8001_0044	FCT	MAC Flow Control Timer Register
0x8001_0048	FCF	MAC Flow Control Format Register
0x8001_004C	AFP	MAC Address Filter Pointer Register
0x8001_0050 - 0x8001_0055	IndAd	MAC Individual Address Register, (shares address space with HashTbl)
0x8001_0050 - 0x8001_0057	HashTbl	MAC Hash Table Register, (shares address space with IndAd)
0x8001_0060	GlIntSts	MAC Global Interrupt Status Register
0x8001_0064	GlIntMsk	MAC Global Interrupt Mask Register
0x8001_0068	GlIntROSts	MAC Global Interrupt Read Only Status Register
0x8001_006C	GlIntFrc	MAC Global Interrupt Force Register
0x8001_0070	TXCollCnt	MAC Transmit Collision Count Register
0x8001_0074	RXMissCnt	MAC Receive Miss Count Register
0x8001_0078	RXRuntCnt	MAC Receive Runt Count Register
0x8001_0080	BMCtl	MAC Bus Master Control Register
0x8001_0084	BMSts	MAC Bus Master Status Register
0x8001_0088	RXBCA	MAC Receive Buffer Current Address Register
0x8001_0090	RXDQBAdd	MAC Receive Descriptor Queue Base Address Register
0x8001_0094	RXDQBLen	MAC Receive Descriptor Queue Base Length Register
0x8001_0096	RXDQCurLen	MAC Receive Descriptor Queue Current Length Register
0x8001_0098	RXDCurAdd	MAC Receive Descriptor Current Address Register
0x8001_009C	RXDEnq	MAC Receive Descriptor Enqueue Register
0x8001_00A0	RXStsQBAdd	MAC Receive Status Queue Base Address Register
0x8001_00A4	RXStsQBLen	MAC Receive Status Queue Base Length Register
0x8001_00A6	RXStsQCurLen	MAC Receive Status Queue Current Length Register

Copyright 2007 Cirrus Logic

Table 9-3. Ethernet Register List (Continued)

Address	Name	Description
0x8001_00A8	RXStsQCurAdd	MAC Receive Status Queue Current Address Register
0x8001_00AC	RXStsEnq	MAC Receive Status Enqueue Register
0x8001_00B0	TXDQBAdd	MAC Transmit Descriptor Queue Base Address Register
0x8001_00B4	TXDQBLen	MAC Transmit Descriptor Queue Base Length Register
0x8001_00B6	TXDQCurLen	MAC Transmit Descriptor Queue Current Length Register
0x8001_00B8	TXDQCurAdd	MAC Transmit Descriptor Current Address Register
0x8001_00BC	TXDEnq	MAC Transmit Descriptor Enqueue Register
0x8001_00C0	TXStsQBAdd	MAC Transmit Status Queue Base Address Register
0x8001_00C4	TXStsQBLen	MAC Transmit Status Queue Base Length Register
0x8001_00C6	TXStsQCurLen	MAC Transmit Status Queue Current Length Register
0x8001_00C8	TXStsQCurAdd	MAC Transmit Status Queue Current Address Register
0x8001_00D0	RXBufThrshld	MAC Receive Buffer Threshold Register
0x8001_00D4	TXBufThrshld	MAC Transmit Buffer Threshold Register
0x8001_00D8	RXStsThrshld	MAC Receive Status Threshold Register
0x8001_00DC	TXStsThrshld	MAC Transmit Status Threshold Register
0x8001_00E0	RXDThrshld	MAC Receive Descriptor Threshold Register
0x8001_00E4	TXDThrshld	MAC Transmit Descriptor Threshold Register
0x8001_00E8	MaxFrmLen	MAC Maximum Frame Length Register
0x8001_00EC	RXHdrLen	MAC Receive Header Length Register
0x8001_0100 - 0x8001_010C		Reserved
0x8001_4000 - 0x8001_FFFF	MACFIFO	MAC FIFO RAM

Control Register Description

RXCtl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD											PauseA	RxFCE1	RxFCE0	BCRC	SRxON

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD		RCRCA	RA	PA	BA	MA	IAHA	RSVD				IA3	IA2	IA1	IA0

Address:

0x8001_0000 - Read/Write

Copyright 2007 Cirrus Logic

9

Chip Reset:

0x0000_0x0x

Rx Reset:

0x0000_0000

Soft Reset:

0x0000_0000

Definition:

Receiver Control Register. The Receive Control register is reset by Rx Reset signal generated by holding the TESTSELn pin low. The same signal is also used to reset the receive MAC. The purpose of having a separate reset signal is to be able to avoid resetting the receive MAC when the AHB bus is in a powered down state (RESET active), and wake-up frames need to be detected.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

Table 9-4. Individual Accept, RxFlow Control Enable and Pause Accept Bits

IA[1:0] Individual Accept	RxFCE[1:0] Receive Flow Control Enable	PauseA Pause Accept	Action
0	X	X	Frame discarded (do not pass the address filter)
1	1	0	MAC Control frames are recognized, flow control action taken, and frames not passed to host. Non pause MAC Control frames are passed on to host.
1	1	1	MAC Control frames are recognized, flow control action taken, and all MAC control frames are passed on to host.
1	0	X	MAC Control frames are not distinguished from other frame types, all frames passed on to host.

Note: The IA field of the table means the same Individual Addresses as RxFCE, that is, IA0 implies RxFCE0 and IA1 implies RxFCE1

PauseA: Pause Accept. When set, Pause frames are passed on to the Host as regular frames. When clear, the frames are discarded. The handling of MAC Control frames depends on the Pause Accept bit as well as the appropriate Individual Accept and RxFlow Control Enable bits, as follows.

Copyright 2007 Cirrus Logic

RxFCE1: Rx Flow Control Enable, bit 1. Setting the RxFCE1 bit causes all receive frames that pass the Individual Address [1] register to be scanned for flow control format and, if detected, the Transmit Flow Control Timer register is set appropriately.

RxFCE0: Rx Flow Control Enable, bit 0. Setting the RxFCE0 bit causes all receive frames that pass the Individual Address [0] register to be scanned for flow control format and, if detected, the Transmit Flow Control Timer register is set appropriately.

BCRC: Buffer CRC. When set, the received CRC is included in the received frame buffer, and the received frame length includes the four byte CRC. When clear, neither the receive buffer nor the receive length includes the CRC.

SRxON: Serial Receive ON. The receiver is enabled when set. When clear, no incoming signals are passed through the receiver. When a frame is being received, and SerRxON is cleared, then that receive frame is completed. No subsequent receive frames are allowed until SerRxON is set again.

RCRCA: Runt CRC Accept. When set, received frames, which pass the destination address filter, but are smaller than 64 bytes, and have a CRC error are accepted. However, the MAC discards any frame of length less than 8 bytes. When clear, frames received less that 64 bytes in length with CRC errors are discarded.

RA: Runt Accept. When set, received frames, which pass the destination address filter, but are smaller than 64 bytes, with a good CRC, are accepted. However, the MAC discards any frame of length less than 8 bytes. When clear, frames received less that 64 bytes in length, with a good CRC are discarded.

PA: Promiscuous Accept. All frames are accepted when set.

BA: Broadcast Accept. When set, received frames are accepted with all 1s in the DA.

MA: Multicast Accept. When set, received frames are accepted if the DA, when hashed, matches one of the hash table bits, and the frame is a multicast frame (first bit of destination address = 1). See Descriptor Processor Transmit Registers.

Copyright 2007 Cirrus Logic

IAHA: Individual Address Hash Accept. When set, received frame are accepted when the DA is an Individual Address (first bit of DA = 0), that is accepted by the hash table. See Descriptor Processor Transmit Registers.

IA3: Individual Accept 3. When set, received frames are accepted which the DA matches the Individual Address 3 Register.

IA2: Individual Accept 2. When set, received frames are accepted which the DA matches the Individual Address 2 Register.

IA1: Individual Accept 1. When set, received frames are accepted which the DA matches the Individual Address 1 Register.

IA0: Individual Accept 0. When set, received frames are accepted which the DA matches the Individual Address 0 Register.

Note: It may become necessary for the host to change the destination address filter criteria and NOT go through a controller reset. This can be done. The host should:

1. Clear SerRxON (RXCtl) to prevent an additional received frame while the filters are being changed.
2. Modify the destination filter bits in this register.
3. Modify the Logical Address Filter, if necessary.
4. Modify the Individual Address Filter, if necessary.
5. Set SerRxON to re-enable the receiver.

When the host changes the destination filter, it is possible that a frame will be missed while SerRxON is clear.

TXCtl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								DefDis	MBE	ICRC	TxPD	OColl	SP	PB	STxON

Address:

0x8001_0004 - Read/Write

Chip Reset:

0x0000_0000

Copyright 2007 Cirrus Logic

9

Soft Reset:

0x0000_0000

Definition:

Transmit Control Register.

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
DefDis:	2-part DefDis. Before a transmission can begin, the MAC follows a deferral procedure. With the 2-part DefDis bit clear, the deferral is the standard two-part deferral as defined in ISO/IEC 8802-3 paragraph 4.2.3.2.1. With the 2-part DefDis bit set, the two-part deferral is disabled. See Transmit Back-Off paragraph.
MBE:	ModBackoff Enable. When clear, the ISO/IEC standard backoff algorithm is used. When set, the Modified Backoff algorithm is used, which delays longer after each of the first three Tx collisions.
ICRC:	Inhibit CRC. When this bit is set, there will be no CRC appended to transmit frames. If the Abort Frame bit is set in the transmit descriptor for a frame, the frame will be terminated with a bad CRC.
TxPD:	Tx Pad Disable. When this bit is set, the MAC will not pad the frame to the legal minimum size (64 bytes). If clear, the MAC will pad the frame to the minimum legal frame size if the supplied length is less than 64 bytes. The padded characters will be the last supplied character in the frame, repeated.
OColl:	One Collision. When this bit is set, no attempt is made to resend frames in the event of a collision.
SP:	Send Pause. When set by the host, this bit causes a pause frame to be transmitted at the earliest opportunity. This is at the end of the current frame, if one is already in progress. This bit will remain set until the transmission of the frame has started. The pause frame is comprised of the following elements:

Destination Address	Individual Address number 6
Source Address	Individual Address number 1
Type Field	Type Field defined in the Flow Control Format register0
Opcode	0x0001
Pause Time	Pause Field defined in the Flow Control Format register
Pad fill	

Copyright 2007 Cirrus Logic

9

PB: Pause Busy: This bit remains set as long as a pause frame is being transmitted. Only one pause frame may be sent at any time, therefore the Send Pause and Pause Busy bits should be zero before a new pause frame is defined.

STxON: Serial Transmit ON. The transmitter is enabled when set. When clear, no transmissions are allowed. When a frame is being transmitted, and STxON is cleared, then that transmit frame is completed. No subsequent frames are transmitted until STxON is set again.

SelfCtl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	MDCDIV						PSPRS	RWP	RSVD	GPO0	PUWE	PDWE	MIIL	RSVD	RESET

Address:
0x8001_0020 - Read/Write

Chip Reset:
0x0000_0F10

Soft Reset:
0x0000_0000

Definition:
Self Control Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

MDCDIV: MDC Clock Divisor. HCLK is divided by MDCDIV + 1 to create the MDC clock frequency. Default value is 0x07, which is divide by 8.

Note: Clause 22.2.2.1 in the IEEE-802-3 specification states that the maximum MDC clock rate is 2.5 MHz. Most PHYs support clock rates faster than 2.5 MHz. So, modify the MDCDIV value according to the PHYs specification.

PSPRS: Preamble Suppress. Default is 1.
1 = The first MDC qualifies an SFD on MDIO.
0 = Get 32 ones before SFD.

Note: The user must check the datasheet of the PHY being used in the design. If the PHY needs a preamble for reading/writing to/from PHY registers, the PSPRS must be cleared (set to 0).

Copyright 2007 Cirrus Logic

The following procedure will correctly set the SelfCtl register value:

1. Read the value of SelfCtl
2. Clear PSPRS bit in SelfCtl Register.
3. Read/write PHY registers.
4. Restore the old value to SelfCtl.

RWP:	Remote Wake Pin. This bit reflects the current state of the **REMWAKE** pin. Following a system power up, caused by a Remote Wake-up frame being detected by the MAC, this bit is set.
GPO0:	General Purpose Output 0. This bit directly controls the GPO[0] pin. A "1" corresponds to a logic high on the pin.
PUWE:	Power Up Wake-up Enable. Setting the Power Up Wake-up enable bit causes the MAC to enter the remote wake-up mode, during normal operation (AHB bus powered up). In this mode all receive frames that pass the destination address filter are scanned for the remote wake-up pattern (six bytes of 0xFF followed directly by sixteen consecutive copies of the Individual address). When this pattern is detected, the **REMWAKE** pin is driven high and Remote Wake-up (Interrupt Status is set).
PDWE:	Power Down Wake-up Enable. Setting the Power Down Wake-up Enable bit causes the MAC to enter the remote wake-up mode when the AHB bus is powered down. In this mode all receive frames that pass the destination address filter are scanned for the remote wake-up pattern (six bytes of FFh followed directly by sixteen consecutive copies of the Individual address). When this pattern is detected, the **REMWAKE** pin is driven high, and can be used to initiate a system power up, the state of the **REMWAKE** pin is visible via the Remote Wake Pin bit of this register.
MIIL:	MII Loopback. Setting the MII Loopback bit causes transmit data to be diverted back into the receive data path prior to the MII interface pins, the transmit data does not appear on the MII bus and the receive data on the MII bus is ignored. The clock for the transmit and receive data is derived from the AHB CLK in the loopback mode. It is strongly recommended that TXCLK and RXCLK come from a single clock source with minimum skew in order to ensure the proper operation of the loopback test. For reliable operation a software reset should be issued when the MII loopback bit is changed.

Copyright 2007 Cirrus Logic

9

RESET: Soft Reset. This is an act-once bit. When set, a Soft Reset is initiated immediately, this will reset the FIFO, mac and Descriptor Processor. This bit is cleared as a result of the reset. Driver software should wait until the bit is cleared before proceeding with MAC initialization.

DiagAd

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								ADDR							

Address:

0x8001_0038 - Read/Write

Chip Reset:

0x0000_0000

Soft Reset:

0x0000_0000

Definition:

Diagnostic Address Register. The Diagnostic Address Register provides an indirect addressing method to point to internal diagnostic locations, which provide access to features not required for normal driver operation. To access the internal registers, the address of the register is written to the Diagnostic Address register, and the Diagnostic Data register is used to access the actual data.

Bit Descriptions:

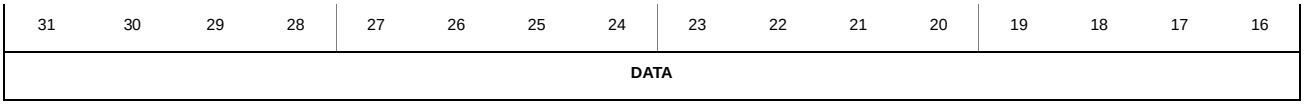
RSVD: Reserved. Unknown During Read.

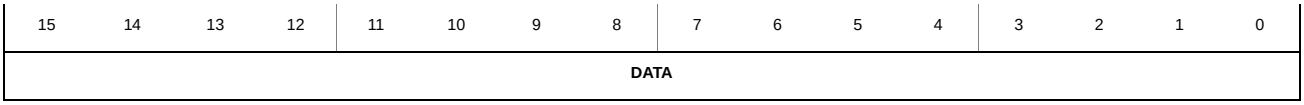
ADDR: Diagnostic Address. The following table identifies the address map.

Address	Register Name
0x00	Debug Control
0x04	Debug FIFO Control
0x08	Debug FIFO Data
0x98	Receive Data FIFO Pointers
0x9C	Transmit Data FIFO Pointers
0xA0	Receive Status FIFO Pointers
0xA4	Transmit Status FIFO Pointers
0xA8	Receive Descriptor FIFO Pointers
0xAC	Transmit Descriptor FIFO Pointers

Copyright 2007 Cirrus Logic

DiagDa





Address:

0x8001_003C - Read/Write

Chip Reset:

0x0000_0000

Soft Reset:

0x0000_0000

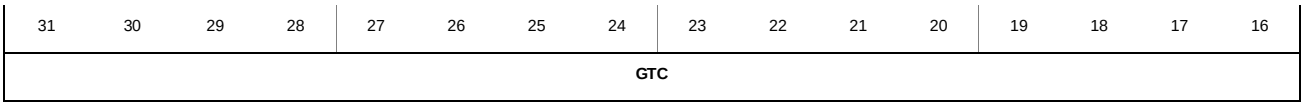
Definition:

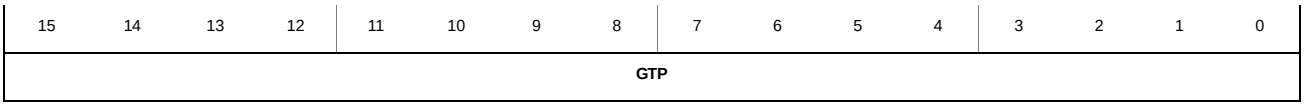
Diagnostic Data Register. The Diagnostic Data Register provides access to the internal register pointed to by the value in the Diagnostic Address register. For debug only.

Bit Descriptions:

DATA: Internal register data value.

GT





Address:

0x8001_0040 - Read/Write

Chip Reset:

0x0000_0000

Soft Reset:

0x0000_0000

Definition:

General Timer Register

Copyright 2007 Cirrus Logic

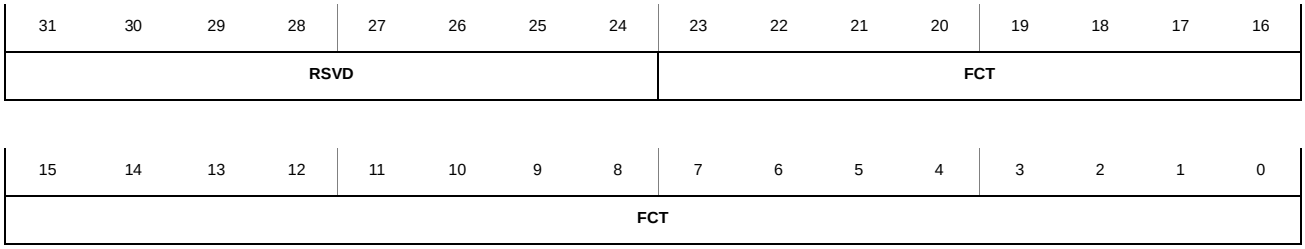
9

Bit Descriptions:

GTC: General Timer Count, read only. The timer count contains the running value of the timer function, it cannot be written to directly. When the General Timer Period is written and the same value is loaded into the General Timer Count, or when the count value reaches 0, it is reloaded from the General Timer Period. Additionally when the count reaches zero, the Timeout Status (Interrupt Status register) is set. The timer value is decremented at 1/8th of the transmit bit rate.

GTP: General Timer Period, read/write. The Timer Period holds the periodic time for the timer. When the period is written, the count is preloaded with the same value. Setting a value of zero in the Period disables the generation of Timeout Status.

FCT



Address:
0x8001_0044 - Read/Write

Chip Reset:
0x0000_0000

Soft Reset:
0x0000_0000

Definition:
Flow Control Timer

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

FCT: Flow Control Timer value. The Flow Control Timer is loaded as a result of receiving a flow control frame, with the pause value from the received frame. The value in the timer is then decremented every 512 bit times, as soon as the transmit line is idle. While the timer is non zero, no new transmit frames are started. The decrement time depends on the speed, but always corresponds to the duration of a 64 byte minimum packet.

Copyright 2007 Cirrus Logic

9

FCF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MACCT															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPT															

Address:
0x8001_0048 - Read/Write

Chip Reset:
0x0000_0000

Soft Reset:
0x0000_0000

Definition:
Flow Control Format Register

Bit Descriptions:

MACCT: MAC Control Type. The MAC Control Type field defines the Ethernet type field for receive and transmit MAC control frames. This is used in the processing of transmit and receive pause frames, which are a special form of MAC control frames. For a receive frame to be identified as a pause frame, the following conditions must be met:

- The destination address must match one of first two individual addresses, with the appropriate RxFlowControlEn bit set.
- The Ethernet type field must match MAC Control Type.
- The first two data bytes of the frame must equal 0x0001.

When a transmit pause command is processed, the MAC Control Type is inserted in the transmit frame as the ethernet type field.

TPT: Transmit Pause Time. When a transmit pause command is processed, the Transmit Pause Time is inserted as the actual time to pause.

The format of a transmit pause frame is:

1. Destination address = Individual address[6] (6 bytes)
2. Source address = Individual address[0] (6 bytes)
3. Type field = MAC Control Type (2bytes)
4. Opcode = 0x0001 (2bytes)

Copyright 2007 Cirrus Logic

5. Pause time = Transmit Pause Time (FCF) (2bytes)
6. Padding to complete minimum size packet.
7. CRC

AFP

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD														AFP	

Address:
0x8001_004C - Read/Write

Chip Reset:
0x0000_0000

Soft Reset:
0x0000_0000

Definition:
Address Filter Pointer Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

AFP: Address Filter Pointer. The Address Filter Pointer controls access to a block of storage which is used to hold MAC addresses, and the destination address hash table. The pointer defines which set of address match functions are visible to the Host at offset 0x0050 through 0x005F.

Table 9-5. Address Filter Pointer

AFP	Data Accessed at Offset 0050 through 005F
000	This is the primary Individual Address, used in the recognition of Wake-up frames, as the source address for transmit pause frames, and may be optionally used to qualify receive pause frames.
001	This is a secondary MAC address that may be optionally used to qualify receive pause frames
010 011	These secondary addresses are only used for qualifying the destination addresses of receive frames.
100 101	These locations are not implemented
110	This address is used as the destination address of transmit pause frames
111	This block comprises the hash table used for qualifying the destination of receive frames.

Copyright 2007 Cirrus Logic

IndAd

47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IAD															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IAD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IAD															

Address:

0x8001_0050 through 0x8001_0055 - 6 Bytes - Read/Write, when AFP = 000b, 010b or 001b

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Individual Address Register. There are five different Individual Addresses accessible at offset 0x050, the Address Filter Pointer determines which one is accessed at any one time. The first four addresses (pointer offset 0x000 through 0x011), may be used to implement destination address filters for receive frames. The first two may also be used to qualify receive frames for flow control processing, and the first address is used for wake-up frame processing. The fifth address (pointer offset 0x110), is only used as the destination address for transmit pause frames.

The least significant byte of the Individual Address corresponds to the first byte of the address on the serial interface, with the least significant bit of the byte corresponding to the first bit on the serial interface.

Bit Descriptions:

IAD: Individual Address.

Copyright 2007 Cirrus Logic

HashTbl

9

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
HTb															

47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
HTb															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HTb															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTb															

Address:

0x8001_0050 through 0x8001_0057 - 8 Bytes - Read/Write, when AFP = 111b

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Hash Table Register. The hash table is used as a way of filtering groups of addresses in the receiver. Following the reception of the destination address (first 6 bytes of a receive frame), the upper 6 bits of the computed CRC are used as an address into the hash table. If the bit accessed by this address is a "1", the frame passes the hash table test, if the bit is a "0", the frame fails the hash table test.

The hash table may be used for either or both of individual addressed frames and group address frames, depending on the IAHA and MA bits in RXCtl. A frame has a group address if the first bit of the frame is a one.

If an individual address frame passes the hash test and the IAHA bit is set, the frame passes the destination filter.

If a group address frame passes the hash test and the MA bit set, the frame passes the destination filter.

Bit Descriptions:

HTb: Hash Table entries.

Copyright 2007 Cirrus Logic

TXCollCnt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXC															

Address:

0x8001_0070 - Read Only

Chip Reset:

0x0000_0000

Soft Reset:

0x0000_0000

Definition:

Transmit Collision Count Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

TXC: Transmit Collision Count. The transmit collision count records the total number of collisions experienced on the transmit interface, including late collisions. When the most significant bit of the count is set, an optional interrupt may be generated. The register is cleared automatically following a read and writing to the register will have no effect.

RXMissCnt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RMC															

Address:

0x8001_0074 - Read Only

Chip Reset:

0x0000_0000

Soft Reset:

0x0000_0000

Copyright 2007 Cirrus Logic

9

Definition:

Receive Miss Count Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RMC: Receive Miss Count. The Receive Miss Count records the number of frames that pass the destination address filter, but fail to be received due to lack of bus availability or lack of receive storage. Frames that are partially stored and marked as overruns are included in the count. When the most significant bit of the count is set, an optional interrupt may be generated. The register is cleared automatically following a read, writing to the register will have no effect.

RXRuntCnt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RRC															

Address:

0x8001_0078 - Read Only

Chip Reset:

0x0000_0000

Soft Reset:

0x0000_0000

Definition:

Receive Runt Count Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RRC: Receive Runt Count. The receive runt count records the total number of runt frames received, including those with bad CRC. When the most significant bit of the count is set, an optional interrupt may be generated. The register is cleared automatically following a read, writing to the register will have no effect.

Copyright 2007 Cirrus Logic

TestCtl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								MACF	MFDX	DB	RSVD				

Address:

0x8001_0008 - Read/Write

Chip Reset:

0x0000_0000

Soft Reset:

0x0000_0000

Definition:

Test Control Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

MACF: MAC Fast. When set, internal MAC timers for link pulses and collision backoff are scaled in order to speed-up controller testing. When clear, normal timing is used.

MFDX: MAC Full Duplex. This bit is used to enable full duplex operation, when set, the transmitter ignores carrier sense for transmit deferral. For normal loopback testing this bit should be set.

DB: Disable backoff. When set, the backoff algorithm is disabled. The MAC transmitter looks only for completion of the Inter Frame Gap before starting transmission. When clear, the backoff algorithm is used as described in Section 9.1.4 on page 9--7.

IntEn

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD / RWIE		RxMIE	RxBIE	RxSQIE	TxLEIE	ECIE	TxUHIE	RSVD					MOIE	TxCOIE	RxROIE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD		RSVD	MIIIE	PHYSIE	TIE	RSVD	SWIE	RSVD				TSQIE	REOFIE	REOBIE	RHDRIE

Copyright 2007 Cirrus Logic

9

Address:
0x8001_0024 - Read/Write

Chip Reset:
0x0000_0000

Soft Reset:
0x0000_0000

Definition:
Interrupt Enable Register

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
RWIE:	Remote Wake-up Interrupt Enable. Setting this bit causes an interrupt to be generated when a remote wake-up frame is detected and the MAC is in the Remote Wake-up mode (RXCtl).
RxMIE:	Receiver Miss Interrupt Enable. When set, this bit will cause an interrupt whenever a complete receive frame is discarded due to lack of storage. This may be as a result of long bus latency or insufficient receive descriptors. The total number of missed frames is also counted in the RxMiss Counter.
RxBIE:	Receive Buffer Interrupt Enable. When set, this bit will cause an interrupt to be generated when the last available receive descriptor has been read into the MAC.
RxSQIE:	Receive Status Queue Interrupt Enable. When this bit is set, an interrupt will be generated when the last available status queue entry has been written (RXStsEnq = 0).
TxLEIE:	Transmit Length Error Interrupt Enable. Setting this bit causes an interrupt to be generated when a transmit frame equals or exceeds the length specified in the Max Frame Length register.
ECIE:	End of Chain Interrupt Enable. The end of chain interrupt is generated when the last transmit descriptor has been loaded into the MAC. There may still be transmit descriptors and or transmit data remaining in the MAC at this time.
TxUHIE:	Transmit Underrun Halt Interrupt Enable. If there is a transmission, and the MAC runs out of data before the full transmitted length, then there is a transmit underrun. If the MAC is programmed to halt in this condition (Bus Master Control), setting TxUnderrunHaltiE will cause an interrupt to be generated.

Copyright 2007 Cirrus Logic

MOIE: Receive Miss Overflow Interrupt Enable. If received frames are lost due to slow movement of receive data out of the receive buffers, then a receive miss is said to have occurred. When this happens, the RxMISS counter is incremented. When the MSB of the count is set, the MissCnt bit in the Interrupt Status Register is set. If the MissCntiE bit is set at this time, an interrupt is generated.

TxCOIE: Transmit Collision Overflow Interrupt Enable. When a transmit collision occurs, the transmit collision count is incremented. When the MSB of the count is set, the TXCollCnt bit in the Interrupt Status Register is set. If the TxCollCntiE is set at this time, an interrupt is generated.

RxROIE: Receive Runt Overflow Interrupt Enable. When a runt frame is received with a CRC error, the RxRuntCnt register is incremented. When the MSB of the count is set the RuntOv bit is set in the Interrupt Status Register. If the RuntOviE bit is set at this time, an interrupt is generated.

MIIIE: MII Management Interrupt Enable. When set, the MII Interrupt enable causes an interrupt to be generated whenever a management read or write cycle is completed on the MII bus.

PHYSIE: The PHY Status Interrupt Enable bit provides a mechanism to generate an interrupt whenever a change of status is detected in the PHY.

TIE: Setting the Timer Interrupt Enable bit will cause an interrupt to be generated whenever the general timer (GT) counter reaches zero.

SWIE: Writing a "1" to this bit causes a software generated interrupt to be generated. The SWint bit in the Interrupt Status register is set to indicate the cause of the interrupt. This bit will always read zero.

TSQIE: Transmit Status Queue Interrupt Enable. Setting this bit causes an interrupt to be generated whenever a transmit status is posted to the transmit status queue.

REOFIE, REOBIE, RHDRIE: Setting all three bits causes interrupts to be generated whenever a receive-end-of-frame status, or a receive-end-of-buffer status, or a receive-header status is written to the receive status queue.

Copyright 2007 Cirrus Logic

IntStsP/IntStsC

9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD	RWI	RxMI	RxBI	RxSQI	TxLEI	ECI	TxUHI	RSVD					MOI	TxCOI	RxROI

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD			MIII	PHYSI	TI	AHBE	SWI	RSVD			OTHER	TxSQ	RxSQ	RSVD	

Address:

0x8001_0028, for IntStsP - Read/Write
0x8001_002C, for IntStsC - Read Only

Chip Reset:

0x0000_0000

Soft Reset:

0x0000_0000

Definition:

Interrupt Status Preserve and Clear Registers. The interrupt status bits are set when the corresponding events occur in the MAC. If the corresponding interrupt enable bit is set in the interrupt enable register, an interrupt signal will be generated.

Interrupt status is available at two different offsets: Interrupt Status Preserve and Interrupt Status Clear. Both offsets are a read of the same storage. Reading the Interrupt Status register Preserve has no effect on the status in the register, but writing a 1 to a location in this register clears the status bit, writing a zero has no effect. Reading the Interrupt Status Clear register clears all the bits in the register that are accessed as defined by the AHB **HSIZE** signal. Therefore a routine which will handle all reported status may read via the Interrupt Status Clear thereby saving a write operation.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RWI: Remote Wake-up Interrupt. The remote wake status is set when a remote wake-up frame is received, and the RemoteWakeEn (RXCtl) is set. A remote wake-up frame must pass the receive destination address filter and have a contiguous sequence of 6 bytes of FFh followed by 8 repetitions of the Individual Address and be a legal frame (legal length and good CRC).

Copyright 2007 Cirrus Logic

RxMI: RxMI is set when a receive frame was discarded due to the internal FIFO being full. This may be as a result of a long latency in acquiring the bus or a lack of receive descriptors. RxMiss is not set in response to a frame that was partially stored in the FIFO and then discarded due to lack of FIFO space. This is marked as an Overrun Error in the Status Queue.

RxBI: RxBuffers is set when the last available receive descriptor has been read into the MAC (RxDesEnq = 0). Free descriptors may still be available in the MAC to accommodate receive frames.

RxSQI: The Receive Status Queue bit is set when the last free status queue location has been written (RXStsEnq = 0).

TxLEI: The Transmit Length Error status is set when any excessively long frame is transferred into the transmit data FIFO. When this occurs, the MAC assumes an error has occurred in the transmit descriptor queue, and therefore stops further transmit DMA transfers. An excessively long frame is defined as one equal or longer than the value programmed in the Max Frame Length register. The frame itself will be terminated with a bad CRC.

ECIE: When set to 1, this bit indicates that the MAC has exhausted the transmit descriptor chain.

TxUHI: This bit is set if the MAC runs out of data during a frame transmission, and the Underrun Halt bit (BMCtl) is set, at this time the Transmit Descriptor Processor will have been halted. If the Underrun Halt bit is clear, the MAC will write an Underrun Status for the frame and continue to the next transmit frame.

MOI: If received frames are lost due to slow movement of receive data out of the receive buffers, then a receive miss is said to have occurred. When this happens, the RxMISS counter is incremented. When the MSB of the count is set, the MissCnt bit in the Interrupt Status Register is set. If the MissCntiE bit is set, an interrupt will be generated.

TxCOI: When a transmit collision occurs, the transmit collision count is incremented. When the MSB of the count is set the TxCOI bit in the Interrupt Status Register is set. If the TxCOIE bit is set, an interrupt will be generated.

Copyright 2007 Cirrus Logic

9

RxROI: When a runt frame is received with a CRC error, the RxRuntCnt register is incremented, when the MSB of the count is set, the RuntOv bit is set in the Interrupt Status Register. If the RxROIE bit is set, an interrupt will be generated.

MIIII: The MII Status bit is set whenever a management operation on the MII bus is completed.

PHYI: The PHY Status bit is set when the MAC detects a change of status event in the PHY.

TI: The Timeout bit is set when the general timer (GT) count register reaches zero.

AHBE: This bit is set if a MAC generated AHB cycle terminated abnormally. The Queue ID bits (Bus Master Status) will indicate the DMA Queue which was active when the abort occurred. DMA operation is halted on all queues until this bit is cleared, and the queues are restarted via the Bus Master Control register.

OTHER: This bit is set when a status other than that covered by bits 10, 3 and 2 is present.

TxSQ: This bit is set when a status affecting the transmit status queue has been posted.

RxSQ: This bit is set when a status affecting the receive status queue has been posted. This bit can only be set if bit 2 (REOFIE), bit 1 (REOBIE) and bit 0 (RHDRIE) of the Interrupt Enable (IntEn) register are set (enabled).

GlIntSts

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT	RSVD														

Address: 0x8001_0060 - Read/Write

Chip Reset: 0x0000_0000

Soft Reset: 0x0000_0000

Copyright 2007 Cirrus Logic

Definition:

Global Interrupt Status Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

INT: Global interrupt bit. This bit is set whenever the MACint signal to the interrupt controller is active. Writing a one to this bit location will clear this bit until a new interrupt condition occurs.

GlIntMsk

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT	RSVD														

Address:

0x8001_0064 - Read/Write

Chip Reset:

0x0000_0000

Soft Reset:

0x0000_0000

Definition:

Global Interrupt Mask Register. This register is used to mask the GlIntSts bit, to allow of block interrupts to the processor.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

INT: Global interrupt mask bit. When set, any interrupt enabled by the Interrupt Enable Register will set the Global Interrupt Status interrupt bit. When clear, no interrupts will reach the processor.

Copyright 2007 Cirrus Logic

9

GlIntROSts

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT	RSVD														

Address:

0x8001_0068 - Read Only

Chip Reset:

0x0000_0000

Soft Reset:

0x0000_0000

Definition:

General Interrupt Read-Only Status register. This is a read-only version of the Global Interrupt Status Register.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

INT: Global interrupt read-only status bit. This bit is set whenever the MACint signal to the interrupt controller is active.

GlIntFrc

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT	RSVD														

Address:

0x8001_006C - Write Only

Chip Reset:

0x0000_0000

Soft Reset:

0x0000_0000

Definition:

Global Interrupt Force Register. This register allows software to generate an interrupt.

Copyright 2007 Cirrus Logic

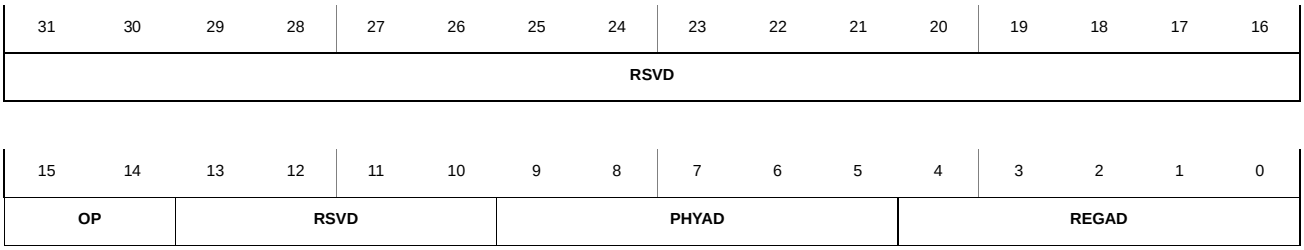
Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
INT:	Global interrupt force bit, write only, always reads zero. Writing a one to this bit will set the Global Interrupt Status bit, if it is enabled. Writing a zero has no effect.

MII/PHY Access Register Descriptions

All PHY registers are accessed through the MII Command, Data and Status Registers. Write operations are accomplished by writing the required data to the MII Data Register and then writing the required Command to the MII Command Register (Opcode = 01, PhyAd = target phy, RegAd = target register), which causes the Busy bit (MII Status) to be set. When the Busy bit is clear, the write operation has been performed. Read operations are performed by writing a read command to the MII Command register (Opcode = 10b, PhyAd = target phy, RegAd = source register), which will also cause the Busy bit (MII Status) to be set. When the read operation has been completed, the Busy bit is cleared and the read data is available in the MII Data register.

MIICmd



Address:
0x8001_0010 - Read/Write

Chip Reset:
0x0000_0000

Soft Reset:
0x0000_0000

Definition:
MII Command Register. Provides read-write access to the external PHY registers using the MII command data port.

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
OP:	OPcode. This Opcode field defines the type of operation to be performed to the appropriate PHY register. 10 - Read register 01 - Write register

Copyright 2007 Cirrus Logic

9

PHYAD: PHY Address. This field defines which external PHY is to be accessed.

REGAD: Register Address. This field defines the particular register in the PHY to be accessed.

MIIData

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIIData															

Address:
0x8001_0014 - Read/Write

Chip Reset:
0x0000_0000

Soft Reset:
0x0000_0000

Definition:
MII Data Transfer Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

MIIData: MII Data Register. This register contains the 16 bit data word that is either written to or read from the appropriate PHY register.

MIISts

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															BUSY

Address:
0x8001_0018 - Read Only

Chip Reset:
0x0000_0000

Copyright 2007 Cirrus Logic

Soft Reset:
0x0000_0000

Definition:
MII Status Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

Busy: MII Busy. The Busy bit is set whenever a command is written to the MII Command Register. It is cleared when the operation has been completed.

Descriptor Processor Registers

The Descriptor Processor Registers are in three parts: the bus master control, receive registers, and transmit registers.

BMCtl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD		MT	TT	UnH	TxChR	TxDis	TxEn	RSVD	EH2	EH1	EEOB	RSVD	RxChR	RxDis	RxEn

Address:
0x8001_0080 - Read/Write

Chip Reset:
0x0000_0000

Soft Reset:
0x0000_0000

Definition:
Bus Master Control Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

Copyright 2007 Cirrus Logic

MT: Manual Transfer. Writing a one to this bit causes all internal FIFOs to be marked pending for transfer, as if they had crossed their threshold. This provides a mechanism for flushing stale status from the internal FIFOs, when the Timed Transfer is not used and non zero thresholds have been set. When the Manual Transfer is set, the Transfer Pending (BMCtl), is set until all FIFOs have been either active for a DMA transfer, or have been determined inactive (that is, an empty receive data FIFO). When reading the BMCtl register, the Manual Transfer bit will always return a zero.

TT: Timed Transfer. Setting the Timed Transfer bit causes the internal FIFOs to be marked as pending for transfer whenever the timer reaches zero. This provides a mechanism for flushing stale status from the internal FIFOs when a non zero threshold has been set.

UnH: Underrun Halt. When set, this bit causes the transmit descriptor to perform the following operations when a transmit underrun is encountered:

1. Halt all transmit DMA operations.

2..Flush the transmit descriptor queue.

3.Set transmit enqueue to zero.

This allows the host to re-initialize the Transmit Descriptor Processor, to start at the desired point. When clear, the MAC will proceed to the next transmit frame in the queue.

TxChR: Transmit Channel Reset. Writing a "1" to Transmit Channel Reset causes the Transmit Descriptor Processor and the transmit FIFO to be reset. This bit is an act-once-bit and will clear automatically when the reset is complete.

TxDis: Transmit Disable. Writing a "1" to Transmit Disable causes the transmit DMA transfers to be halted. If a transmit frame is currently in progress, transfers are halted when the transmit status is written to the status buffer. When transfers have been halted, the TxAct bit (Bus Master Status) is clear. TxDis is an act-once-bit and will clear immediately.

Copyright 2007 Cirrus Logic

TxEn: Transmit Enable. Writing a one to Transmit Enable causes transmit DMA transfers to be enabled. This is reflected in TxAct (Bus Master Status) being set. TxEn is an act-once-bit and will clear automatically when the enable is complete. The first time the TxEn bit is set following an AHB reset, or a TxChRes, the MAC performs a transmit channel initialization. During this initialization the TXDEnq is cleared, and the Transmit Descriptor and Status Queues are calculated. When the initialization is complete, the TxAct (BMSts) is set.

EH2: Enable Header 2. When Enable Header2 is set, a status is written to the receive status queue when the number of bytes specified in Receive Header Length2 have been transferred to the receive data buffer. If the transfer either fills a receive buffer or ends a receive frame, only an end of buffer or end of frame status is generated. The value in Receive Header Length 2 should be greater than the value in Receive Header Length 1 in order to generate a status event.

EH1: Enable Header 1. When Enable Header1 is set, a status is written to the receive status queue when the number of bytes specified in Receive Header Length1 have been transferred to the receive data buffer. If the transfer either fills a receive buffer or ends a receive frame, only an end of buffer or end of frame status is generated.

EEOB: Enable EOB. When Enable End Of Buffer bit is set, a status is written to the receive status queue whenever an end of receive buffer is reached. If reaching the end of the receive buffer coincides with the end of frame, only one status is written to the queue.

RxChR: Receive Channel Reset. Writing a "1" to Receive Channel Reset causes the Receive Descriptor Processor and the receive FIFO to be reset. This bit is an act-once-bit and will clear automatically when the reset is complete.

RxDis: Receive Disable. Writing a "1" to Receive Disable causes receive DMA transfers to be halted. If a receive frame is currently in progress, transfers will be halted when the receive frame status has been transferred to the status buffer. When the transfers are halted, the RxAct bit (Bus Master Status) is cleared. This bit is an act-once-bit and will clear immediately.

Copyright 2007 Cirrus Logic

RxEn: Receive Enable. Writing a one to Receive Enable causes receive DMA transfers to be enabled. This is reflected in RxAct (Bus Master Status) being set. This bit is an act-once-bit and will clear automatically when the enable is complete. The first time the RxEn bit is set following a AHB reset, or a RxChRes, the MAC performs a receive channel initialization. During this initialization the RXDEnq, and RXStsEnq registers are cleared and the endpoints of the Receive Descriptor and Status Queues are calculated. When the initialization is complete, the RxAct (BMSts) is set.

BMSts

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								TxAct	RSVD		TP	RxAct	QID		

Address: 0x8001_0084 - Read Only

Chip Reset: 0x0000_0000

Soft Reset: 0x0000_0000

Definition: Bus Master Status Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

TxAct: Transmit Active. When this bit is set, the channel is active and may be in the process of transferring transmit data. Following a TxDisable Command (Bus Master Control), when transfers have been halted, this bit is cleared.

TP: Transfer Pending. When the Manual Transfer bit (BMCtl) is set, the Transfer Pending bit is set, until all internal FIFOs have either been active for a DMA transfer, or have been determined to be inactive (that is, empty transmit status FIFO).

Copyright 2007 Cirrus Logic

RxAct: Receive Active. When this bit is set, the channel is active and may be in the process of transferring receive data. Following a RxDisable Command (Bus Master Control), when transfers have been halted, this bit is cleared.

QID: Queue ID. The queue ID reflects the current or last DMA queue active on the AHB bus. When an AHB error halts DMA operation, this field may be used to determine the queue that caused the error.

ID Type of transfer
000 - Receive data
001 - Transmit data
010 - Receive status
011 - Transmit status
100 - Receive descriptor
101 - Transmit descriptor

Descriptor Processor Receive Registers

RXDQBAdd

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDBA															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDBA															

Address:

0x8001_0090 - Read/Write

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Receive Descriptor Queue Base Address register. The Receive Descriptor Queue Base Address defines the system memory address of the receive descriptor queue, this address is used by the MAC to reload the Receive Current Descriptor Address whenever the end of the descriptor queue is reached. The base address should be set at initialization time and must be set to a word aligned memory address.

Bit Descriptions:

RDBA: Receive Descriptor Base Address.

Copyright 2007 Cirrus Logic

RXDQBLen

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDBL															

Address:

0x8001_0094 - Read/Write

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Receive Descriptor Queue Base Length register. The Receive Descriptor Queue Base Length defines the actual number of bytes in the receive descriptor queue, which thereby sets the number of receive descriptors that can be supplied to the MAC. The length should be set at initialization time and must define an integral number of receive descriptors.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RDBL: Receive Descriptor Base Length.

RXDQCurLen

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDCL															

Address:

0x8001_0096 - Read/Write. Note half word alignment.

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Copyright 2007 Cirrus Logic

Definition:

Receive Descriptor Queue Current Length register. The Receive Descriptor Queue Current Length defines the number of bytes between the Receive Descriptor Current Address and the end of the receive descriptor queue. This value is used internally to wrap the pointer back to the start of the queue. The register should not normally be written.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RDCL: Receive Descriptor Current Length.

RXDCurAdd

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDCA															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDCA															

Address:

0x8001_0098 - Read/Write

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Receive Descriptor Current Address register. The Receive Current Descriptor Address contains the pointer to the next entry to be read from the receive descriptor queue. This should be set at initialization time to the required starting point in the descriptor queue. During operation the MAC will update this address following successful descriptor reads. Intermediate values in this register will not necessarily align to descriptor boundaries, nor directly effect the current descriptor in use because several descriptors may be buffered inside the MAC.

Bit Descriptions:

RDCA: Receive Descriptor Current Address.

Copyright 2007 Cirrus Logic

RXDEnq

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDV															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								RDI							

Address:

0x8001_009C - Read/Write

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Receive Descriptor Enqueue register. The Receive Descriptor Enqueue register is used to define the number of valid entries in the descriptor queue. The register operates as follows: only the Receive descriptor Increment field is writable and any value written to this field is added to the existing Receive Descriptor Value. Whenever complete descriptors are read by the MAC, the Receive Descriptor Value is decremented by the number read. For example, if the Receive Descriptor Value is 0x07 and the Host writes 03 to the Receive Descriptor Increment, the new Value will be 0x0A. If the controller then reads two descriptors, the Value will be 0x08.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RDV: Receive Descriptor Value.

RDI: Receive Descriptor Increment.

RXBCA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RBCA															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RBCA															

Address:

0x8001_0088 - Read/Write

Copyright 2007 Cirrus Logic

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Receive Buffer Current Address register. The Receive buffer current address contains the current address being used to transfer receive data. This value may be useful in debugging.

Bit Descriptions:

RBCA: Receive Buffer Current Address.

RXStsQBAdd

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSQBA															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSQBA															

Address:

0x8001_00A0 - Read/Write

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Receive Status Queue Base Address. The Receive Status Queue Base Address defines the system memory address of the receive status queue. This address is used by the MAC to reload the Receive Current Status Address whenever the end of the status queue is reached. The base address should be set at initialization time and must be set to a word aligned memory address.

Bit Descriptions:

RSQBA: Receive Status Queue Base Address.

Copyright 2007 Cirrus Logic

RXStsQBLen

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSQBL															

Address:

0x8001_00A4 - Read/Write

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Receive Status Queue Base Length. The Receive Status Queue Base Length defines the actual number of bytes in the receive status queue. The length should be set at initialization time and must define an integral number of receive statuses.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RSQBL: Receive Status Queue Base Length.

RXStsQCurLen

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSQCL															

Address:

0x8001_00A6 - Read/Write. Note half word alignment.

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Copyright 2007 Cirrus Logic

Definition:

Receive Status Queue Current Length. The Receive Status Queue Current Length defines the number of bytes between the Receive Status Current Address and the end of the receive status queue. This value is used internally to wrap the pointer back to the start of the queue. The register should not normally be written to.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RSQCL: Receive Status Queue Current Length.

RXStsQCurAdd

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSQCA															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSQCA															

Address:

0x8001_00A8 - Read/Write

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

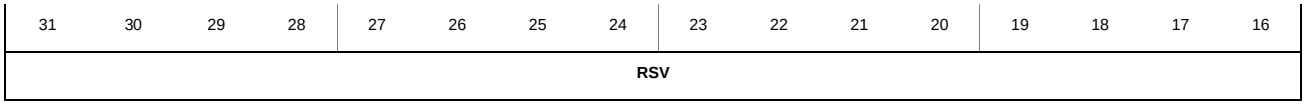
Receive Status Queue Current Address. The Receive Status Queue Base Address defines the system memory address of the receive status queue. This address is used by the MAC to reload the Receive Status Queue Current Status Address whenever the end of the status queue is reached. The base address should be set at initialization time and must be set to a word aligned memory address.

Bit Descriptions:

RSQCA: Receive Status Queue Current Address.

Copyright 2007 Cirrus Logic

RXStsEnq



Address:

0x8001_00AC - Read/Write

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Receive Status Enqueue register. The Receive Status Enqueue register is used to define the number of free entries available in the status queue. Only the Receive Status Increment field is writable and any value written to this field will be added to the existing Receive Status Value. Whenever complete statuses are written by the MAC, the Receive Status Value is decremented by the number read. For example, if the Receive Status Value is 0x07, and the Host writes 0x03 to the Receive Status Increment, the new Receive Status Value will be 0x0A. If the controller then reads two descriptors, the Value will be 0x08.

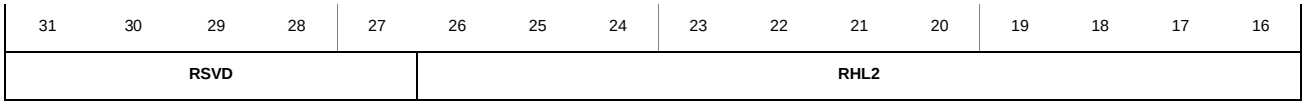
Bit Descriptions:

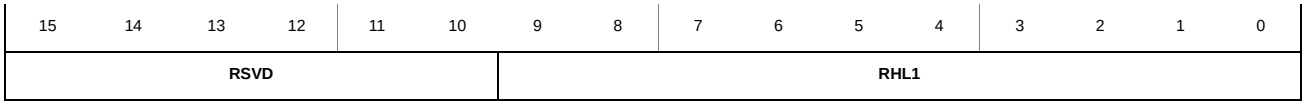
RSVD: Reserved. Unknown During Read.

RSV: Receive Status Value.

RSI: Receive Status Increment.

RXHdrLen





Address:

0x8001_00EC - Read/Write

Copyright 2007 Cirrus Logic

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Receive Header Length register. The Receive Header Length registers are used to generate status after receiving a specific portion of a receive frame. When the number of bytes specified in either register has been transferred to the external data buffer, an appropriate status is generated. The status for a receive header will reflect the number of bytes transferred for the current frame, the address match field will be valid, and the other status bits will be set to zero. A status will only be generated for header length 2 if the length is greater than that specified for header length 1.

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
RHL2:	Receive Header Length 2.
RHL1:	Receive Header Length 1.

Descriptor Processor Transmit Registers

TXDQBAdd

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDBA															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDBA															

Address:

0x8001_00B0 - Read/Write

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Transmit Descriptor Base Address register. The Transmit Descriptor Queue Base Address defines the system memory address of the transmit descriptor queue. This address is used by the MAC to reload the Transmit Current Descriptor Address whenever the end of the descriptor queue is reached. The base address should be set at initialization time and must be set to a word aligned memory address.

Copyright 2007 Cirrus Logic

9

Bit Descriptions:

TDBA: Transmit Descriptor Base Address.

TXDQBLen

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDBL															

Address:

0x8001_00B4 - Read/Write

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Transmit Descriptor Queue Base Length register. The Transmit Descriptor Queue Base Length defines the actual number of bytes in the transmit descriptor queue, which thereby sets the maximum number of transmit descriptors that can be supplied to the MAC at any one time. The length should be set at initialization time and must define an integral number of transmit descriptors.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

TDBL: Transmit Descriptor Base Length.

TXDQCurLen

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDCL															

Address:

0x8001_00B6 - Read/Write. Note half word alignment.

Chip Reset:

0x0000_0000

Copyright 2007 Cirrus Logic

Soft Reset:

Unchanged

Definition:

Transmit Descriptor Queue Current Length register. The Transmit Descriptor Queue Current Length defines the number of bytes between the Transmit Descriptor Current Address and the end of the transmit descriptor queue. This value is used internally to wrap the pointer back to the start of the queue. The register should not normally be written.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

TDCL: Transmit Descriptor Current Length.

TXDQCurAdd

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDCA															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDCA															

Address:

0x8001_00B8 - Read/Write

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Transmit Descriptor Queue Current Address register. The Transmit Descriptor Queue Current Address contains the pointer to the next memory location to be read from the transmit descriptor queue. This should be set at initialization time to the required starting point in the descriptor queue. During operation, the MAC will update this address following successful descriptor reads. Intermediate values in this register will not necessarily align to descriptor boundaries, nor directly effect the current descriptor in use because several descriptors may be buffered inside the MAC.

Bit Descriptions:

TDCA: Transmit Descriptor Current Address.

Copyright 2007 Cirrus Logic

TXDEnq

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDV															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								TDI							

Address:

0x8001_00BC - Read/Write

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Transmit Descriptor Enqueue register. The Transmit Descriptor Enqueue register is used to define the number of valid descriptors available in the transmit descriptor queue. Only the Transmit descriptor Increment field is writable and any value written to this field will be added to the existing Transmit Descriptor Value. When complete descriptors are read by the MAC, the Transmit Descriptor Value is decremented by the number read. For example if the Transmit Descriptor Value is 0x07, and the Host writes 0x03 to the Transmit Descriptor Increment, the new Value will be 0x0A. If the controller then reads two descriptors, the Value will be 0x08.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

TDV: Transmit Descriptor Value.

TDI: Transmit Descriptor Increment.

TXStsQBAdd

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSQBA															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSQBA															

Address:

0x8001_00C0 - Read/Write

Chip Reset:

Copyright 2007 Cirrus Logic

0x0000_0000

Soft Reset:

Unchanged

Definition:

Transmit Status Queue Base Address. The Transmit Status Queue Base Address defines the system memory address of the transmit status queue. This address is used by the MAC to reload the Transmit Current Status Address whenever the end of the status queue is reached. The base address should be set at initialization time and must be set to a word aligned memory address.

Bit Descriptions:

TSQBA: Transmit Status Queue Base Address.

TXStsQBLen

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSQBL															

Address:

0x8001_00C4 - Read/Write

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Transmit Status Queue Base Length. The Transmit Status Queue Base Length defines the actual number of bytes in the transmit status queue. The length should be set at initialization time and must define an integral number of transmit statuses.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

TSQBL: Transmit Status Queue Base Length.

Copyright 2007 Cirrus Logic

9

TXStsQCurLen

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSQCL															

Address:

0x8001_00C6 - Read/Write. Note half word alignment.

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Transmit Status Queue Current Length. The Transmit Status Queue Current Length defines the number of bytes between the Transmit Status Current Address and the end of the transmit status queue. This value is used internally to wrap the pointer back to the start of the queue. The register should not normally be written.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

TSQCL: Transmit Status Queue Current Length.

TXStsQCurAdd

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSQCA															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSQCA															

Address:

0x8001_00C8 - Read/Write

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Copyright 2007 Cirrus Logic

Definition:

Transmit Status Queue Current Address. The Transmit Status Queue Current Address contains the address being used to transfer transmit status to the queue. This register is available for debugging.

Bit Descriptions:

TSQCA: Transmit Status Queue Current Address.

RXBufThrshld

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD								RDHT						0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								RDST						0	0

Address:

0x8001_00D0 - Read/Write

Suggested Value:

0x0080_0040

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Receive Buffer Threshold register. The receive buffer thresholds are used to set a limit on the amount of receive data which is held in the receive data FIFO before a bus request will be scheduled. When the number of words in the FIFO exceeds the threshold value, the Descriptor Processor will schedule a bus request to transfer data. The actual posting of the bus request may be delayed due to lack of resources in the MAC, such as no active receive descriptor.

Note: There are other reasons to schedule bus transfers other than reaching the threshold. One of these is when an end of frame is received. The lower 2 bits of each threshold are always zero.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

0: Must be written as "0".

RDHT: Receive Data Hard Threshold.

Copyright 2007 Cirrus Logic

9

RDST: Receive Data Soft Threshold. The hard and soft threshold work in exactly the same manner except one. The soft threshold will not cause a bus request to be made if the bus is currently in use, but only when it is deemed to be idle (no transfers for four AHB clocks). The hard threshold takes effect immediately, regardless of the state of the bus. This operation allows for more efficient use of the AHB bus by allowing smaller transfers to take place when the bus is lightly loaded and requesting larger transfers only when the bus is more heavily loaded.

TXBufThrshld

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD								TDHT						0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								TDST						0	0

Address:

0x8001_00D4 - Read/Write

Suggested Value:

0020_0010

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Transmit Buffer Threshold register. The transmit buffer thresholds are used to set a limit on the amount of empty space allowed in the transmit FIFO before a bus request will be scheduled. When the number of empty words in the FIFO exceeds the threshold value, the Descriptor Processor will schedule a bus request to transfer data. The actual posting of the bus request may be delayed due to lack of resources in the MAC, such as no active transmit descriptor. The lower two bits of the thresholds are always zero.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

TDHT: Transmit Data Hard Threshold.

Copyright 2007 Cirrus Logic

TDST: Transmit Data Soft Threshold. The hard and soft threshold work in exactly the same manner except one. The soft threshold will not cause a bus request to be made if the bus is currently in use, but only when it is deemed to be idle (no transfers for four AHB clocks). The hard threshold takes effect immediately regardless of the state of the bus. This operation allows for more efficient use of the AHB bus by allowing smaller transfers to take place when the bus is lightly loaded and requesting larger transfers only when the bus is more heavily loaded.

RXStsThrshld

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD										RSHT				0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD										RSST				0	0

Address:

0x8001_00D8 - Read/Write

Suggested Value:

0x0004_0002

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Receive Status Threshold register. The receive status threshold are used to set a limit on the amount of receive status which is held in the receive status FIFO before a bus request will be scheduled. When the number of words in the FIFO exceeds the threshold value, the Descriptor Processor will schedule a bus request to transfer status. The actual posting of the bus request may be delayed due to lack of resources in the MAC, such as the RXStsEnq register being equal to zero. The lower two bits of the thresholds are always zero.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RSHT: Receive Status Hard Threshold.

RSST: Receive Status Soft Threshold.

Copyright 2007 Cirrus Logic

9

The hard and soft threshold work in exactly the same manner except one. The soft threshold will not cause a bus request to be made if the bus is currently in use, but only when it is deemed to be idle (no transfers for four AHB clocks). The hard threshold takes effect immediately regardless of the state of the bus. This operation allows for more efficient use of the AHB bus by allowing smaller transfers to take place when the bus is lightly loaded and requesting larger transfers only when the bus is more heavily loaded.

TXStsThrshld

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD										TSHT				0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD										TSST				0	0

Address:

0x8001_00DC - Read/Write

Suggested Value:

0x0004_0002

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Transmit Status Threshold register. The transmit status thresholds are used to set a limit on the amount of transmit status which is held in the transmit status FIFO before a bus request will be scheduled. When the number of words in the FIFO exceeds the threshold value, the Descriptor Processor will schedule a bus request to transfer status. The lower two bits of the thresholds are always zero.

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
0:	Must be written as "0".
TSHT:	Transmit Status Hard Threshold.
TSST:	Transmit Status Soft Threshold.

Copyright 2007 Cirrus Logic

The hard and soft threshold work in exactly the same manner except one. The soft threshold will not cause a bus request to be made if the bus is currently in use, but only when it is deemed to be idle (no transfers for four AHB clocks). The hard threshold takes effect immediately regardless of the state of the bus. This operation allows for more efficient use of the AHB bus by allowing smaller transfers to take place when the bus is lightly loaded and requesting larger transfers only when the bus is more heavily loaded.

RXDThrshld

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD										RDHT				0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD										RDST				0	0

Address:

0x8001_00E0 - Read/Write

Suggested Value:

0x0004_0002

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Receive Descriptor Threshold register. The receive descriptor thresholds are used to set a limit on the amount of empty space allowed in the MAC's receive descriptor FIFO before a bus request will be scheduled. When the number of empty words in the FIFO exceeds the threshold value, the Descriptor Processor will schedule a bus request to transfer descriptors. The actual posting of the bus request may be delayed due to lack of resources in the MAC, such as a RXDEnq equal to zero. The lower two bits of the thresholds are always zero.

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
0:	Must be written as "0".
RDHT:	Receive Status Hard Threshold.

Copyright 2007 Cirrus Logic

9

RDST: Receive Descriptor Soft Threshold.

The hard and soft threshold work in exactly the same manner except one. The soft threshold will not cause a bus request to be made if the bus is currently in use, but only when it is deemed to be idle (no transfers for four AHB clocks). The hard threshold takes effect immediately regardless of the state of the bus. This operation allows for more efficient use of the AHB bus by allowing smaller transfers to take place when the bus is lightly loaded and requesting larger transfers only when the bus is more heavily loaded.

TXDThrshld

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD										TDHT				0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD										TDST				0	0

Address:

0x8001_00E4 - Read/Write

Suggested Value:

0x0004_0002

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Transmit Descriptor Threshold register. The transmit descriptor thresholds are used to set a limit on the amount of empty space allowed in the MAC's transmit descriptor FIFO before a bus request will be scheduled. When the number of empty words in the FIFO exceeds the threshold value, the Descriptor Processor will schedule a bus request to transfer descriptors. The actual posting of the bus request may be delayed due to lack of resources in the MAC, such as a TXDEnq equal to zero. The lower two bits of the thresholds are always zero.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

0: Must be written as "0".

TDHT: Transmit Descriptor Hard Threshold.

Copyright 2007 Cirrus Logic

TDST: Transmit Descriptor Soft Threshold.

The hard and soft threshold work in exactly the same manner except one. The soft threshold will not cause a bus request to be made if the bus is currently in use, but only when it is deemed to be idle (no transfers for four AHB clocks). The hard threshold takes effect immediately regardless of the state of the bus. This operation allows for more efficient use of the AHB bus by allowing smaller transfers to take place when the bus is lightly loaded and requesting larger transfers only when the bus is more heavily loaded.

MaxFrmLen

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				MFL											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				TST											

Address:

0x8001_00E8 - Read/Write

Chip Reset:

0x0000_0000

Soft Reset:

Unchanged

Definition:

Maximum Frame Length and Transmit Start Threshold register.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

MFL: Maximum Frame Length. The maximum frame length is a limit for the amount of data permitted to be transferred across the AHB bus for a transmit frame, or on the wire for a receive frame. When this limit is reached for a transmit frame, the Transmit Descriptor Processor is halted and a transmit length error is set in the Interrupt Status register. When the limit is reached for a receive frame, no further data will be transferred to memory for the current frame. The status written for the frame will indicate the length error, and further frames will continue as normal, (the Receive Descriptor Processor will not halt).

Copyright 2007 Cirrus Logic

TST:	Transmit Start Threshold. The transmit start threshold defines the number of bytes that must be written to the transmit data FIFO before a frame will start transmission on the serial interface. This value is primarily of concern when the transmit frame is spread across multiple descriptors and the first descriptors define small amounts of data.

Copyright 2007 Cirrus Logic

Chapter 10

DMA Controller

10.1 Introduction

The DMA Controller can be used to interface streams from 20 internal peripherals to the system memory using 10 fully-independent programmable channels that consist of 5 Memory to Internal Peripheral (M2P) transmit channels and 5 Peripheral to Memory (P2M) receive channels.

The DMA Controller can also be used to interface streams from Memory to Memory (M2M), from Memory to Internal Peripheral (M2P), or from Memory to External Peripheral (M2P), using 2 dedicated M2M channels. External handshake signals are optionally available to support Memory to/from External Peripheral transfers (M2P/P2M). A software trigger is available for Memory to Memory transfers, and a hardware trigger is available for Memory to Internal Peripheral.

On the EP93xx chip the following peripherals may be allocated to the 10 channels.

- I^2S (which contains 3 Tx and 3 Rx DMA Channels)
- AAC (which contains 3 Tx and 3 Rx DMA Channels)
- UART1 (which contains 1 Tx and 1 Rx DMA Channels)
- UART2 (which contains 1 Tx and 1 Rx DMA Channels)
- UART3 (which contains 1 Tx and 1 Rx DMA Channels)
- IrDA (which contains 1 Tx and 1 Rx DMA Channels)

Each peripheral has it's own bi-directional DMA bus capable of transferring data in both directions simultaneously. All memory transfers take place via the main system AHB bus.

SSP and IDE can also use the M2M channels to send or receive data using their memory mapping to perform transfers.

SSPRx, SSPTx, and IDE have access to DMA M2M hardware transfer requests.

10.1.1 DMA Features List

DMA specific features are:

- Ten fully independent, programmable DMA controller internal M2P/P2M channels (5 Tx and 5 Rx).
- Two dedicated channels for Memory-to-Memory (M2M) and Memory-to-External Peripheral Transfers (external M2P/P2M).

Copyright 2007 Cirrus Logic

- Five hardware requests for M2M transfers; 2 for external peripherals that follow the handshake protocol, and 3 simple requests from IDE, SSPRx and SSPTx.
- Independent source and destination address registers. Source and destination can be programmed to auto-increment or not for Memory-to-Memory channels.
- Two buffer descriptors per M2P/P2M and M2M channel to avoid potential data underflow/overflow due to software introduced latency.
- For the internal M2P/P2M channels, buffer size is independent of the peripheral's packet size. Transfers can automatically switch between buffers.
- Per channel maskable interrupt generation.
- For DMA Data transfer sizes, byte, word and quad-word data transfers are supported using a 16-byte data bay. Programmable max data transfer size per M2M channel.
- Per-channel clock gating reduces power in channels which have not been enabled by software.

10.1.2 Managing Data Transfers Using a DMA Channel

A set of control and status registers are available to the system processor for setting up DMA operations and monitoring their status, and monitoring system interrupts generated when any of the DMA channels wish to inform the processor to update the buffer descriptor. The DMA controller can service 10 out of 20 possible peripherals using the 10 internal M2P/P2M DMA channels, each with its own peripheral DMA bus capable of transferring data in both directions simultaneously.

The UART1/2/3 and IrDA can each use two DMA channels, one for transmit and one for receive. The AC'97 interface can use six DMA channels (three transmit and three receive) to allow different sample frequency data queues to be handled with low software overhead. The I^2S interface can also use up to six DMA channels (three transmit and three receive) to allow up to six channels of audio out and six channels of audio in.

To perform block moves of data from one memory address space to another with minimum of program effort and time the DMA controller includes a memory-to-memory transfer feature. An M2M software trigger capability is provided. It can also fill a block of memory with data from a single location.

A hardware trigger is also provided for internal peripherals (IDE or SSP) or for external peripherals which don't use a handshaking protocol, to allow data streams between their internal memory location (or the SMC) and the system memory.

For byte or word wide peripherals, the DMA can be programmed to request byte- or word-wide AHB transfers respectively.

The transfer is completed when the Byte Count Register of the active buffer descriptor reaches zero. Status bits will indicate if the actual byte count is equal to the programmed limit. Completion of transfer will cause a DMA interrupt on that channel and rollover to the "other" buffer descriptor, if configured.

Copyright 2007 Cirrus Logic

The DMA controller memory-to-memory channels can also be used in "Memory to External Peripheral" mode with handshaking protocol. A set of external handshake signals **DREQ**, **DACK** and **TC/DEOT** are provided for each of 2 M2M channels.

- **DREQ** (input) can be programmed edge or level active, and active high or low. The peripheral may hold **DREQ** active for the duration of the block transfers or may assert/deassert on each transfer.
- **DACK** (output) can be programmed active high or low. **DACK** will cycle with each read or write, the timing is to coincide with the **nOE** or **nWE** of the EBI.
- **TC/DEOT** is a bidirectional signal, the direction and the active sense is programmable. When configured as an output, the DMA will assert **TC** (Terminal Count) on the final transfer to coincide with the **DACK**, typically when the byte count has expired. When configured as an input, the peripheral must assert **DEOT** concurrent with **DREQ** for the final transfer in the block.

Transfer is completed either on **DEOT** being asserted by the external peripheral or the byte count expiring. Status bits will indicate if the actual byte count is equal to the programmed limit, and also if the count was terminated by peripheral asserting **DEOT**. Completion of transfer will cause a DMA interrupt on that channel and rollover to the "other" buffer descriptor if configured.

For byte or word wide peripherals, the DMA will be programmed to request byte or word wide AHB transfers respectively. The DMA will not issue an AHB **HREQ** for a transfer until it has sampled **DREQ** asserted after **DACK** of the previous transfer has been asserted for the duration of the programmed wait states in the SMC (and possibly **DREQ** is sampled in the cycle **DACK** is deasserted).

10.1.3 DMA Operations

The operation of the DMA controller can be defined in terms of channel functionality. Two types of channels exist:

- Memory-to-Memory (M2M) channel
- Memory-to/from-Internal-Peripheral (M2P/P2M) channel.

10.1.3.1 Memory-to-Memory Channels

The two M2M channels support data transfers between:

- Memory locations which may be located in any accessible system memory banks.

  These memory to memory moves can be initiated by software, in which case the transfer will begin as soon as the channel is configured and enabled for memory to memory move. For this transfer type, the DMA first fills the internal 16-byte data bay by initiating read accesses on the source bus. It then empties the data from the data bay to the destination bus by initiating write accesses.

- Memory locations related to IDE or SSP.

Copyright 2007 Cirrus Logic

The transaction is initiated by a SSP or IDE request.This request is masked after each peripheral width transfer, in order to allow latency for the peripheral to deassert its request line.The transfer terminates when the Byte Count Register equals zero.

- Memory and External Bus.

  These can be memory- or FIFO-based and memory-mapped through the SMC. Working with peripheral devices may optionally use the external signals **DREQ, DACK** and **DEOT/TC** to control the data transfer using the following rules:

  - The peripheral sets a request for data to be read-from/written-to by asserting **DREQ**.
  - The peripheral transfers/samples the data when **DACK** is asserted.
  - To terminate the current transfer, depending on the programmed direction of **DEOT/TC**, the peripheral asserts **DEOT** coincident with **DREQ** or the DMA asserts TC coincident with **DACK**.

These data transfer handshaking signals are optional: if the external device doesn't use them, then the transfer will operate like an internal peripheral transfer. To support an external DMA peripheral, each request generates one peripheral-width DMA transfer. The M2M Channel 0 is dedicated to servicing External device 0 and the M2M Channel 1 is dedicated to servicing External device 1.

10.1.3.2 Memory-to-Peripheral Channels

The 5 M2P and 5 P2M channels support data transfers between Memory and Internal Peripherals (which are byte-wide). Five dedicated channels are available to transfer data between internal peripheral and memory (receive direction), and five channels are available to transfer data between memory and peripheral (transmit direction). Transfers are controlled using a REQ/ACK handshake protocol supported by each peripheral.

10.1.4 Internal M2P or P2M AHB Master Interface Functional Description

The AHB Master interface is used to transfer data between the system memory and the DMA Controller internal M2P/P2M channels in both receive and transmit directions as follows:

- In the receive direction, data is transferred to system memory from a packer unit.
- In the transmit direction, data is transferred from the system memory into the unpacker unit.

The AHB bus burst transfer size is a quad-word, that is, if the base memory address programmed into the BASEx register is quad-word aligned then a quad-word transfer either to memory from the 16-byte receive packer, or from memory to the 16-byte transmit packer is carried out.

The internal M2P **RxEnd** signals are asserted by the peripheral to indicate the end of received data or a receiver error. This causes the AHB master interface to write any valid data in the receive packer to main memory. If **RxEnd** signals an error in receive data, and if the ICE bit (Ignore Channel Error) is set, then the DMA continues transfers as normal. The **RxEnd** is asserted by the peripheral coincident with the last good data before the overrun

Copyright 2007 Cirrus Logic

occurred. If the ICE bit is not set, then the DMA flushes the last good data out to memory and terminates the transfer for the current buffer. Where whole words are present in the packer, word transfers are used. For the remaining bytes (up to a maximum of 3), byte transfers are used. Thus the maximum number of bus transfers performed to empty the packer is 6, that is, 3 word transfers and 3 byte transfers.

If the number of bytes transferred from a receive peripheral reaches the MaxTransfer count then this has the same effect as the **RxEnd** signals being asserted by the peripheral. The DMA controller asserts **RxTC** to the peripheral to indicate this condition.

The end of the transfer is signalled by the transfer count being reached, or by the peripheral. In the latter case, any data remaining in a packer unit is written to memory. Any data in an un-packer unit is considered invalid, and therefore discarded, as is data remaining in the transmit FIFO.

When a peripheral receive transfer is complete any data in the packer unit is written to memory. The data may not form a complete quad-word. If an incomplete quad-word is present, data is transferred to memory in either word or byte accesses. The number of valid bytes remaining to be transferred is used to control the type of access. If the number of bytes is 16, then a normal quad word write is performed. If the number of bytes is more than 4, then word accesses are performed until the number of bytes is less than 4. If the number of bytes is less than 4, then byte accesses are performed until the remainder of the data has been transferred.

If the peripheral ended the transfer with an error code, an interrupt is generated, and operation continues as normal using the next buffer descriptor (if it has been set up) to ensure that a minimal amount of data is lost. The point at which the transfer failed can be determined by reading the channel current address register for the last buffer. An example of an internal peripheral error code is the Transmit FIFO underflow error in the AAC.

10.1.5 M2M AHB Master Interface Functional Description

The AHB Master interface is also used to transfer data between either the system memory or external peripheral and the DMA Controller M2M channels in both receive and transmit directions.

10.1.5.1 Software Trigger Mode

When a M2M channel receives a software trigger and the buffer descriptor has been programmed, the AHB master interface begins to read data from memory into the data bay. When the DMA_MEM_RD state is exited (that is, data transfer to the data bay has finished) this causes the AHB master interface to write the data contained in the data bay to main memory. The data may not form a complete quad-word. If an incomplete quad-word is present, data is transferred to memory in either word or byte accesses. The number of valid bytes remaining to be transferred is used to control the type of access. If the number of bytes is 16, then a normal quad word write is performed. If the number of bytes is more than 4, then word accesses are performed until the number of bytes is less than 4. If the number of bytes is less than 4, then byte accesses are performed until the remainder of the data has been transferred.

Copyright 2007 Cirrus Logic

10.1.5.2 Hardware Trigger Mode for Internal Peripherals (SSP and IDE) and for External Peripherals without Handshaking Signals

When a M2M channel is set up to transfer to/from SSP, IDE or an external peripheral, the transfer width used (that is, the AMBA **HSIZE** signal) is determined by the peripheral width - programmed via the CONTROL.PW bits of the channel. This means that the transfers occur one at a time, as opposed to burst transfer operation for software triggered M2M. Thus the 16-byte data bay which is available for software triggered transfers is never fully utilized - at most 1 word of it is used (depending on PW bits).

10.1.5.3 Hardware Trigger Mode for External Peripherals with Handshaking Signals

When a M2M channel is set up to transfer to/from an external peripheral, the transfer width used (that is, the AMBA **HSIZE** signal) is determined by the peripheral width - programmed via the CONTROL.PW bits of the channel. This means that the transfers occur one at a time, as opposed to burst transfer operation for software triggered M2M. Thus the 16-byte data bay which is available for software triggered transfers is never fully utilized - at most 1 word of it is used (depending on PW bits).

10.1.6 AHB Slave Interface Limitations

The AHB slave interface is used to access all control and status registers.

The behavior of the AMBA AHB signals complies with the standard described in AMBA Specification (Rev 2.0) from ARM Limited. The DMA does not utilize the AHB slave split capabilities, so does not receive **HMASTER** or **HMASTERLOCK** and does not drive **HSPLIT**. It does not receive **HPROT** or **HRESP** and does not drive **HLOCK**.

10.1.7 Interrupt Interface

Each of the 12 DMA channels (10 M2P/P2M and 2 M2M) generates a single interrupt signal which is a combination of the interrupt sources for that channel. There are 3 interrupt sources, which are enabled in the channel control register (for both M2P/P2M and M2M): **DONE**, **STALL** and **NFB.**The interrupt signals are ORed before being transmitted to the **DMA_INT** output bus. Status of the interrupt bus is reflected in the DMA Global Interrupt Register (DMAGIInt). The status of each interrupt source per channel is found in the channel's interrupt register.

10.1.8 Internal M2P/P2M Data Unpacker/Packer Functional Description

The DMA controller transfers data to and from the system memory in four word bursts. The peripheral DMA bus protocol is used to transfer data to and from the peripherals as single bytes. In order to build the quad word bursts from the single bytes received from the peripheral, the DMA controller uses the Rx Burst Packers. To decompose the quad word bursts into byte transfers to the peripherals the Tx Burst Un-Packers are used.

Copyright 2007 Cirrus Logic

The data received on each of the five peripheral receive DMA Rx Data buses is transferred into an internal receive packer unit. The packer unit is used to convert the byte-wide data received from the peripheral into words to be transferred over the system bus to the memory. The packer unit stores 4 words (one quad-word) of data, which is the size of the burst transfers to and from memory over the system bus. Provision for the memory access latency is provided by FIFOs within the peripheral. The size of the FIFOs can be selected as appropriate for the data rate generated by the peripheral.

Transmit data is fetched from system memory by the AHB master interface and placed into the transmit un-packer. The transmit un-packer converts the quad-word burst of DMA data into byte data for transmission over the transmit peripheral DMA bus. The transmit un-packer contains 4 words (one quad-word) of storage. Additional latency is provided by FIFOs within the peripheral, the size of which can be selected as appropriate for the peripheral.

The number of data transfers over the peripheral DMA bus (that is, the number of bytes) are counted by packer/un-packer unit. If the number of bytes transferred reaches the MaxTransfer count, the appropriate **RxTC/TxTC** signal is asserted causing the flush to memory of data from a packer unit, and the invalidation of any data remaining in an un-packer unit.

10.1.9 Internal M2P/P2M DMA Functional Description

10.1.9.1 Internal M2P/P2M DMA Buffer Control Finite State Machine

Each DMA internal M2P/P2M channel is controlled by a finite state machine (FSM) which determines whether the channel is transferring data, and whether it is currently generating an interrupt.

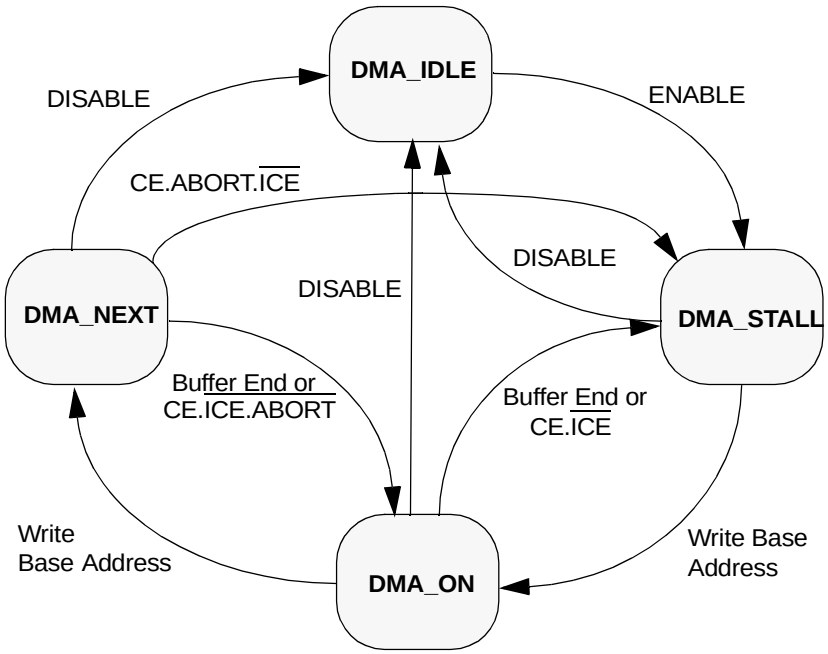


Figure 10-1. DMA M2P/P2M Finite State Machine

Copyright 2007 Cirrus Logic

CE: Channel (Peripheral) Error

ICE: CONTROL[6] - Ignore Channel Error. This bit may be set for data streams whereby the end user can tolerate occasional bit errors. If it is not set then the DMA will abort its transfer in receipt of a peripheral error.

ABORT: CONTROL[5]

10

10.1.9.1.1 DMA_IDLE

The DMA Channel FSM always resets to the DMA_IDLE state.

The DMA Channel FSM always enters the DMA_IDLE state when the channel is disabled (CONTROL[4]).

10.1.9.1.2 DMA_STALL

The DMA Channel FSM enters the DMA_STALL state when the channel enabled, no STALL interrupt is generated for this condition.

The DMA Channel FSM enters the DMA_STALL state if a memory buffer completes in the ON state. A DMA_STALL interrupt is generated for this condition.

The DMA Channel FSM enters the DMA_STALL state and terminates the current memory buffer if there is a peripheral error (**TxEnd/RxEnd** indication) while in the DMA_ON state, and ICE is not active.

The DMA Channel FSM enters the DMA_STALL state and terminates the current memory buffer if there is a peripheral error (**TxEnd/RxEnd** indication) while in the DMA_NEXT state, and ABORT is active, and ICE inactive. No STALL interrupt is generated for this condition.

No data transfers occur in this state.

10.1.9.1.3 DMA_ON

The DMA Channel FSM enters this state when a base address is written in the stall state.

Data transfers occur in this state.

The DMA Channel FSM enters this state when the current memory buffer expires, or when a peripheral error occurs that does not cause an abort, while in the DMA_NEXT state. The transition from DMA_NEXT to DMA_ON state results in a NFB interrupt being generated.

10.1.9.1.4 DMA_NEXT

The DMA Channel FSM enters this state when a base address register is written in the DMA_ON state (that is, for buffer Y). The DMA will continue to transfer using the buffer (that is, buffer X) that it began with in the DMA_ON state. When buffer X expires or when a peripheral error occurs, then the DMA will automatically switch over to using the next buffer (buffer Y). It will generate an interrupt (NFBint) to signal to the processor that it is switching over to a new buffer and that the old buffer descriptor (buffer X) is available to be updated.

Data transfers occur in this state.

Copyright 2007 Cirrus Logic

10.1.9.2 Data Transfer Initiation and Termination

The DMA Controller initiates data transfer in the receive direction when:

- A packer unit becomes full
- A packer unit, dependent on the next address access, contains enough data for an unaligned byte/word access.

The DMA Controller stops data transfers in the receive direction and moves onto the next buffer when:

- **RxEnd** signal is asserted to indicate end of received data or received error.

  No matter what the alignment up to now, this causes the AHB Master interface to write any valid data in the receive packer to main memory. If RxEnd signals the end of received data then all data which is present in the receive packer gets flushed to memory. If RxEnd signals an error in receive data, and if the ICE bit (Ignore Channel Error) is not set, then the erroneous byte is not written to memory. Only valid bytes are written. If ICE bit is set then the erroneous byte is written to memory. The DMA will update the Channel Status Register, generating a system interrupt which informs the processor that a new buffer needs to be allocated, and DMA will also indicate (NEXTBUFFER field) which pair of buffer descriptor registers (MAXCNTx, BASEx) should be used for the next buffer.

- The number of bytes transferred from a receive peripheral reaches MAXCNTx.

Note: This refers to bytes entering the data packer and not just data transmitted over the AHB bus (that is, has same effect as RxEnd signal generated by the peripheral). The DMA Controller asserts RxTC to the peripheral to indicate this condition. The DMA will update the Channel Status Register, generating a system interrupt, which informs the processor that a new buffer needs to be allocated and DMA will also indicate (NEXTBUFFER field) which pair of buffer descriptor registers (MAXCNTx, BASEx) should be used for the next buffer.

The DMA Controller initiates data transfers in the transmit direction when an Un-packer unit becomes empty.

The DMA Controller stops data transfer in the transmit direction when:

- **TxEnd** signal is asserted to indicate that the transfer is the last in the transmit data stream. Any data remaining in the Un-packer unit is considered invalid and flushed. At this point, the Channel Status Register will be updated and next buffer defined.
- **TxTC** signal asserted by DMA Controller to indicate to the peripheral that the transfer is the last as the byte count limit has been reached. At this point, the Channel Status Register will be updated and next buffer defined.
- Bursting across buffers cannot be carried out in either transmit or receive directions. The reason is that buffer pairs may not be contiguous, as required by HTRANS SEQ transfer type (where address = address of previous transfer + size in bytes).

Copyright 2007 Cirrus Logic

10

10.1.10 M2M DMA Functional Description

10.1.10.1 M2M DMA Control Finite State Machine

Each DMA M2M channel is controlled by 2 finite state machines (FSM) which determine whether the channel is transferring data to or from memory, which buffer from the double-buffer descriptor set it is using, and whether it is currently generating an interrupt.

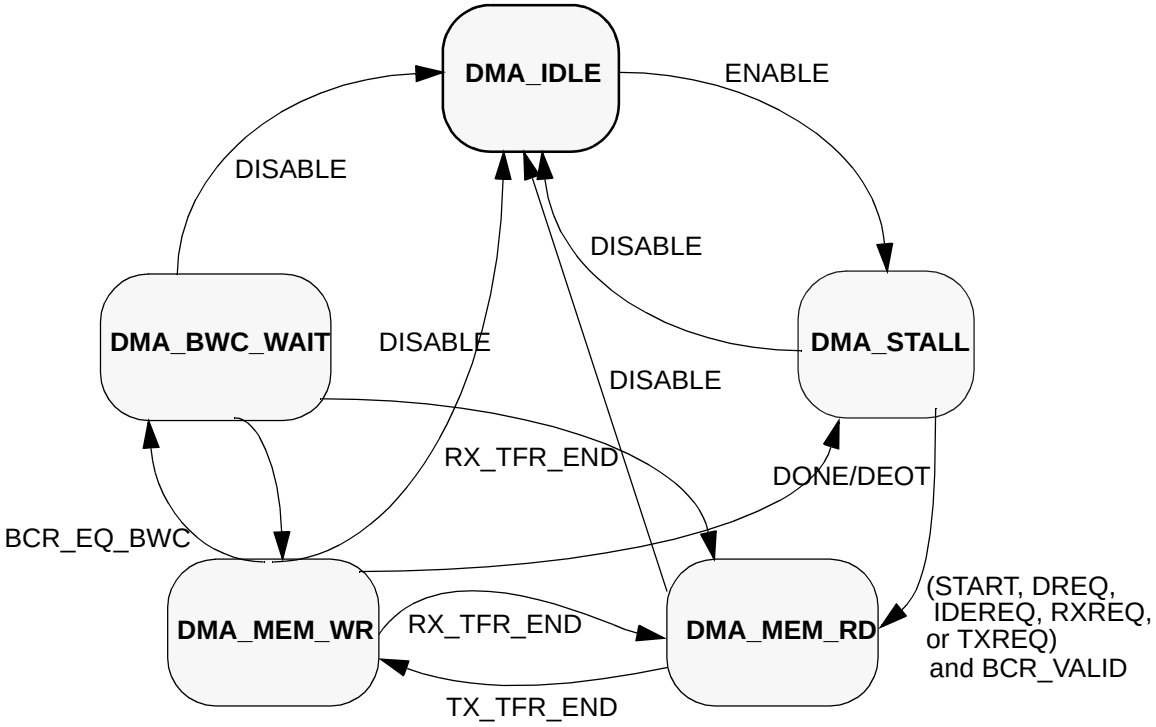


Figure 10-2. M2M DMA Control Finite State Machine

10.1.10.1.1 DMA_IDLE

The DMA M2M Control FSM always resets to the DMA_IDLE state.

The DMA Control M2M FSM always enters the DMA_IDLE state when a channel is disabled (CONTROL[3]).

The DMA Control M2M FSM exits the DMA_IDLE state and moves to the DMA_STALL state when the ENABLE bit of the CONTROL register is set.

10.1.10.1.2 DMA_STALL

The DMA M2M Control FSM enters the DMA_STALL state when an M2M channel is enabled. No STALL interrupt is generated for this condition.

The DMA M2M Control FSM enters the DMA_STALL state when a memory-to-memory transfer has completed successfully. The DONE and STALL interrupts are generated for this condition, if enabled.

Copyright 2007 Cirrus Logic

No data transfers occur in this state.

10.1.10.1.3 DMA_MEM_RD

The DMA M2M Control FSM enters the DMA_MEM_RD state when a M2M channel has received a software trigger to begin a transfer, that is, the START bit is set (CONTROL[4]) and CONTROL.TM = "00"; or when IDE or SSP asserts its request line and CONTROL.TM = "01" or "10"; or when an external device asserts its **DREQ** o/p to the DMA and CONTROL.TM = "01" or "10". At least one of the BCRx registers must contain a valid value, otherwise the DMA stays in the DMA_STALL state. For software triggered mode a valid BCR value is any non-zero value. For external DMA mode a valid BCR value depends on the peripheral width (programmed via the PW bits of the CONTROL register). For word/half-word/byte-wide peripherals the BCR value must be greater than or equal to four/two/one respectively.

The DMA M2M Control FSM enters the DMA_MEM_RD state when a memory write transfer has finished and the BCR register is still not equal to zero, that is, more data needs to be transferred from memory-to-memory. For external bus and IDE/SSP transfers, BCR not-equal-to 0 must be qualified with a **DREQ** before the DMA_MEM_RD state is entered again.

The DMA M2M Control FSM enters the DMA_MEM_RD state on exit from the DMA_BWC_WAIT state, if all the data present in the data bay had been transferred to memory when DMA_BWC_WAIT state was entered.

The DMA M2M Control FSM stays in this state until the data transfer from memory has completed for software trigger mode, that is, the data bay is filled with 16 bytes (or less depending on transfer size and BCR value etc.).

The DMA M2M Control FSM enters the DMA_MEM_RD state when the BCR register is equal to zero for the current buffer, and the other buffer descriptors BCR register has been programmed non-zero. DMA will proceed to do a memory read using the new buffer and the NFB interrupt is generated, if enabled.

Data transfers from memory or external bus/device (depending on the CONTROL.TM bits), occur in this state.

10.1.10.1.4 DMA_MEM_WR

The DMA M2M Control FSM enters the DMA_MEM_WR state when a memory read transfer has completed.

The DMA M2M Control FSM enters the DMA_MEM_WR state on exit from the DMA_BWC_WAIT state, if all the data present in the data bay had not been transferred to memory when DMA_BWC_WAIT state was entered.

The DMA M2M Control FSM stays in this state until the data transfer to memory has completed, that is, the data bay is emptied.

Data transfers, to memory or external peripheral (depending on the CONTROL.TM bits), occur in this state.

Copyright 2007 Cirrus Logic

10.1.10.1.5 DMA_BWC_WAIT

The DMA M2M Control FSM enters the DMA_BWC_WAIT state when the byte count is within 15 bytes of a multiple of the BWC value.

The DMA M2M Control FSM stays in this state for one cycle only.

10.1.10.2 M2M Buffer Control Finite State Machine

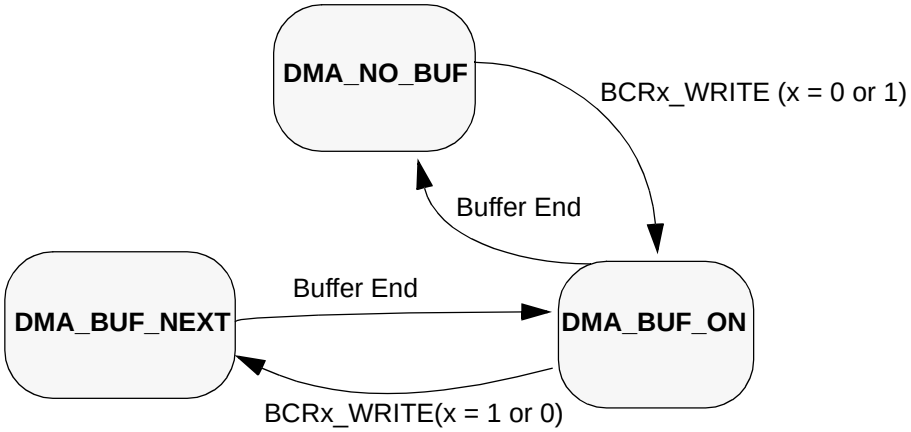


Figure 10-3. M2M DMA Buffer Finite State Machine

10.1.10.2.1 DMA_NO_BUF

The DMA M2M Buffer FSM resets to the DMA_NO_BUF state. This state reflects that no buffer descriptor has as yet been programmed in the DMA controller.

The DMA M2M Buffer FSM exits this state when one of the BCRx (x = 0 or 1) registers is programmed. If BCR0 is written to, then the FSM moves to the DMA_BUF_ON state and buffer0 becomes the active buffer available for a transfer. If BCR1 is written to then the FSM moves to the DMA_BUF_ON state and buffer1 becomes the active buffer available for a transfer.

10.1.10.2.2 DMA_BUF_ON

The DMA Buffer FSM enters the DMA_BUF_ON state from the DMA_NO_BUF state when one of the BCRx registers is written to.

The DMA Buffer FSM enters the DMA_BUF_ON state from the DMA_BUF_NEXT state when the transfer from the active buffer has ended. This end-of-buffer can be due to the BCRx register value reaching zero, or receipt of a **DEOT** input from the external device (when in external DMA transfer mode and **DEOT** is configured as an input signal to the DMA).

Data transfers to or from memory or external bus can occur in the DMA_BUF_ON state. When the DMA Buffer FSM transitions from DMA_BUF_NEXT to DMA_BUF_ON state, the NFB (Next Frame Buffer) interrupt is generated. This signals to software that rollover is occurring to the other buffer and also that one of the BCRx registers is now free for update

Copyright 2007 Cirrus Logic

(which BCRx is free can be determined using the STATUS.Nextbuffer status bit - see "STATUS" on page 10-37).

When the DMA Buffer FSM transitions from DMA_BUF_ON to DMA_NO_BUF state due to end of buffer, the DONE status bit is asserted and the DONE interrupt is set if enabled. The **TC** (Terminal Count) output is asserted by the DMA to the external device if the BCR register has expired for the current buffer (when in external DMA transfer mode and **TC** is programmed as an output signal from the DMA). The end of buffer can also be due to receipt of a **DEOT** input from the external device (when in external DMA transfer mode and **DEOT** is configured as an input signal to the DMA). The TCS and EOTS status bits of the STATUS register indicate what caused the end of buffer.

10.1.10.2.3 DMA_BUF_NEXT

The DMA Buffer FSM enters the DMA_BUF_NEXT state from the DMA_BUF_ON state when a write occurs to the second of the BCRx registers (that is, the BCRx register that was not written to when in the DMA_NO_BUF state).

The DMA Buffer FSM stays in this state until the transfer using the active buffer has ended, either as a result of BCRx reaching zero or due to receipt of a **DEOT** input from the external device (when in external DMA transfer mode and **DEOT** is configured as an input signal to the DMA). The TCS and EOTS status bits of the STATUS register indicate what caused the end of buffer.

Data transfers to/from memory or external device can occur in the DMA_BUF_NEXT state.

When the DMA Buffer FSM transitions from DMA_BUF_NEXT to DMA_BUF_ON state as a result of the BCR count expiring, the **TC** (Terminal Count) output is asserted by the DMA to the external device to indicate that the BCR register has expired for the current buffer (when in external DMA transfer mode and **TC** is programmed as an output signal from the DMA).

When the DMA Buffer FSM transitions from DMA_BUF_NEXT to DMA_BUF_ON state, the NFB (Next Frame Buffer) interrupt is generated (if enabled). This signals that one of the buffer descriptors is now free for update. For example the following sequence of events could occur:

- BCR0 is programmed => move to DMA_BUF_ON state.
- BCR1 is programmed => move to DMA_BUF_NEXT state.
- Channel is enabled => transfers begin using Buffer0.
- Buffer0 transfer ends => move to DMA_BUF_ON state and begin transfers with Buffer1.
- NFB interrupt is generated when FSM moves to DMA_BUF_ON state, signalling that
- Buffer0 is now free for update.

10.1.10.3 Data Transfer Initiation

Memory-to-memory transfers require a read-from and a write-to memory to complete each transfer.

Copyright 2007 Cirrus Logic

The DMA Controller initiates memory-to-memory transfers in the receive direction (that is, from memory/peripheral to DMA) under the following circumstances:

- A channel has been triggered by software, that is, setting the START bit to "1". Setting the START bit causes the channel to begin requesting the bus, and when granted ownership it will start transferring data immediately. The DMA controller drives the SAR_BASEx value onto the internal AHB address bus. If CONTROL.SCT is not set, the SAR_BASEx increments by the appropriate number of bytes upon a successful read cycle. The DMA initiates the write portion of the transfer when the appropriate number of read cycles is completed, that is, either when the 16-byte data bay has been filled, or when it contains the number of bytes (less than 16) that remain to be transferred, or when it contains sufficient data for an unaligned byte/word access (dependant on the next address access).
- A channel receives a transfer request from SSP or IDE or an external device without handshaking signals (that is, CONTROL.NO_HDSK = "1"), and the transfer mode is set to be either memory-to-external bus mode or external device-to-memory mode (that is, CONTROL.TM = "01"/"10" respectively). The DMA drives the SAR_BASEx value onto the address bus and requests a transfer size equal to the programmed peripheral width. In the case of CONTROL.TM = "10" where the external device (which is the source for the data) is FIFO-based, it is up to software to program the SAH bit correctly (Source Address Hold), so that on successive transfers from the peripheral, the SAR_CURRENTx value will not increment, thus reflecting the FIFO-nature of the peripheral.
- A channel receives a request from an external device and the transfer mode is set to be either memory-to-external device mode or external device-to-memory mode (that is, CONTROL.TM = "01" or "10" respectively). The DMA drives the SAR_BASEx value onto the address bus and requests a transfer size equal to the programmed peripheral width. In the case of CONTROL.TM = "10" where the external device (which is the source for the data) is FIFO-based, it is up to software to program the SAH bit correctly (Source Address Hold), so that on successive transfers from the peripheral, the SAR_CURRENTx value will not increment, thus reflecting the FIFO-nature of the peripheral.
- When the current transfer terminates the DMA will check if the BCR register for the "other" buffer (of the double-buffer set) has been programmed. If BCR is non-zero and CONTROL.TM = "00", that is, software trigger mode, then the DMA will proceed immediately to request the AHB bus and begin a transfer from memory to DMA using the other buffer descriptor. Software does not need to reprogram the START bit, it is enough to have the second buffer descriptor set up while the first buffer transfer is in progress. In the case where TM is such that external-device mode is set up, then rollover to the other buffer will also occur if the current transfer terminates, but the DMA will wait until it receives a **DREQ** from the external peripheral before initiating a transfer.

The DMA Controller initiates memory-to-memory transfers in the transmit direction (that is, from DMA to memory/external bus) under the following circumstances:

Copyright 2007 Cirrus Logic

- For a software-triggered M2M transfer, a memory-write is initiated when the 16-byte data bay has been filled (in the case where 16 or more bytes remain to be transferred) or when it contains the appropriate number of bytes (equal to BCR register value if BCR is less than 16). The DMA controller drives the DAR_BASEx onto the address bus. This address can be any aligned byte address. The BCR register decrements by the appropriate number of bytes. When BCR = 0 then the transfer is complete. If BCR is greater than zero, another read/write transfer is initiated.
- For transfers involving external devices or SSP/IDE, the DMA memory-write phase is initiated when the data bay contains the byte/half-word/word data, depending on PW value, that is, peripheral width. The DMA will then drive the DAR_BASEx onto the address bus and will set the AMBA **HSIZE** signal in accordance with the PW value. Once the DMA has received confirmation that the write is done (from **HREADY** in case of an internal memory write, or from the SMC acknowledge signal in case of an external device write), a wait state counter is started. During the count, the hardware request line is masked, in order to allow the related peripheral to de-assert its request. In the case of CONTROL.TM = "01" and the external device (which is the destination for the data) is FIFO-based, it is up to software to program the DAH bit correctly (Destination Address Hold), so that on successive transfers to the peripheral, the DAR_CURRENTx value will not increment, thus reflecting the FIFO-nature of the peripheral.

10.1.10.4 Data Transfer Termination

The DMA Controller terminates a memory-to-memory channel transfer under the following conditions:

- For software-triggered transfers which use a single buffer, the transfer is terminated when the BCR register of the active buffer has reached zero. The DONE status bit and corresponding interrupt (if enabled) are set. In the case of double/multiple buffer transfers, termination occurs when the BCR registers of both buffer descriptors has reached zero. The DONE status bit and corresponding interrupt (if enabled) are set. When the DONE interrupt is set the processor can then write a one to clear the interrupt before reprogramming the DMA to carry out another M2M transfer.
- For hardware-triggered transfers involving SSP or IDE or external devices without handshaking signals, the transfer is also terminated when the BCR register of the active buffer has reached zero. The DONE status bit and corresponding interrupt (if enabled) are set. When the DONE interrupt is set, the processor can then write a one to clear the interrupt before reprogramming the DMA to carry out another external DMA transfer.
- For operations involving external devices using a single buffer, the transfer is terminated on the first occurrence of **DEOT** being asserted by the device or the byte count expiring for the active buffer. In the case of the DMA receiving a **DEOT** from the peripheral (which is aligned to **DREQ**) the DMA knows that this is the final transfer to be performed. The DONE status bit and corresponding interrupt (if enabled) are set. In the case of double/multiple buffer transfers, termination occurs on either the occurrence of the DMA receiving a **DEOT** from the device while it is transferring to/from the last buffer (that is, no other buffer has been set up), or when the BCR registers of both buffer descriptors has reached zero.

Copyright 2007 Cirrus Logic

When the DONE interrupt is set, the processor can then write a one to clear the interrupt before reprogramming the DMA to carry out another external DMA transfer. If the **DEOT_TC** pin is configured as an output pin (**TC**), the DMA asserts **TC** when each buffers byte count expires. It then rolls over to the other buffer. If the **DEOT_TC** pin is configured as an input pin (**DEOT**), the DMA terminates transfers from the active buffer when **DEOT** is asserted and rolls over to the other buffer. The DONE interrupt is not asserted when the DMA has another buffer available to which it can roll over. However the NFB interrupt is generated when the rollover occurs.

10.1.10.5 Memory Block Transfer

The DMA Controller M2M channels provide a feature whereby block moves of data from one memory location can occur. If the CONTROL.SCT register bit is set for a channel, then its source address will not increment. In order to use this feature, both the source and destination addresses must be word-aligned, thus facilitating the transfer of a word of data from 1 location to a block of memory with the number of destination memory addresses written to is determined by the byte count register. For example, to copy a word to 10 consecutive destination addresses, then BCR must be set to 40.

10.1.10.6 Bandwidth Control

The Bandwidth Control feature makes it possible to force the DMA off the AHB bus during M2M transfers, to allow access to another device/peripheral. CONTROL.BWC register bits provide 12 levels of block transfer sizes. If the BCR decrements to within 15 bytes of a multiple of the decode of BWC, then the DMA bus request is negated until the bus cycle terminates, to allow the AHB bus arbiter to switch masters.

If BWC is equal to zero, then the bus request stays asserted until BCR = zero, that is, the transfer is finished. If the initial value of BCR is equal to the BWC decode, the bus request will not be negated straight away. Some data must first be transferred.

10.1.10.7 External DMA Request (DREQ) Mode

When the external device requires DMA service, it asserts **DREQ**, which may be configured as either edge or level sensitive using bit DREQP[1] of the CONTROL register.

External DMA requests are processed as follows:

- In level-sensitive mode, the external device requests service by asserting **DREQ** and leaving it asserted as long as it needs service. The DMA synchronizes the **DREQ** input using 2 HCLK flip-flops for metastability protection. To prevent another transfer from taking place, the external device must deassert the **DREQ** pin during the **DACK** (DMA Acknowledge) cycle. The number of cycles that **DACK** is asserted is governed by the number of wait states in the Static Memory Controller.
- For external devices that generate a pulsed signal for each transfer, edge-sensitive mode should be used. When the DMA detects a rising/falling edge on **DREQ** (as configured by bit DREQP[0] of the CONTROL register), a request becomes pending. The DMA synchronizes the latched **DREQ** input using 2 HCLK flip-flops for metastability protection. The DREQS status bit is set to indicate that a request is pending.

Copyright 2007 Cirrus Logic

Subsequent changes on **DREQ** are ignored until the pending request begins to be serviced. When the pending request has begun to be serviced, the DREQS status bit is cleared and subsequent edge-triggered requests are again recognized (latched) by the DMA. The DREQS status bit can be cleared by a software write to the channel STATUS register, thus causing the DMA to ignore the request.

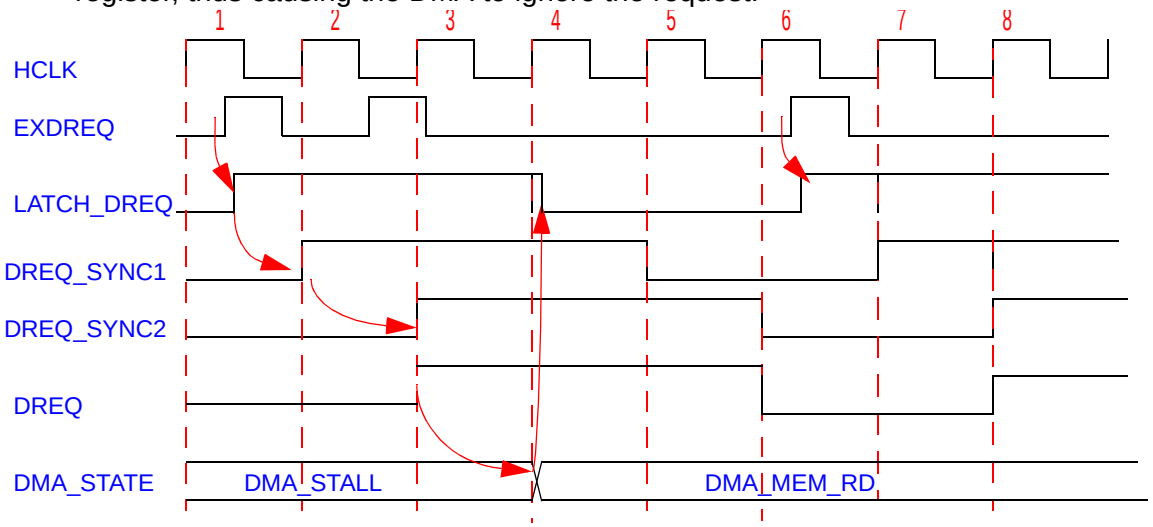


Figure 10-4. Edge-triggered DREQ Mode

1. A **DREQ** rising edge (**DREQ** is active high) is latched onto **LATCH_DREQ** during cycle 1.
2. This signal is synchronized using 2 HCLK flip-flops. The DREQS status bit indicates a request is pending at start of cycle 3.
3. The DMA state machine moves into the DMA_MEM_RD state to begin servicing the first request in cycle 4.
4. The **DREQ** latch is reset as a result of this state change and 2 cycles later the DREQS status bit is cleared.
5. A second request cannot be recognized until DREQS is cleared. Hence the request received during cycle 2 is ignored by the DMA.
6. A rising edge on **DREQ** during cycle 6 is latched and causes the DREQS status bit to be set again, thus indicating that another external peripheral request is pending.

10.1.11 DMA Data Transfer Size Determination

10.1.11.1 Software Initiated M2M and M2P/P2M Transfers

Data transfer size flexibility is guaranteed by allowing the start address of a DMA transfer to be aligned to any arbitrary byte boundary since this is the case for the 10 internal byte-wide M2P/P2M channels and for the 2 M2M channels when used in software initiated mode.

Copyright 2007 Cirrus Logic

At the start of a receive or transmit data transfer, the AHB Master Interface uses the low order 4 bits of the current DMA address to decide on the data transfer size to use. If the low-order 4 bits are zero, the first transfer is a quad word access. If they are not all zero, then if the low-order two bits are zero, then the first transfer is a word transfer. Word transfers will continue, and the current address incremented each time by one word, until the low-order address bits indicate that the address is quad-word aligned. If the start address is not word aligned, then the first transfer is a byte transfer, and the current address is incremented by one byte each time until the current address is word aligned. Transfers will then be performed as word transfers until the address is quad-word aligned. (Unless the address becomes quad-word aligned immediately, in which case quad word transfers are used). Note that in the case of the M2M channels, source address alignment takes precedence over destination address alignment. This means that if the source is aligned on a quad-word boundary and the destination address is aligned on a byte boundary, the channel will burst data into the data bay and then perform byte transfers to the destination.

The maximum transfer count can be any arbitrary number of bytes.

Table 10-1. Data Transfer Size

Current DMA Addr Bits [3:0]	Transfer Type
0000	Quad-Word access (unless there are less than 4 word addresses remaining)
0100,1000,1100	Word access
xx01, xx10, xx11	Byte access

The DMA Controller transfers data when it owns the AHB bus. Note that with byte/word/quad-word scheme that the DMA Controller employs, it can never burst across a 1KB boundary. The reason is that the DMA Controller only bursts when the 4 LSB Address bits are 0000b. A 1 KB boundary has the LSB 10 Address bits being zero. (ref: ARM AMBA Specification).

10.1.11.2 Hardware-Initiated M2M Transfers

The data transfer size for DMA transfers to/from external devices or SSP/IDE is dictated by the peripheral width. For byte, half-word or word wide peripherals, the DMA is programmed, using the PW bits of a channels control register, to request byte, half-word or word wide transfers respectively. Each external device request generates one peripheral width DMA transfer. If the memory involved is narrower than the peripheral then multiple memory accesses may be needed, for example, a word wide peripheral transferring to byte wide memory requires 4 memory transfers. The memory controller handles the generation of multiple memory accesses if necessary (and not the DMA).

10.1.12 Buffer Descriptors

A "buffer" refers to the area in system memory that is characterized by a buffer descriptor, that is, a start address and the length of the buffer in bytes.

Copyright 2007 Cirrus Logic

10.1.12.1 Internal M2P/P2M Channel Rx Buffer Descriptors

Only one Rx buffer descriptor is allocated per transaction. There are five Rx buffer descriptors, one for each of the five receive channels. Each buffer descriptor allows a channel double buffering scheme by containing programming for two buffers, that is, two system buffer base addresses and two buffer byte counts. This ensures that there is always one free buffer available for transfers to avoid potential data over/under-flow due to software-introduced latency.

10.1.12.2 Internal M2P/P2M Channel Tx Buffer Descriptors

Only one Tx buffer descriptor is allocated per transaction. There are five Tx buffer descriptors, one for each of the five transmit channels Each buffer descriptor allows a channel double buffering scheme by containing programming for two buffers, that is, two system buffer base addresses and two buffer byte counts. This ensures that there is always one free buffer available for transfers to avoid potential data over/under-flow due to software introduced latency.

10.1.12.3 M2M Channel Buffer Descriptors

Only one M2M channel buffer descriptor is allocated per transaction. There are two M2M buffer descriptors, one for each of the 2 M2M channels. Each buffer descriptor allows a channel double buffering scheme by containing programming for two buffers, that is, two source base addresses, two destination base addresses and two buffer byte counts. The buffers are limited to 64 kBytes (0xFFF). This ensures that there is always one free buffer available for transfers which avoids potential data overflow/underflow due to software introduced latency.

10.1.13 Bus Arbitration

When ready to do a transfer, the DMA Controller arbitrates internally between DMA Channels, then requests AHB bus access to the external AHB bus arbiter. Then a default setting of M2P having a higher priority than M2M is implemented. The default setting is programmable and can be changed if required (DMA Arbitration register bit[0] = CHARB).

The channel arbitration scheme is based on rotating priority, the order is as shown below in Table 10-2:

Table 10-2. M2P DMA Bus Arbitration

Internal Arbitration Priority		
	CHARB = 0	**CHARB = 1**
Highest	M2P Ch 0	M2M Ch 0
	M2P Ch 1	M2M Ch 1
	M2P Ch 2	M2P Ch 0
	M2P Ch 3	M2P Ch 1
	M2P Ch 4	M2P Ch 2
	M2P Ch 5	M2P Ch 3
	M2P Ch 6	M2P Ch 4

Copyright 2007 Cirrus Logic

Table 10-2. M2P DMA Bus Arbitration (Continued)

Internal Arbitration Priority		
	CHARB = 0	**CHARB = 1**
	M2P Ch 7	M2P Ch 5
	M2P Ch 8	M2P Ch 6
	M2P Ch 9	M2P Ch 7
	M2M Ch 0	M2P Ch 8
Lowest	M2M Ch 1	M2P Ch 9

During normal operation, using the "fair" rotating priority scheme shown in Table 10-2, the last channel to be serviced becomes the lowest priority channel with the others rotating accordingly. In addition, any device requesting service is guaranteed to be recognized after no more than eleven higher priority services has occurred. This prevents any one channel from monopolizing the system. When the bus is idle, the scheme reverts to a fixed priority whereby the highest priority request gets in first (as shown in Table 10-2) when the bus resumes to normal operation.

In the case where the two M2M channels are requesting a service, the [PW] size of the read or write transfers for the first channel are completed before the read transfer for the second channel begins. See subsections under Section 10.1.5 for detailed information about handshaking protocols for hardware and software-triggered M2M channel transfers.

10.2 Registers

10.2.1 DMA Controller Memory Map

Table 10-3 defines the DMA Controller mapping for each of 10 M2P (memory-to-peripheral) channels (5 Tx and 5 Rx), plus the 2 M2M (memory-to-memory) channels.

Before programming a channel, the clock for that channel must be turned on by setting the appropriate bit in the PwrCnt register of the Clock and State Controller block.

Table 10-3. DMA Memory Map

ARM920T Address	Description	Channel Base Address
0x8000_0000 -> 0x8000_003C	M2P Channel 0 Registers (Tx)	0x8000_0000
0x8000_0040 -> 0x8000_007C	M2P Channel 1 Registers (Rx)	0x8000_0040
0x8000_0080 -> 0x8000_00BC	M2P Channel 2 Registers (Tx)	0x8000_0080
0x8000_00C0 -> 0x8000_00FC	M2P Channel 3 Registers (Rx)	0x8000_00C0
0x8000_0100 -> 0x8000_013C	M2M Channel 0 Registers	0x8000_0100
0x8000_0140 -> 0x8000_017C	M2M Channel 1 Registers	0x8000_0140
0x8000_0180 -> 0x8000_01BC	Not Used	
0x8000_01C0 -> 0x8000_01FC	Not Used	
0x8000_0200 -> 0x8000_023C	M2P Channel 5 Registers (Rx)	0x8000_0200
0x8000_0240 -> 0x8000_027C	M2P Channel 4 Registers (Tx)	0x8000_0240
0x8000_0280 -> 0x8000_02BC	M2P Channel 7 Registers (Rx)	0x8000_0280
0x8000_02C0 -> 0x8000_02FC	M2P Channel 6 Registers (Tx)	0x8000_02C0
0x8000_0300 -> 0x8000_033C	M2P Channel 9 Registers (Rx)	0x8000_0300

Copyright 2007 Cirrus Logic

Table 10-3. DMA Memory Map

ARM920T Address	Description	Channel Base Address
0x8000_0340 -> 0x8000_037C	M2P Channel 8 Registers (Tx)	0x8000_0340
0x8000_0380	DMA Channel Arbitration register	
0x8000_03C0	DMA Global Interrupt register	
0x8000_03C4 -> 0x8000_FFFC	Not Used	0x8000_03C4

10.2.2 Internal M2P/P2M Channel Register Map

The DMA Memory Map above includes the base address mapping for the channel registers for each of the 10 M2P/P2M channels that are shown in the following table, the Internal M2P/P2M Channel Register Map. This mapping is common for each channel thus offset addresses from the bases in Table 10-3 are shown in Table 10-4.

Table 10-4. Internal M2P/P2M Channel Register Map

Offset	Register Name	Access	Bits	Reset Value
Channel Base Address + 0x0000	"CONTROL"	R/W	6	0
Channel Base Address + 0x0004	"INTERRUPT"	R/W TC *	3	0
Channel Base Address + 0x0008	"PPALLOC"	R/W	4	Channel dependant (see register description)
Channel Base Address + 0x000C	"STATUS"	RO	8	0
Channel Base Address + 0x0010	Reserved			
Channel Base Address + 0x0014	"REMAIN"	RO	16	0
Channel Base Address + 0x0018	Reserved			
Channel Base Address + 0x001C	Reserved			
Channel Base Address + 0x0020	"MAXCNTx"	R/W	16	0
Channel Base Address + 0x0024	"BASEx"	R/W	32	0
Channel Base Address + 0x0028	"CURRENTx"	RO	32	0
Channel Base Address + 0x002C	Reserved			
Channel Base Address + 0x0030	"MAXCNTx"	R/W	16	0
Channel Base Address + 0x0034	"BASEx"	R/W	32	0
Channel Base Address + 0x0038	"CURRENTx"	RO	32	0
Channel Base Address + 0x003C	Reserved			

Note:See Table 10-3 for Channel Base Addresses

Note: - write this location once to clear the interrupt (see Interrupt register description for which bits this rule applies to).*

Copyright 2007 Cirrus Logic

Register Descriptions

10

CONTROL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									ICE	ABORT	ENABLE	ChErrorIntEn	RSVD	NFBIntEn	STALLIntEn

Address:

Channel Base Address + 0x0000 - Read/Write

Definition:

This is the Channel Control Register, used to configure the DMA Channel.

Important Programming Note: The control register should be read immediately after being written. This action will allow hardware state machines to transition and prevent a potential problem when the registers are being written in back to back clock cycles.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

STALLIntEn: Setting this bit to 1 enables the generation of the STALL interrupt in the STALL State of the DMA Channel State machine. Setting this bit to zero disables generation of the STALL Interrupt.

NFBIntEn: Setting this bit to 1 enables the generation of the NFB (next frame buffer) interrupt in the ON State of the DMA Channel State machine. Setting this bit to zero disables generation of the NFB Interrupt. Normally when the channel is enabled, this bit should be 1. However in the case where the current buffer is the last, then this bit can be cleared to prevent the generation of an interrupt while the DMA State machine is in the ON State.

ChErrorIntEn: Setting this bit to 1 enables the ChError Interrupt which indicates if the buffer transfer occurred with an error.

ENABLE: Setting this bit to 1 enables the channel, clearing this bit disables channel, and causes the remaining unpacker/packer data to be discarded. The channel must always be enabled before writing the Base address register.

Copyright 2007 Cirrus Logic

ABORT: This bit determines how the DMA Channel State machine behaves while in the NEXT state and in receipt of a peripheral error, indicated on **RxEnd/TxEnd**. This bit is ignored when ICE is set.
0 - NEXT -> ON state, effectively ignoring the error.
1 - NEXT -> STALL state, effectively disabling the channel. No STALLInt interrupt is set for this condition.

ICE: Ignore Channel Error bit. Setting this bit results in suppression of the generation of the ChErrorInt interrupt and does not result in buffer termination. This bit may be set for data streams whereby the end user is tolerant to occasional bit errors.

PPALLOC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												PPALLOC			

Address:

Channel Base Address + 0x0008 - Read/Write

Definition:

This is the Peripheral Port Allocation register used to configure the internal M2P channel programmability. It is possible to program a channels use on one of a number of different peripherals.

There can be 20 external peripherals - 10 Tx and 10 Rx - connected to the 20 "ports" of the DMA. The 10 internal M2P DMA channels can serve 10 of these ports at one time.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

Note: PPALLOC:Table 10-5, Table 10-6, and Table 10-7 give the PPALLOC decode for the port allocation for both a transmit channel and a receive channel.

Two channels cannot be programmed to serve the same port since, in the case of an erroneous software write operation, the lower channel number is given priority. For example, if software writes the value 0x01 to Channel 0 Tx PPALLOC[3:0], and also writes this same value to Channel 2 Tx PPALLOC[3:0], then the Channel 0 Tx will be configured for Port 0 and Channel 2 will not function correctly.

The PPALLOC register must be written to before a channel is enabled. If this is not done, then the default allocation of the ports will be used.

Copyright 2007 Cirrus Logic

NOTE: The naming convention used for channels and ports is as follows - even numbers correspond to transmit channels/ports and odd numbers correspond to receive channels/ports.

Table 10-5. PPALLOC Register Bits Decode for a Transmit Channel

Ch 0, 2, 4, 6, 8 PPALLOC[3:0]	Port allocated	Peripheral Allocated
0000	PORT 0	I2S1 Tx
0001	PORT 2	I2S2 Tx
0010	PORT 4	AAC1 Tx
0011	PORT 6	AAC2 Tx
0100	PORT 8	AAC3 Tx
0101	PORT 10	I2S3 Tx
0110	PORT 12	UART1 Tx
0111	PORT 14	UART2 Tx
1000	PORT 16	UART3 Tx
1001	PORT 18	IrDA Tx
other values	not used	

Table 10-6. PPALLOC Register Bits Decode for a Receive Channel

Ch 1, 3, 5, 7, 9 PPALLOC[3:0]	Port allocated	Peripheral Allocated
0000	PORT 1	I2S1 Rx
0001	PORT 3	I2S2 Rx
0010	PORT 5	AAC1 Rx
0011	PORT 7	AAC2 Rx
0100	PORT 9	AAC3 Rx
0101	PORT 11	I2S3 Rx
0110	PORT 13	UART1 Rx
0111	PORT 15	UART2 Rx
1000	PORT 17	UART3 Rx
1001	PORT 19	IrDA Rx
other values	not used	

Table 10-7. PPALLOC Register Reset Values

M2P Channel	PPALLOC[3:0]	Port allocated on reset
0	0000	PORT 0
1	0000	PORT 1
2	0001	PORT 2
3	0001	PORT 3
4	0010	PORT 4
5	0010	PORT 5
6	0011	PORT 6
7	0011	PORT 7
8	0100	PORT 8
9	0100	PORT 9

Copyright 2007 Cirrus Logic

INTERRUPT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												ChErrorInt	0	NFBInt	STALLInt

Address:

Channel Base Address + 0x0004 - Read/Write

Definition:

This is the interrupt status register. The register is read to obtain interrupt status for enabled interrupts. An interrupt is enabled by writing the corresponding bits in the CONTROL register.

Write this location once to clear the interrupt. (See Interrupt Register Bit Descriptions for the bits where this rule applies.)

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

STALLInt: Indicates channel has stalled. This interrupt is generated on a Channel State machine transition from ON to STALL state, if STALLIntEn set. This is a critical interrupt as it indicates that an over/underflow condition will occur as soon as the peripheral's FIFO is full/empty. The interrupt is cleared by either disabling the channel or writing a new base address which will move the state machine onto the ON state.

NFBInt: Indicates channel requires a new buffer. This interrupt generated on a Channel State machine transition from NEXT to ON state if NFBIntEn set. The interrupt is cleared by either disabling the channel or writing a new base address, which will move the state machine onto the next state.

ChErrorInt: This interrupt is activated when the peripheral attached to the DMA Channel detects an error in the data stream. The peripherals signal this error by ending the current transfer with a **TxEnd/RxEnd** error response. The interrupt is cleared by writing either a "1" or a "0" to this bit.

Copyright 2007 Cirrus Logic

STATUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				BYTES					NextBuffer	Current State		ChError	RSVD	NFB	STALL

Address:

Channel Base Address + 0x000C - Read Only

Definition:

This is the channel status register, which is a read-only register, used to provide status information with respect to the DMA channel.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

Stall: A "1" indicates channel is stalled and cannot currently transfer data because a base address has not been programmed. When the channel is first enabled, the Stall bit is suppressed until the first buffer has been transferred, that is, no stall interrupt generated when STALL state entered from IDLE state, only when entered from ON State. The STALL state can be cleared by writing a base address or disabling the DMA channel. The reason for channel completion can be ascertained by reading the BYTES_REMAINING register, if it is zero, the channel was stopped by the DMA Channel; if it is non-zero, the peripheral ended transfer with **TxEnd/RxEnd**. If the transfer ended with error, ChError bit/interrupt is set.

NFB: A "1" indicates the Channel FSM has moved from NEXT State to ON State. This means that the channel is currently transferring data from a DMA buffer but the next base address for the next buffer in the transfer has not been programmed, and may now be.
0 - Not in ON State, not ready for next buffer update.
1 - In ON State, ready for next buffer BASE/MAXCOUNT updates. NFB interrupt generated if not masked.

ChError: Indicates error status of buffer transfer:
0 - The last buffer transfer completed without error.
1 - The last buffer transfer terminated with an error.

BYTES: This is the number of valid DMA data currently stored by the channel in the DMA Controller in packer or unpacker. Usually used for test/debug.

Copyright 2007 Cirrus Logic

Current State: Indicates the state that the Channel FSM is currently in:
00 - IDLE
01 - STALL
10 - ON
11 - NEXT

NextBuffer: Informs the NFB service routine, after a NFB interrupt, which pair of BASEx/MAXCOUNTx registers is free for update.
0 - Update MAXCNT0/BASE0
1 - Update MAXCNT1/BASE1

The NextBuffer bit gets set to "1" when a write occurs to BASE0 and it gets set to "0" when a write occurs to BASE1. This bit alone cannot be used to determine which of the two buffers is currently being transferred to. For example, if BASE0 is written to, then NextBuffer gets set to "1" and transfers will occur using buffer0. If, during this transfer BASE1 gets written to, then NextBuffer gets set to "0", but the current transfer will continue using buffer0 until it terminates. Then the DMA switches over to using buffer1, at which time the NFB interrupt is generated and software reads the NextBuffer status bit to determine what buffer descriptor is now free for update. In this case it is buffer0.

The NextBuffer status bit can be used in conjunction with the CurrentState status bits to determine the active buffer.

If CurrentState = DMA_ON and NextBuffer = 1 then Buffer0 is the active buffer.

If CurrentState = DMA_ON and NextBuffer = 0 then Buffer1 is the active buffer.

If CurrentState = DMA_NEXT and NextBuffer = 0 then Buffer0 is the active buffer.

If CurrentState = DMA_NEXT and NextBuffer =1 then Buffer1 is the active buffer.

Copyright 2007 Cirrus Logic

REMAIN

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REMAIN															

Address:

Channel Base Address + 0x0014 - Read Only

Definition:

The Channel Bytes Remaining Register contains the number of bytes remaining in the current DMA transfer. Only the lower 16 bits are valid

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

REMAIN: Loaded from the Channel MAXCNT register when the DMA Channel State Machine enters the ON State. Although there are 2 Data transfer states, ON and NEXT, this register need only be assigned in the ON state, because in this state the next buffer to be used is determined (there is only one) and this MAXCNT value is assigned to REMAIN. The DMA State Machine counts down by one byte every time a byte is transferred between the DMA Controller and the Peripheral. When this register reaches zero, the current buffer transfer is complete and the **TxTC/RxTC** are generated and used to indicate this to the peripheral. DMA transfers may also be stopped with the **TxEnd/RxEnd** signals from the peripheral, where the REMAIN register is non-zero at the end of transfer, allowing software to determine the last valid data in a buffer.

Copyright 2007 Cirrus Logic

MAXCNTx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAXCNTx															

Address:

MAXCNT0: Channel Base Address + 0x0020 - Read/Write
MAXCNT1: Channel Base Address + 0x0030 - Read/Write

Definition:

x = "0" or "1". Maximum byte count for the buffer. Represents the double buffer per channel. Only the low order 16 bits are used. Each MAXCNTx register must be programmed before it's corresponding BASEx register.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

MAXCNTx: Maximum byte count for the buffer.

BASEx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASEx															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASEx															

Address:

BASE0: Channel Base Address + 0x0024 - Read/Write
BASE1: Channel Base Address + 0x0034 - Read/Write

Definition:

Base address for the current and next DMA transfer.

Bit Descriptions:

BASEx: x = "0" or "1". Base address for the current and next DMA transfer. Loaded with start address after enabling the DMA Channel, the latter event required to take the Channel State machine into the STALL state, the former event required to enter the ON State.

Copyright 2007 Cirrus Logic

CURRENTx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURRENTx															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURRENTx															

Address:

CURRENT0: Channel Base Address + 0x0028 - Read Only
CURRENT1: Channel Base Address + 0x0038 - Read Only

Definition:

This is the Channel Current Address Register.

Bit Descriptions:

CURRENTx: Returns the current value of the channel address pointer. Upon enabling the DMA Channel and writing the BASE Address Register the contents of this register is loaded into the CURRENTx register and the x buffer becomes active. Following completion of a transfer from a buffer, the post-incremented address is stored in this register so that a software service routine can detect the point in the buffer at which transfer was terminated.

M2M Channel Register Map

The DMA Memory Map defines the mapping for the channel registers for each of the 2 M2M channels that are shown in Table 10-8, the M2M Channel Register Map. This mapping is common for each channel thus offset addresses are shown.

Note that M2M Channel 0 is dedicated to servicing External Device 0, and M2M Channel 1 is dedicated to servicing External Device 1 (when in external DMA transfer mode).

Table 10-8. PPALLOC Register Reset Values

Offset	Name	Access	Bits	Reset Value
Channel Base Address + 0x0000	CONTROL	R/W	32	0
Channel Base Address + 0x0004	INTERRUPT	R/W TC*	3	0
Channel Base Address + 0x0008	Reserved			
Channel Base Address + 0x000C	STATUS	R/W TC*	14	0
Channel Base Address + 0x0010	BCR0	R/W	16	0
Channel Base Address + 0x0014	BCR1	R/W	16	0
Channel Base Address + 0x0018	SAR_BASE0	R/W	32	0
Channel Base Address + 0x001C	SAR_BASE1	R/W	32	0
Channel Base Address + 0x0020	Reserved			
Channel Base Address + 0x0024	SAR_CURRENT0	RO	32	0

Copyright 2007 Cirrus Logic

Table 10-8. PPALLOC Register Reset Values (Continued)

Offset	Name	Access	Bits	Reset Value
Channel Base Address + 0x0028	SAR_CURRENT1	RO	32	0
Channel Base Address + 0x002C	DAR_BASE0	R/W	32	0
Channel Base Address + 0x0030	DAR_BASE1	R/W	32	0
Channel Base Address + 0x0034	DAR_CURRENT0	RO	32	0
Channel Base Address + 0x0038	Reserved			
Channel Base Address + 0x003C	DAR_CURRENT1	RO	32	0

Note:See Table 10-3 for Channel Base Addresses

Note: Write this location once to clear the bit (see Interrupt/Status register description for which bits this rule applies to).*

CONTROL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PWSC							NO_HDSK	RSS		NFBintEn	DREQP		RSVD	DACKP	ETDP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETDP	TM		SAH	DAH	PW		BWC				START	ENABLE	DONEIntEn	SCT	STALLIntEn

Address:

Channel Base Address + 0x0000 - Read/Write

Definition:

This is the Channel Control Register. Used to configure the DMA M2M Channel. All control bits should be programmed before the ENABLE bit is set.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

STALLIntEn: Setting this bit to "1" enables the generation of the STALL interrupt in the STALL State of the DMA Channel State machine. Setting this bit to "0" disables generation of the STALL Interrupt.

SCT: Source Copy Transfer. This bit is used to set up a block transfer from 1 memory source location. If SCT = 1, then one word is read from the source memory location and copied to a block of memory (the number of destination locations written to is determined by BCR). If SCT = 0 then the source address increments as normal after each successful transfer as determined by the transfer size (this is the default setting). In order to use this feature the SAR_BASEx and DAR_BASEx registers must contain word-aligned addresses - the DMA will ignore the 2 LSB's

Copyright 2007 Cirrus Logic

of the source and destination addresses to avoid any problems in the case where software erroneously programs a byte-aligned address. The SCT bit is used only when in M2M software-triggered transfer mode.

DoneIntEn: Setting this bit to "1" enables the generation of the DONE Interrupt which indicates if the transfer completed successfully.

ENABLE: Setting this bit to 1 enables the channel, clearing this bit disables the channel. The channel must always be enabled after writing the Source/Destination Base address registers and the BCR register. When a channel is disabled, the external peripheral signals will be placed in their inactive state.

START: Start Transfer. When this bit is set, the DMA begins M2M transfer in accordance with the values in the control registers. START is cleared automatically after one clock cycle and is always read as a logic 0. This bit, in effect, provides a "Software-triggered DMA capability". A channel must be configured and enabled before setting the START bit. This bit is not used for external DMA transfers, or for IDE and SSP transfers. For a double-buffer software triggered DMA transfer, the START bit need only be set once, that is, at the very beginning of transfer. It is sufficient for software to program the 'other' buffer descriptor only, in order to guarantee rollover to the second buffer when the byte count of the first buffer has been reached.

BWC: Bandwidth Control. These 4 bits are used to indicate the number of bytes in a block transfer. When the BCR register value is within 15 bytes of a multiple of the BWC value, the DMA releases the bus by negating the AHB bus request strobe allowing lower priority masters to be granted control of the bus. BWC = 0000 specifies the maximum transfer rate: other values specify a transfer rate limit.

The BWC bits should only be set for software triggered M2M transfers, where **HREQ** stays asserted throughout the transfer. For transfer to/from external devices, **HREQ** is released after every transfer, and so bandwidth control is not needed.

The BWC bits are ignored when in external DMA transfer mode.

Copyright 2007 Cirrus Logic

Example: if BWC = 1010b (indicating 1024 bytes, see Table 10-9, below), the DMA relinquishes control of the bus on completion of the current burst transfer after BCR values which are within 15 bytes of multiples of 1024.

Table 10-9. BWC Decode Values

BWC	Bytes
0000	Full DMA transfer completes
0001	16
0010	16
0011	16
0100	16
0101	32
0110	64
0111	128
1000	256
1001	512
1010	1024
1011	2048
1100	4096
1101	8192
1110	16384
1111	32768

PW: Peripheral Width. For external DMA transfers, these bits are used to program the DMA to request byte/half-word/word wide AHB transfers, depending on the width of the external device. These bits are not used for software triggered M2M transfers.

00 - Byte (8 bits)
01 - Half-word (16 bits)
10 - Word (32 bits)
11 - Not used

For word accesses the lower 2 bits of the source/destination address are ignored.

For half-word accesses the lower bit of the source/destination address is ignored.

Copyright 2007 Cirrus Logic

DAH: Destination Address Hold - This bit is used for external M2P transfers where the external memory destination is a memory-mapped FIFO-based device (with one address location) or for internal peripheral transfers (M2P) to the peripheral's FIFO buffer.
1 - Hold the destination address throughout the transfer (do not increment).
0 - Increment the destination address after each transfer in the transaction.

SAH: Source Address Hold - This bit is used for external DMA transfers where the external memory source is a memory-mapped FIFO-based device (with one address location) or for internal register locations.
1 - Hold the source address throughout the transfer (do not increment).
0 - Increment the source address after each transfer in the transaction.

TM: Transfer Mode:
00 - Software initiated DMA transfer.
01 - Hardware initiated external DMA transfer, that is, transfer from memory to external device or to IDE or SSP.
10 - Hardware initiated external DMA transfer, that is, transfer from external device (or IDE/SSP) to memory.
11 - Not used.

ETDP: End-of-Transfer/Terminal Count pin Direction & Polarity:
00 - The **DEOT/TC** pin is programmed as an active low end-of-transfer input.
01 - The **DEOT/TC** pin is programmed as an active high end-of-transfer input.
10 - The **DEOT/TC** pin is programmed as an active low terminal count output.
11 - The **DEOT/TC** pin is programmed as an active high terminal count output.

DACKP: DMA Acknowledge pin Polarity:
0 - **DACK** is active low.
1 - **DACK** is active high.

DREQP: DMA Request pin Polarity. These bits must be set before the channels ENABLE bit is set. Otherwise the reset value, "00", will cause the DMA to look for an active low, level sensitive **DREQ**.
00 - **DREQ** is active low, level sensitive.
01 - **DREQ** is active high, level sensitive.
10 - **DREQ** is active low, edge sensitive.
11 - **DREQ** is active high, edge sensitive.

Copyright 2007 Cirrus Logic

NFBIntEn: Setting this bit to "1" enables the generation of the NFB interrupt in the DMA_BUF_ON state of the DMA channel buffer state machine. Setting this bit to zero disables generation of the NFB Interrupt. Normally when the channel is enabled, this bit should be 1. However in the case where the current buffer is the last, then this bit can be cleared to prevent the generation of an interrupt while the DMA State machine is in the DMA_BUF_ON state.

RSS: Request Source Selection.
00 - External DReq.
01 - Internal SSPRx.
10 - Internal SSPTx.
11 - Internal IDE.

NO_HDSK: When set, the peripheral doesn't require the regular handshake protocol. This is optional for external DMAs, but this bit needs to be set for SSP and IDE operations. Setting this bit will imply the use of a wait state counter that will mask hardware requests after each DMA write.

PWSC: Peripheral Wait States Count. Gives the latency (in HCLK cycles) needed by the peripheral to de-assert its request line once the M2M transfer is finished.During this latency period, the DMA channel will not consider any request. This wait state count is triggered after each peripheral width transfer, right after the DMA write phase.In the case of internal DMA, this means that the count will start when the DMA has had confirmation from AHB that the write is accepted and done. In the case of an external DMA that doesn't use a handshaking protocol, the count will start when the DMA has received the acknowledge of the write from the SMC.If the acknowledge from the SMC takes too long to arrive, the processor can still cancel the counter stall by writing the CONTROL register.

INTERRUPT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD													NFBint	DONEInt	STALLInt

Address:

Channel Base Address + 0x0004 - Read/Write

Copyright 2007 Cirrus Logic

Definition:

This is the interrupt status register. The register is read to obtain interrupt status for enabled interrupts. An interrupt is enabled by writing the corresponding bits in the CONTROL register.

Write this location once to clear the interrupt. (See the Interrupt Register Bit Descriptions for the bits where this applies.)

10

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
STALLInt:	Indicates channel has stalled. This interrupt is generated on a Channel State machine transition from MEM_RD (memory read) or MEM_WR (memory write) to the STALL state, assuming STALLIntEn set. The interrupt is cleared by either disabling the channel or by triggering a new transfer.
DONEInt:	Transaction is done. When enabled, this interrupt is set when all DMA controller transactions complete normally, as determined by the transfer count/external peripheral **DEOT** signal. When a transfer completes, software must clear the DONE bit before reprogramming the DMA, by writing either a "0" or "1" to this bit. This must be done even if the DMA interrupt is disabled. The DMA will ignore any additional DREQs that it receives from the external peripheral (if operating in external DMA mode) until the software clears the DONE interrupt and reprograms the DMA with new BCRx values.
NFBInt:	Indicates that a channels buffer descriptor is free for update. This interrupt is generated if NFBIntEn is set, when a transfer begins using the second buffer of the double-buffer set, thus informing software that it can now set up the other buffer. The interrupt is cleared by either disabling the channel or writing a new BCR value to set up a new buffer descriptor. The interrupt is not generated for a single-buffer transfer. In software triggered M2M mode, servicing of the NFB interrupt is dependent on the system level AHB arbitration since the DMA's **HREQ** (AHB request) may be continuously held high.

Copyright 2007 Cirrus Logic

STATUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD		DREQS	NB	NFB	EOTS		TCS		DONE	CurrentState					STALL

Address:

Channel Base Address + 0x000C - Read/Write

Definition:

This is the channel status register, used to provide status information with respect to the DMA channel. All register bits are read-only except for the DREQS status bit which can be cleared by a write (either a "0" or a "1") to this register.

Write this location once to clear the interrupt (see Interrupt Register Bit Descriptions for which bits this rule applies to).

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

Stall: A "1" indicates channel is stalled and cannot currently transfer data because the START bit has not been programmed or an external device has not asserted **DREQ**. When the channel is first enabled, the Stall bit is suppressed until the first buffer has been transferred, that is, no stall interrupt generated when STALL state entered from IDLE state, only when entered from MEM_WR State. The STALL state can be cleared by:

- Setting the START bit
- An external peripheral requesting service (depending on transfer mode)
- Disabling the DMA channel
- A request from SSP or IDE

Copyright 2007 Cirrus Logic

CurrentState: Indicates the states that the M2M Channel Control FSM and M2M Buffer FSM are currently in:

CurrentState[2:0] - These indicate the state of M2M Channel Control FSM:
000 - DMA_IDLE
001 - DMA_STALL
010 - DMA_MEM_RD
011 - DMA_MEM_WR
100 - DMA_BWC_WAIT

CurrentState[4:3] - These indicate the state of M2M Buffer FSM:
00 - DMA_NO_BUF
01 - DMA_BUF_ON
10 - DMA_BUF_NEXT

DONE: Transfer completed successfully. The transfer is terminated on the occurrence of DEOT being asserted by the peripheral or the byte count expiring, whichever happens sooner. When a transfer completes, software must clear the Interrupt.DONEInt bit before reprogramming the DMA, by writing either "0" or "1" to this bit. The DMA will ignore any more **DREQ**s that it receives from the external device (if operating in external peripheral mode) until such time that software clears the DONE interrupt and reprograms the DMA with new BCRx values, and this even if the DMA interrupt is disabled.

TCS: Terminal Count status. This status bit reflects whether or not the actual byte count has reached the programmed limit for buffer descriptor "0" or "1" respectively:
00 - Terminal Count has not been reached for either buffer descriptor 1 or 0.
01 - Terminal Count has not been reached for buffer 1 and has been reached for buffer descriptor 0.
10 - Terminal Count has been reached for buffer 1 and has not been reached for buffer descriptor 0.
11 - Terminal Count has been reached for both buffer descriptors.

The TCS status bit for a buffer descriptor is cleared when the BCR register of that buffer descriptor has been programmed with a new value.

Copyright 2007 Cirrus Logic

EOTS: End-Of-Transfer status (valid only if the **DEOT/TC** pin has been programmed for the DEOT function, that is, the control reg bit ETDP[1] = 0) for buffer descriptor 1 or 0 respectively.

00 - End of transfer has not been requested by external device for either buffer descriptor.
01 - End of transfer has been requested by external device for buffer descriptor 0 only.
10 - End of transfer has been requested by external device for buffer descriptor 1 only.
11 - End of transfer has been requested by external peripheral for both buffer descriptors.

NFB: A "1" indicates that the channel is currently transferring data from a DMA buffer but the next byte count register for the next buffer in the transfer has not been programmed, and may now be programmed. This interrupt is generated when the DMA buffer state machine moves from the DMA_BUF_NEXT state to the DMA_BUF_ON state, that is, when transfer begins using the second buffer of the double buffer pair. Thus for a double-buffer transfer both BCR registers must be programmed once before the NFB status bit can be used to determine when the next BCR register should be programmed.
0 - Not ready for next buffer update.
1 - Ready for next buffer updates. NFB interrupt generated if not masked.

NB: NextBuffer status bit - Informs the NFB service routine, after a NFB interrupt, which pair of SAR_BASEx/DAR_BASEx/BCRx registers is free for update.
0 - Update SAR_BASE0/DAR_BASE0/BCR0
1 - Update SAR_BASE1/DAR_BASE1/BCR1

The NextBuffer bit gets set to "1" when a write occurs to BCR0 and it gets set to "0" when a write occurs to BCR1. This bit alone cannot be used to determine which of the two buffers is currently being transferred to - for example if BCR0 is written, then NextBuffer gets set to "1" and transfers will occur using buffer0. If, during this transfer BCR1 gets written, then NextBuffer gets set to "0", but the current transfer will continue using buffer0 until it terminates. Then the DMA switches over to using buffer1 at which time the NFB interrupt is generated and software reads the NextBuffer status bit to determine what buffer descriptor is now free for update - in this case it is buffer0.

Copyright 2007 Cirrus Logic

The NextBuffer status bit can be used in conjunction with the CurrentState status bits to determine the active buffer according to the following rules:

If CurrentState[4:3] = DMA_BUF_ON and NextBuffer = 1 then Buffer0 is the active buffer.

If CurrentState[4:3] = DMA_BUF_ON and NextBuffer = 0 then Buffer1 is the active buffer.

If CurrentState[4:3] = DMA_BUF_NEXT and NextBuffer = 0 then Buffer0 is the active buffer.

If CurrentState[4:3] = DMA_BUF_NEXT and NextBuffer =1 then Buffer1 is the active buffer.

DREQS: **DREQ** Status - This bit reflects the status of the synchronized external DMA Request signal or IDE/SSP requests:
0 - No external DMA request is pending or, in the case of a transfer without handshaking, the request is not validated yet, the wait state counter is running.
1 - An external DMA request or a validated IDE/SSP or external peripheral without handshaking request is pending.

DREQS can be polled by software at any time. It can, for example, be used to determine whether or not the DMA needs to be set up for a transfer when the DMA is in the STALL state and is receiving DREQs, but the BCRx registers have not been programmed. It is important to notice that, in the case of a transfer without handshaking (external DMA or IDE or SSP), DREQS might be clear if a request is pending but is not validated as a result of a wait state counter still running.

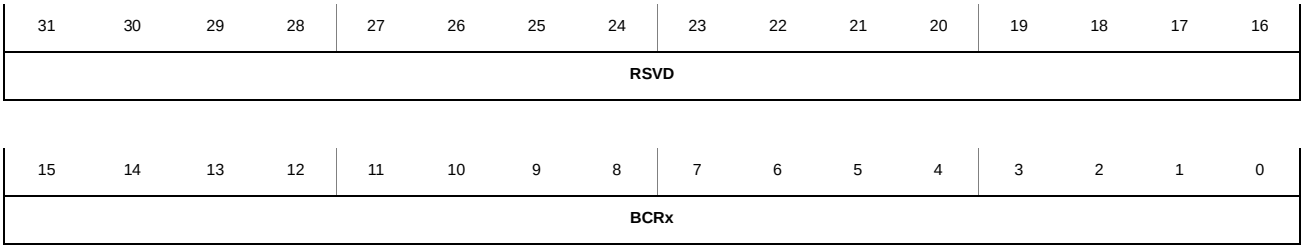
When the channel STATUS register is written with any 32-bit value, this will cause the DREQS bit of the STATUS register to be cleared. A write to the STATUS register only affects the DREQS bit. If an edge is detected on **DREQ** when no previous request is still pending in the DMA (that is, DREQS clear), then the DREQS bit is set by the DMA

Copyright 2007 Cirrus Logic

to indicate that the external device has requested service. The STATUS register is written by software to clear the DREQS status bit, thus causing the DMA to ignore the request.

For level-sensitive **DREQ** mode, do not attempt to clear the DREQS status bit, as the request will keep coming from the external device. The hardware ensures that a write to the STATUS register has no effect when in level-sensitive mode.

BCRx



Address:

BCR0: Channel Base Address + 0x0010 - Read/Write
BCR1: Channel Base Address + 0x0014 - Read/Write

Definition:

The Channel Bytes Count Register contains the number of bytes yet to be transferred for a given block of data in a M2M transfer. Only the lower 16 bits are valid.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

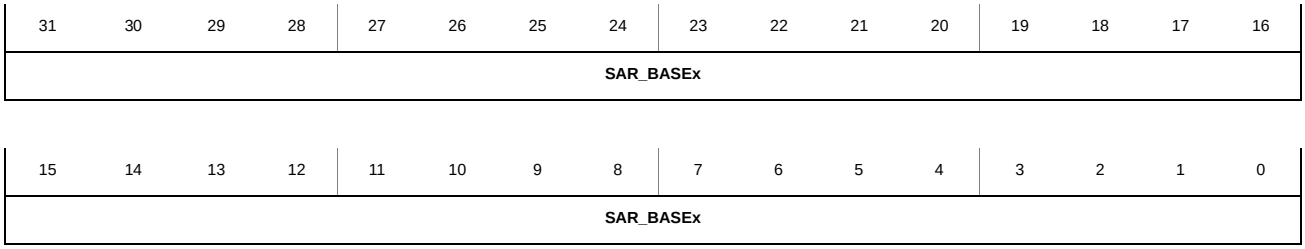
BCRx: x = "0" or "1" representing the double buffer per channel. The BCR register must be loaded with the number of byte transfers to occur. It decrements on the successful completion of the address transfer during the write-to-memory state of the M2M transfer. At least 1 of the BCRx registers must be programmed to a non-zero value before the ENABLE bit and the START bit (in the case of software-trigger M2M mode) are set in the Control register. Writing to a BCRx register causes a next buffer update, that is, only the BCR of the buffer descriptor has to be written to in order to use that buffer since the SAR_BASEx and DAR_BASEx registers do not have to be continuously updated.

Copyright 2007 Cirrus Logic

For a double/multiple buffer transfer, the second buffer descriptor can be programmed while the transfer using the first buffer is being carried out (thus reducing software latency impact). The NFB interrupt is generated when transfer begins using the second buffer. The NFB interrupt service routine can then be used to update the free buffer descriptor (in the case where a third buffer is required).

If BCRx = 0 when the transfer is triggered, then NO transfers will occur, that is, the DMA will stay in the STALL state.

SAR_BASEx



Address:

SAR_BASE0: Channel Base Address + 0x0018 - Read/Write
SAR_BASE1: Channel Base Address + 0x001C - Read/Write

Definition:

This register contains the base memory address from which the DMA controller requests data.

Bit Descriptions:

SAR_BASEx: x = "0" or "1" representing the double buffer per channel. This register contains the base memory address from which the DMA controller requests data. At least 1 of the SAR_BASEx registers must be programmed before the ENABLE bit and the START bit (in the case of software-trigger M2M mode) are set in the Control register, and also before the corresponding BCRx register is programmed. The second buffer descriptor can be programmed while the transfer using the "other" buffer is being carried out (thus reducing software latency impact). When transferring from external device to memory, the SAR_BASEx will contain the base address of the memory mapped device.

Copyright 2007 Cirrus Logic

DAR_BASEx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DAR_BASEx															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAR_BASEx															

Address:

DAR_BASE0: Channel Base Address + 0x002C- Read/Write
DAR_BASE1: Channel Base Address + 0x0030 - Read/Write

Definition:

This register contains the base memory address to which the DMA controller transfers data.

Bit Descriptions:

DAR_BASEx: x = 0 or 1 representing the double buffer per channel. This register contains the base memory address to which the DMA controller sends data. At least 1 of the DAR_BASEx registers must be programmed before the ENABLE bit and the START bit (in the case of software trigger M2M mode) are set in the Control register, and also before the corresponding BCRx register is programmed. The second buffer descriptor can be programmed while the transfer using the 'other' buffer is being carried out (thus reducing software latency impact). When transferring from memory to external peripheral, the DAR_BASEx will contain the base address of the memory mapped device.

SAR_CURRENTx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SAR_CURRENTx															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAR_CURRENTx															

Address:

SAR_CURRENT0: Channel Base Address + 0x0024 - Read Only
SAR_CURRENT1: Channel Base Address + 0x0028 - Read Only

Definition:

This is the Channel Current Source Address Register.

Copyright 2007 Cirrus Logic

Bit Descriptions:

SAR_CURRENTx: Returns the current value of the channel source address pointer. Upon writing the BCRx register, the contents of the SAR_BASEx register is loaded into the SAR_CURRENTx register and the x buffer becomes active. Following completion of a transfer from a buffer, the post-incremented address is stored in this register so that a software service routine can detect the point in the buffer at which transfer was terminated.

DAR_CURRENTx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DAR_CURRENTx															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAR_CURRENTx															

Address:

DAR_CURRENT0: Channel Base Address + 0x0044 - Read Only
DAR_CURRENT1: Channel Base Address + 0x003C - Read Only

Definition:

This is the Channel Current Destination Address Register.

Bit Descriptions:

DAR_CURRENTx: Returns the current value of the channel destination address pointer. Upon writing the BCRx register the contents of the DAR_BASEx register is loaded into the DAR_CURRENTx register and the x buffer becomes active. Following completion of a transfer from a buffer, the post-incremented address is stored in this register so that a software service routine can detect the point in the buffer at which transfer was terminated.

DMAGlInt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Copyright 2007 Cirrus Logic

Address:

0x8000_03C0 - Read/Write

Definition:

DMA Global Interrupt Register. This register indicates which channels have an active interrupt. It is a read only register.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

D0 - D1: These interrupts are per channel interrupts, as shown in Table 10-10. Each bit is a logical OR of the INTERRUPT register per channel. There are no dedicated storage of these channel interrupts. Once each Channel's Interrupts' are clear, the associated channel interrupt is clear.

Note: The order of the internal M2P channel interrupts is for compatibility reasons with previous versions of software.

Table 10-10. DMA Global Interrupt (DMAGlInt) Register

Bit No.	Description
D[31:12]	RSVD
D11	M2M Channel 1 Interrupt
D10	M2M Channel 0 Interrupt
D9	M2P Channel 8 Interrupt
D8	M2P Channel 9 Interrupt
D7	M2P Channel 6 Interrupt
D6	M2P Channel 7 Interrupt
D5	M2P Channel 4 Interrupt
D4	M2P Channel 5 Interrupt
D3	M2P Channel 2 Interrupt
D2	M2P Channel 3 Interrupt
D1	M2P Channel 0 Interrupt
D0	M2P Channel 1 Interrupt

DMAChArb

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															CHARB

Address:

0x8000_0380 - Read/Write

Copyright 2007 Cirrus Logic

Definition:

DMA Channel Arbitration Register. This bit controls the DMA channel arbitration.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

CHARB: This bit controls DMA channel arbitration. It is reset to "0", thus giving a default setting of internal Memory-to-Peripheral channels having a higher priority than Memory-to-Memory channels. This bit can be set to "1" to reverse the default order, that is, giving M2M transfers a higher priority than internal M2P.

Copyright 2007 Cirrus Logic

Chapter 11
Universal Serial Bus Host Controller

11

11.1 Introduction

Note: The EP9301 and EP9302 processors each have 2 USB 2.0 Host ports.

Note: The EP9307, EP9312, and EP9315 processors each have 3 USB 2.0 Host ports.

The Universal Serial Bus (USB) Host Controller enables communication to USB 2.0 low-speed (1.2 Mbps) and full-speed (12 Mbps) devices. The controller supports three root hub ports and complies with the Open Host Controller Interface (OpenHCI) specification, version 1.0a. (For additional information, see Section P.3 in Chapter P, "Preface".)

11.1.1 Features

The features of the USB Host Controller are:

- Open Host Controller Interface Specification (OpenHCI) Rev 1.0 compliant.
- Universal Serial Bus Specification Rev. 2.0 compliant.
- Support for both low speed and full speed USB devices.
- Root Hub has three downstream ports
- Master and Slave AHB interfaces
- DMA functionality

The USB Host Controller is partitioned into the key sub blocks as indicated in Figure 11-6.

11.2 Overview

Figure 11-1 shows four main focus areas of a USB system. These areas are:

- Client Software/USB Driver
- Host Controller Driver (HCD)
- Host Controller (HC)
- USB Device.

Copyright 2007 Cirrus Logic

The Client Software/USB Device and Host Controller Driver are implemented in software. The Host Controller and USB Device are implemented in hardware. OpenHCI specifies the interface between the Host Controller Driver and the Host Controller and describes the fundamental operation of each.

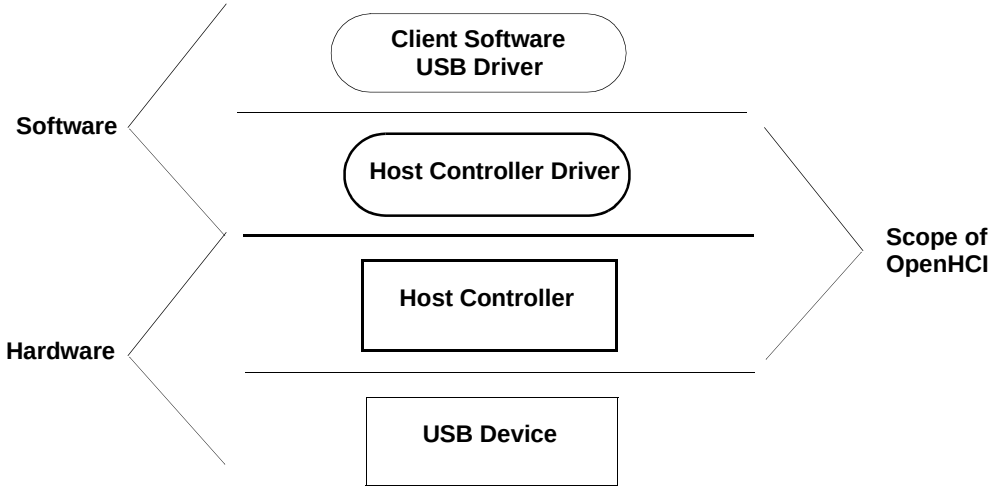


Figure 11-1. USB Focus Areas

The Host Controller Driver and Host Controller work in tandem to transfer data between client software and a USB device. Data is translated from shared-memory data structures at the client software end to USB signal protocols at the USB device end, and vice-versa.

11.2.1 Data Transfer Types

There are four data transfer types defined in USB. Each type is optimized to match the service requirements between the client software and the USB device. The four types are:

- Interrupt Transfers - Small data transfers used to communicate information from the USB device to the client software. The Host Controller Driver polls the USB device by issuing tokens to the device at a periodic interval sufficient for the requirements of the device.
- Isochronous Transfers - Periodic data transfers with a constant data rate. Data transfers are correlated in time between the sender and receiver.
- Control Transfers - Nonperiodic data transfers used to communicate configuration/command/status type information between client software and the USB device.
- Bulk Transfers - Nonperiodic data transfers used to communicate large amounts of information between client software and the USB device.

In OpenHCI the data transfer types are classified into two categories: periodic and nonperiodic. Periodic transfers are interrupt and isochronous since they are scheduled to run at periodic intervals. Nonperiodic transfers are control and bulk since they are not scheduled to run at any specific time, but rather on a time-available basis.

Copyright 2007 Cirrus Logic

11.2.2 Host Controller Interface

11.2.2.1 Communication Channels

There are two communication channels between the Host Controller and the Host Controller Driver. The first channel uses a set of operational registers located on the HC. The Host Controller is the target for all communication on this channel. The operational registers contain control, status, and list pointer registers. Within the operational register set is a pointer to a location in shared memory named the Host Controller Communications Area (HCCA). The HCCA is the second communication channel. The Host Controller is the master for all communication on this channel. The HCCA contains the head pointers to the interrupt Endpoint Descriptor lists, the head pointer to the done queue, and status information associated with start-of-frame processing. Figure 11-2 shows the communication channels.

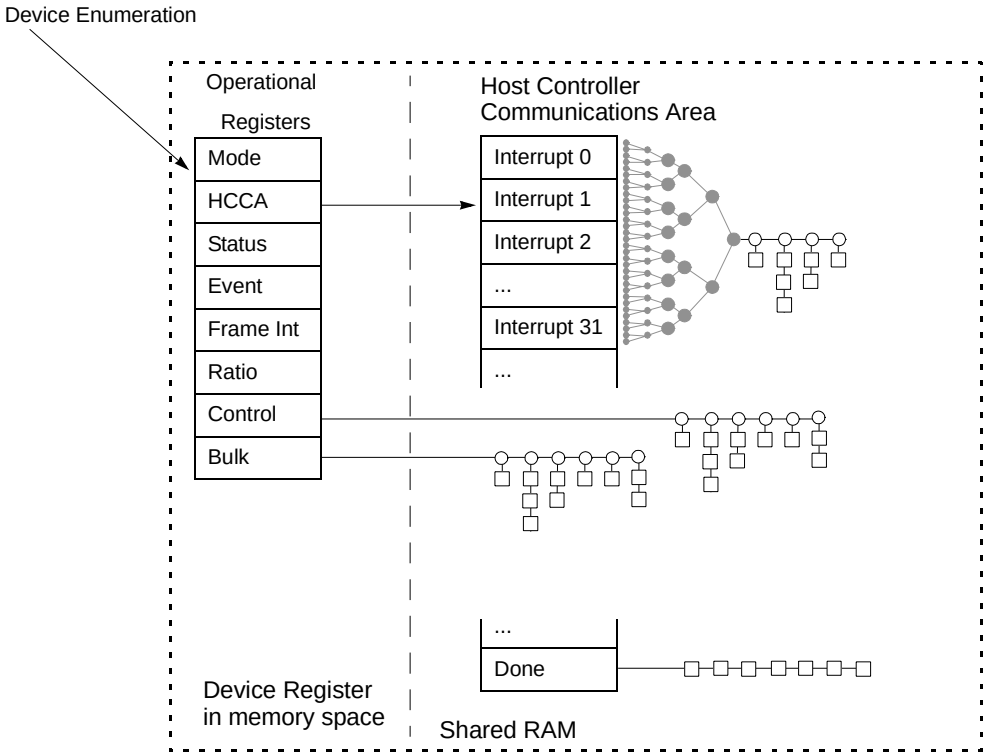


Figure 11-2. Communication Channels

Copyright 2007 Cirrus Logic

11.2.2.2 Data Structures

The basic building blocks for communication across the interface are the Endpoint Descriptor (ED) and Transfer Descriptor (TD).

The Host Controller Driver assigns an Endpoint Descriptor to each endpoint in the system. The Endpoint Descriptor contains the information necessary for the Host Controller to communicate with the endpoint. The fields include the maximum packet size, the endpoint address, the speed of the endpoint, and the direction of data flow. Endpoint Descriptors are linked in a list.

A queue of Transfer Descriptors is linked to the Endpoint Descriptor for the specific endpoint. The Transfer Descriptor contains the information necessary to describe the data packets to be transferred. The fields include data toggle information, shared memory buffer location, and completion status codes. Each Transfer Descriptor contains information that describes one or more data packets. The data buffer for each Transfer Descriptor ranges in size from 0 to 8192 bytes with a maximum of one physical page crossing. Transfer Descriptors are linked in a queue: the first one queued is the first one processed.

Each data transfer type has its own linked list of Endpoint Descriptors to be processed. Figure 11-3, Typical List Structure, is a representation of the data structure relationships.

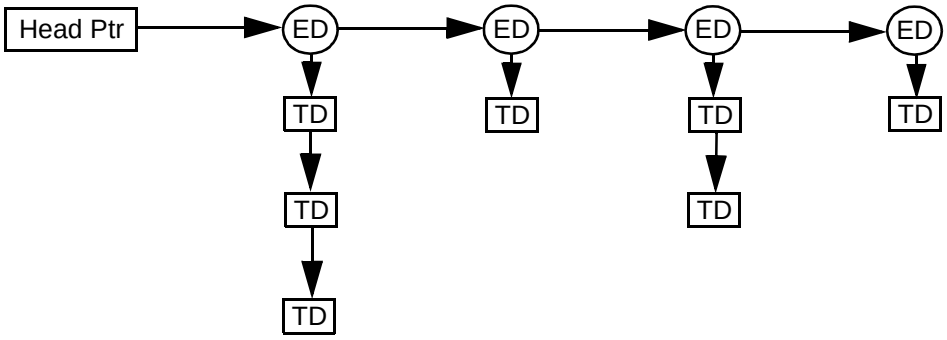


Figure 11-3. Typical List Structure

The head pointers to the bulk and control Endpoint Descriptor lists are maintained within the operational registers in the HC. The Host Controller Driver initializes these pointers prior to the Host Controller gaining access to them. Should these pointers need to be updated, the Host Controller Driver may need to halt the Host Controller from processing the specific list, update the pointer, then re-enable the HC.

The head pointers to the interrupt Endpoint Descriptor lists are maintained within the HCCA. There is no separate head pointer for isochronous transfers. The first isochronous Endpoint Descriptor simply links to the last interrupt Endpoint Descriptor. There are 32 interrupt head pointers. The head pointer used for a particular frame is determined by using the last 5 bits of the Frame Counter as an offset into the interrupt array within the HCCA.

The interrupt Endpoint Descriptors are organized into a tree structure with the head pointers being the leaf nodes. The desired polling rate of an Interrupt Endpoint is achieved by

Copyright 2007 Cirrus Logic

scheduling the Endpoint Descriptor at the appropriate depth in the tree. The higher the polling rate, the closer to the root of the tree the Endpoint Descriptor will be placed since multiple lists will converge on it. Figure 11-4 illustrates the structure for Interrupt Endpoints. The Interrupt Endpoint Descriptor Placeholder indicates where zero or more Endpoint Descriptors may be enqueued. The numbers on the left are the index into the HCCA interrupt head pointer array.

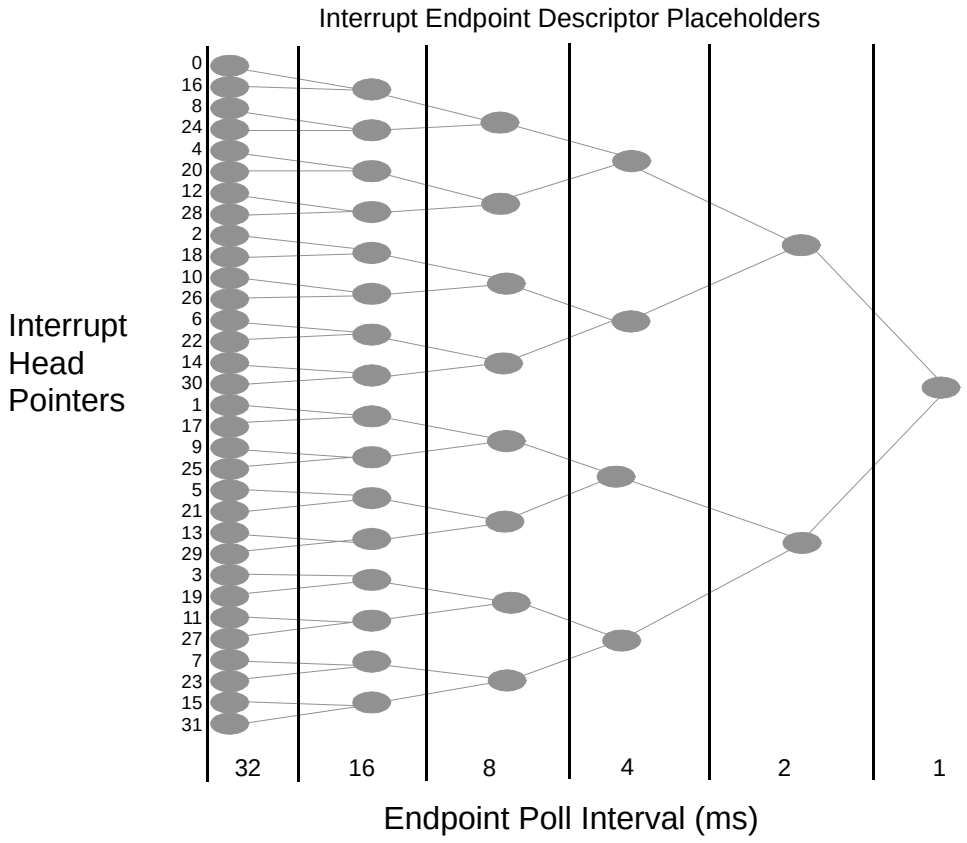


Figure 11-4. Interrupt Endpoint Descriptor Structure

Figure 11-5 is a sample Interrupt Endpoint schedule. The schedule shows one Endpoint Descriptors at a 1 ms poll interval, two Endpoint Descriptors at a 2 ms poll interval, one Endpoint at a 4 ms poll interval, two Endpoint Descriptors at an 8 ms poll interval, two Endpoint Descriptors at a 16 ms poll interval, and two Endpoint Descriptors at a 32 ms poll interval. Note that in this example unused Interrupt Endpoint Placeholders are bypassed and the link is connected to the next available Endpoint in the hierarchy.

Copyright 2007 Cirrus Logic

11

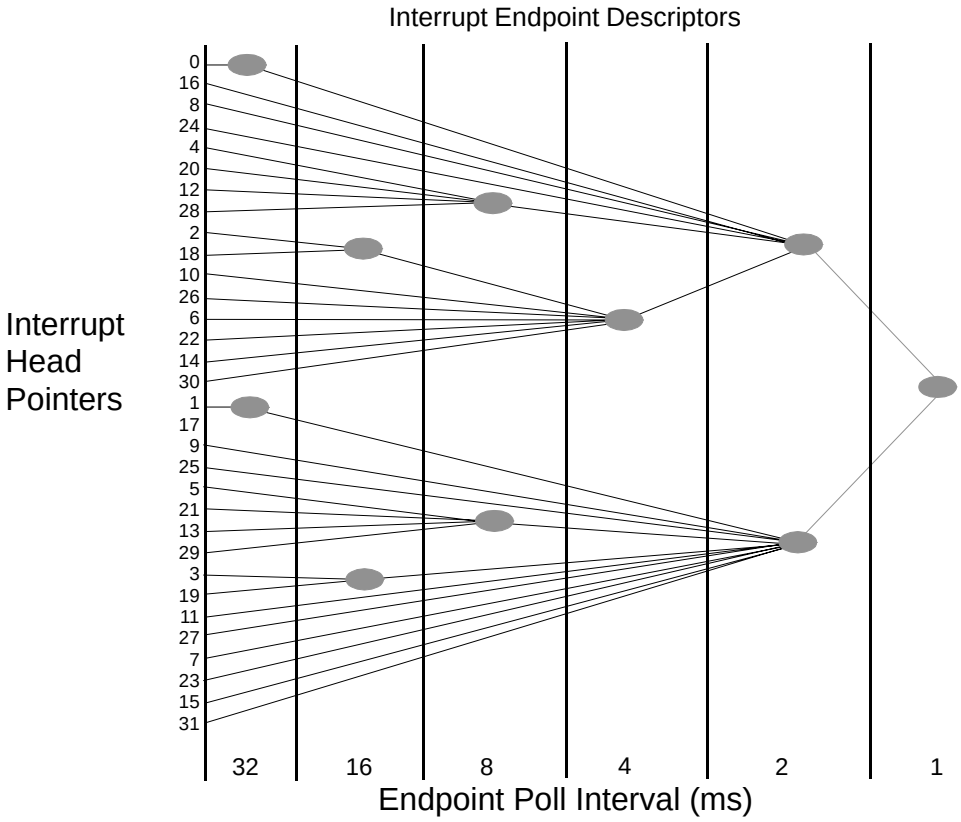


Figure 11-5. Sample Interrupt Endpoint Schedule

11.2.3 Host Controller Driver Responsibilities

This section summarizes the Host Controller Driver (HCD) responsibilities.

11.2.3.1 Host Controller Management

The Host Controller Driver manages the operation of the Host Controller (HC). It does so by communicating directly to the operational registers in the Host Controller and establishing the interrupt Endpoint Descriptor list head pointers in the HCCA.

The Host Controller Driver maintains the state of the HC, list processing pointers, list processing enables, and interrupt enables.

11.2.3.2 Bandwidth Allocation

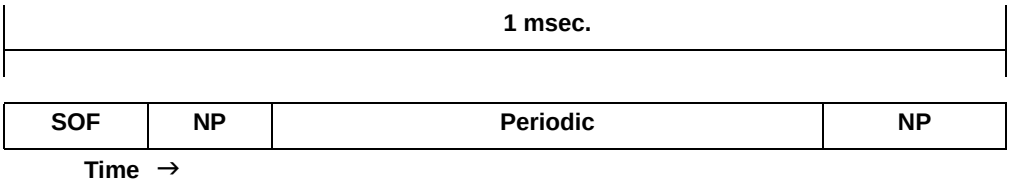
All access to the USB is scheduled by the Host Controller Driver. The Host Controller Driver allocates a portion of the available bandwidth to each periodic endpoint. If sufficient bandwidth is not available, a newly-connected periodic endpoint will be denied access to the bus.

Copyright 2007 Cirrus Logic

A portion of the bandwidth is reserved for nonperiodic transfers. This ensures that some amount of bulk and control transfers will occur in each frame period. The frame period is defined for USB to be 1.0 ms.

The bandwidth allocation policy for OpenHCI is shown in Table 11-1. Each frame begins with the Host Controller sending the Start of Frame (SOF) synchronization packet to the USB bus. This is followed by the Host Controller servicing nonperiodic transfers until the frame interval counter reaches the value set by the Host Controller Driver, indicating that the Host Controller should begin servicing periodic transfers. After the periodic transfers complete, any remaining time in the frame is consumed by servicing nonperiodic transfers once more.

Table 11-1. Frame Bandwidth Allocation



11.2.3.3 List Management

The transport mechanism for USB data packets is via Transfer Descriptor queues linked to Endpoint Descriptor lists. The Host Controller Driver creates these data structures then passes control to the Host Controller for processing.

The Host Controller Driver is responsible for enqueuing and dequeuing Endpoint Descriptors. Enqueuing is done by adding the Endpoint Descriptor to the tail of the appropriate list. This may occur simultaneously with the Host Controller processing the list without requiring any lock mechanism. Before dequeuing an Endpoint Descriptor, the Host Controller Driver may disable the Host Controller from processing the entire Endpoint Descriptor list of the data type being removed to ensure that the Host Controller is not accessing the Endpoint Descriptor.

The Host Controller Driver is also responsible for enqueuing Transfer Descriptors to the appropriate Endpoint Descriptor. Enqueuing is done by adding the Transfer Descriptor to the tail of the appropriate queue. This may occur simultaneously to the Host Controller processing the queue without requiring any lock mechanism. Under normal operation, the Host Controller dequeues the Transfer Descriptor. However, the Host Controller Driver dequeues the Transfer Descriptor when the Transfer Descriptor is being canceled due to a request from the client software or certain error conditions. In this instance, the Endpoint Descriptor is disabled prior to the Transfer Descriptor being dequeued.

11.2.3.4 Root Hub

The Root Hub is integrated into the HC. The internal registers of the Root Hub are exposed to the Host Controller Driver which is responsible for providing the proper hub-class protocol with the USB Driver and proper control of the Root Hub.

Copyright 2007 Cirrus Logic

11.2.4 Host Controller Responsibilities

This section summarizes the Host Controller (HC) responsibilities.

11.2.4.1 USB States

There are four USB states defined in OpenHCI: *UsbOperational, UsbReset, UsbSuspend,* and *UsbResume*. The Host Controller puts the USB bus in the proper operating mode for each state.

11.2.4.2 Frame Management

The Host Controller keeps track of the current frame counter and the frame period. At the beginning of each frame, the Host Controller generates the Start of Frame (SOF) packet on the USB bus and updates the frame count value in system memory. The Host Controller also determines if enough time remains in the frame to send the next data packet.

11.2.4.3 List Processing

The Host Controller operates on the Endpoint Descriptors and Transfer Descriptors enqueued by the Host Controller Driver.

For interrupt and isochronous transfers, the Host Controller begins at the Interrupt Endpoint Descriptor head pointer for the current frame. The list is traversed sequentially until one packet transfer from the first Transfer Descriptor of all interrupt and isochronous Endpoint Descriptors scheduled in the current frame is attempted.

For bulk and control transfers, the Host Controller begins in the respective list where it last left off. When the Host Controller reaches the end of a list, it loads the value from the head pointer and continues processing. The Host Controller processes *n* control transfers to 1 bulk transfer where the value of *n* is set by the Host Controller Driver.

When a Transfer Descriptor completes, either successfully or due to an error condition, the Host Controller moves it to the Done Queue. Enqueuing on the Done Queue occurs by placing the most recently completed Transfer Descriptor at the head of the queue. The Done Queue is transferred periodically from the Host Controller to the Host Controller Driver via the HCCA.

Copyright 2007 Cirrus Logic

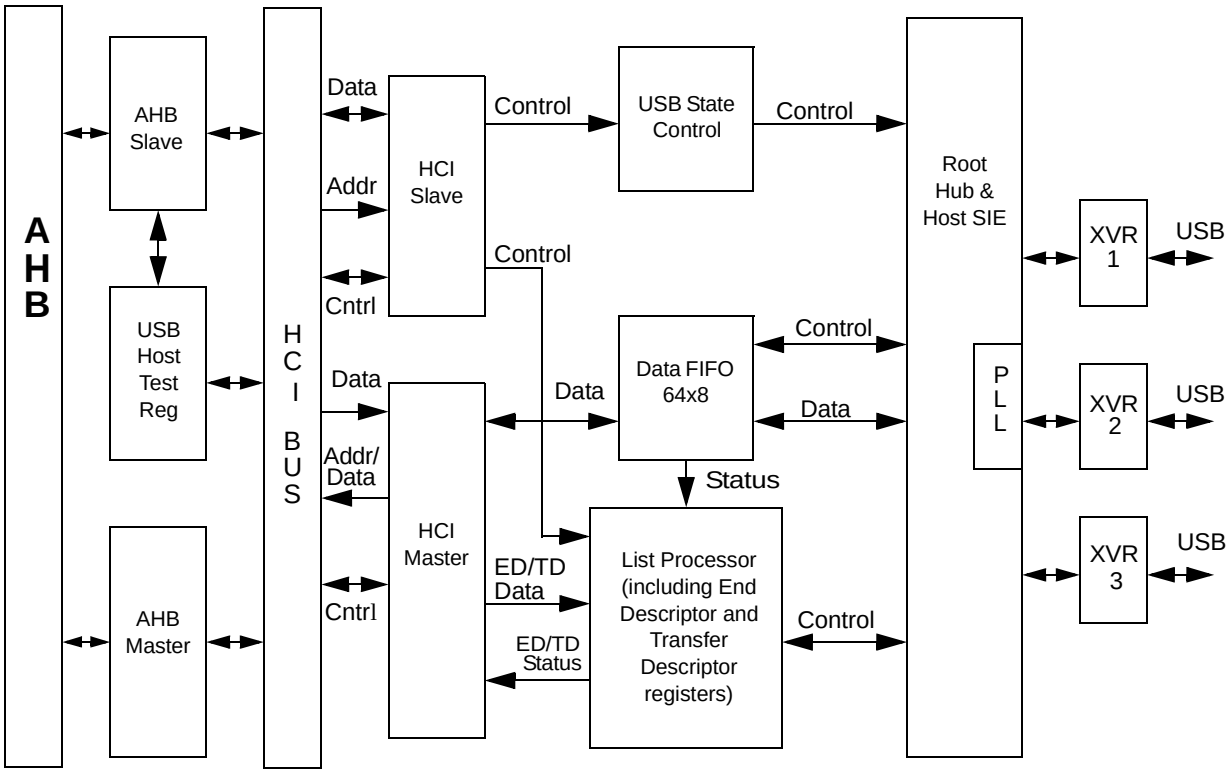


Figure 11-6. USB Host Controller Block Diagram

11.2.5 USB Host Controller Blocks

11.2.5.1 AHB Slave

This block allows access to the OHCI operational registers from/to the AHB via the HCI Bus.

11.2.5.2 AHB Master

This block enables the USB Host Controller to be an AHB Master peripheral and interfaces with the HCI Master block via the HCI Bus.

The AHB Master includes a Data FIFO which will use a 44x37 bit Data FIFO. 32-bit data, 4-bit HCI_MBeN[3:0] (byte lane enables) and HCI_MWBstOnN (burst on) make up the width of the Data FIFO.

11.2.5.3 HCI Slave Block

This block contains the OHCI operational registers, which are programmed by the Host Controller Driver (HCD).

Copyright 2007 Cirrus Logic

11.2.5.4 HCI Master Block

The HCI Master Block handles read/write requests to system memory that are initiated by the List Processor while the Host Controller (HC) is in the operational state and is processing the lists queued in by HCD. It generates the addresses for all the memory accesses, which is the DMA functionality. The major tasks handled by this block are:

- Fetching Endpoint Descriptors (ED) and Transfer Descriptors (TD)
- Read/Write endpoint data from/to system memory
- Accessing HC Communication Area (HCCA)
- Write Status and Retire TDs

11.2.5.5 USB State Control

This block implements:

- The USB operational states of the Host Controller, as defined in the OHCI Specification.
- It generates SOF tokens every 1 ms
- It triggers the List Processor while HC is in the operational states.

11.2.5.6 Data FIFO

This block contains a 64x8 FIFO to store the data returned by endpoints on IN tokens, and the data to be sent to the endpoints on OUT Tokens. The FIFO is used as a buffer in case the HC does not get timely access to the host bus.

11.2.5.7 List Processor

The List Processor processes the lists scheduled by HCD according to the priority set in the operational registers.

11.2.5.8 Root Hub and Host SIE

The Root Hub propagates Reset and Resume to downstream ports and handles port connect and disconnect. The Host Serial Interface Engine (HSIE) converts parallel to serial, serial to parallel, Non-Return to Zero Interface (NRZI) encoding/decoding and manages USB serial protocol.

Copyright 2007 Cirrus Logic

11.3 Registers

The Host Controller (HC) contains a set of on-chip operational registers that are used by the Host Controller Driver (HCD). According to the function of these registers, they are divided into four partitions, specifically for Control and Status, Memory Pointer, Frame Counter and Root Hub. All of the registers should be read and written as Dwords. The memory map is shown in Table 11-2.

Table 11-2. OpenHCI Register Addresses

Address	Register Name
0x8002_0000	HcRevision
0x8002_0004	HcControl
0x8002_0008	HcCommandStatus
0x8002_000C	HcInterruptStatus
0x8002_0010	HcInterruptEnable
0x8002_0014	HcInterruptDisable
0x8002_0018	HcHCCA
0x8002_001C	HcPeriodCurrentED
0x8002_0020	HcControlHeadED
0x8002_0024	HcControlCurrentED
0x8002_0028	HcBulkHeadED
0x8002_002C	HcBulkCurrentED
0x8002_0030	HcDoneHead
0x8002_0034	HcFmInterval
0x8002_0038	HcFmRemaining
0x8002_003C	HcFmNumber
0x8002_0040	HcPeriodicStart
0x8002_0044	HcLSThreshold
0x8002_0048	HcRhDescriptorA
0x8002_004C	HcRhDescriptorB
0x8002_0050	HcRhStatus
0x8002_0054	HcRhPortStatus[1]
0x8002_0058	HcRhPortStatus[2]
0x8002_005C	HcRhPortStatus[3]
0x8002_0080	USBCfgCtrl *
0x8002_0084	USBHCISts *

Note: In Table 11-2, "*" marks registers in address space 0x8002_0080 - 0x8002_0084 that are not OHCI implementation-specific. This address space is reserved for test software use.

Note: Important - Before setting up any of the Host controller registers it is necessary to set the USH_EN bit (bit 28 of the PwrCnt register).

Copyright 2007 Cirrus Logic

OpenHCI Implementation Specific Registers

The Root Hub partition contains registers that have power-on reset values that are implementation specific. The values for the processor are indicated in the Default field for each register, below.

HcRevision

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								REV							

Address:

0x8002_0000

Default:

0x0000_0010

Definition:

Defines the revision of the OHCI specification with which this implementation is compatible.

Bit Description:

RSVD: Reserved. Unknown During Read.

REV: This read-only field contains the BCD representation of the version of the HCI specification that is implemented by this HC.
0x10 = Compatible with OHCI 1.0.

HcControl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD					RWE	RWC	IR	HCFS		BLE	CLE	IE	PLE	CBSR	

Address:

0x8002_0004

Default:

0x0000_0000

Copyright 2007 Cirrus Logic

Definition:

Controls the host controller's operating modes.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

CBSR: ControlBulkServiceRatio:
This specifies the service ratio between Control and Bulk EDs. Before processing any of the nonperiodic lists, HC must compare the ratio specified with its internal count on how many nonempty Control EDs have been processed, in determining whether to continue serving another Control ED or switching to Bulk EDs. The internal count will be retained when crossing the frame boundary. In case of reset, HCD is responsible for restoring this value.
0 0 = 1:1
0 1 = 2:1
1 0 = 3:1
1 1 = 4:1

PLE: PeriodicListEnable:
This bit is set to enable the processing of the periodic list in the next Frame. If cleared by HCD, processing of the periodic list does not occur after the next SOF. HC must check this bit before it starts processing the list.

IE: IsochronousEnable:
This bit is used by HCD to enable/disable processing of isochronous EDs. While processing the periodic list in a Frame, HC checks the status of this bit when it finds an Isochronous ED (F=1). If set (enabled), HC continues processing the EDs. If cleared (disabled), HC halts processing of the periodic list (which now contains only isochronous EDs) and begins processing the Bulk/Control lists. Setting this bit is guaranteed to take effect in the next Frame (not the current Frame).

CLE: ControlListEnable:
This bit is set to enable the processing of the Control list in the next Frame. If cleared by HCD, processing of the Control list does not occur after the next SOF. HC must check this bit whenever it determines to process the list. When disabled, HCD may modify the list. If HcControlCurrentED is pointing to an ED to be removed, HCD must advance the pointer by updating HcControlCurrentED before re-enabling processing of the list.

Copyright 2007 Cirrus Logic

BLE: BulkListEnable:
This bit is set to enable the processing of the Bulk list in the next Frame. If cleared by HCD, processing of the Bulk list does not occur after the next SOF. HC checks this bit whenever it determines to process the list. When disabled, HCD may modify the list. If HcBulkCurrentED is pointing to an ED to be removed, HCD must advance the pointer by updating HcBulkCurrentED before re-enabling processing of the list.

HCFS: HostControllerFunctionalState:
A transition to USBOPERATIONAL from another state causes SOF generation to begin 1 ms later. HCD may determine whether HC has begun sending SOFs by reading the StartofFrame field of HcInterruptStatus. This field may be changed by HC only when in the USBSUSPEND state. HC may move from the USBSUSPEND state to the USBRESUME state after detecting the resume signaling from a downstream port. HC enters USBSUSPEND after a software reset, whereas it enters USBRESET after a hardware reset. The latter also resets the Root Hub and asserts subsequent reset signaling to downstream ports.
0 0 = USBRESET
0 1 = USBRESUME
1 0 = USBOPERATIONAL
1 1 = USBSUSPEND

IR: InterruptRouting:
This bit determines the routing of interrupts generated by events registered in HcInterruptStatus. If clear, all interrupts are routed to the normal host bus interrupt mechanism. If set, interrupts are routed to the System Management Interrupt. HCD clears this bit upon a hardware reset, but it does not alter this bit upon a software reset. HCD uses this bit as a tag to indicate the ownership of HC.

RWC: RemoteWakeupConnected:
This bit indicates whether HC supports remote wakeup signaling. If remote wakeup is supported and used by the system it is the responsibility of system firmware to set this bit during POST. HC clears the bit upon a hardware reset but does not alter it upon a software reset. Remote wakeup signaling of the host system is host-bus-specific and is not described in this specification.

Copyright 2007 Cirrus Logic

RWE: RemoteWakeupEnable:
This bit is used by HCD to enable or disable the remote wakeup feature upon the detection of upstream resume signaling. When this bit is set and the ResumeDetected bit in HcInterruptStatus is set, a remote wakeup is signaled to the host system. Setting this bit has no impact on the generation of hardware interrupt.

HcCommandStatus

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD														SOC	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												OCR	BLF	CLF	HCR

Address:
0x8002_0008

Default:
0x0000_0000

Definition:
Provides current controller status and accepts controller commands.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

HCR: HostControllerReset:
This bit is set by HCD to initiate a software reset of HC. Regardless of the functional state of HC, it moves to the USBSUSPEND state in which most of the operational registers are reset except those stated otherwise; e.g., the InterruptRouting field of HcControl, and no Host bus accesses are allowed. This bit is cleared by HC upon the completion of the reset operation. The reset operation must be completed within 10 ms. This bit, when set, should not cause a reset to the Root Hub and no subsequent reset signaling should be asserted to its downstream ports.

Copyright 2007 Cirrus Logic

CLF: ControlListFilled:
This bit is used to indicate whether there are any TDs on the Control list. It is set by HCD whenever it adds a TD to an ED in the Control list. When HC begins to process the head of the Control list, it checks CLF. As long as ControlListFilled is 0, HC will not start processing the Control list. If CF is 1, HC will start processing the Control list and will set ControlListFilled to 0. If HC finds a TD on the list, then HC will set ControlListFilled to 1 causing the Control list processing to continue. If no TD is found on the Control list, and if the HCD does not set ControlListFilled, then ControlListFilled will still be 0 when HC completes processing the Control list and Control list processing will stop.

BLF: BulkListFilled:
This bit is used to indicate whether there are any TDs on the Bulk list. It is set by HCD whenever it adds a TD to an ED in the Bulk list. When HC begins to process the head of the Bulk list, it checks BF. As long as BulkListFilled is 0, HC will not start processing the Bulk list. If BulkListFilled is 1, HC will start processing the Bulk list and will set BF to 0. If HC finds a TD on the list, then HC will set BulkListFilled to 1 causing the Bulk list processing to continue. If no TD is found on the Bulk list, and if HCD does not set BulkListFilled, then BulkListFilled will still be 0 when HC completes processing the Bulk list and Bulk list processing will stop.

OCR: OwnershipChangeRequest:
This bit is set by an OS HCD to request a change of control of the HC. When set HC will set the OwnershipChange field in HcInterruptStatus. After the changeover, this bit is cleared and remains so until the next request from OS HCD.

SOC: SchedulingOverrunCount:
These bits are incremented on each scheduling overrun error. It is initialized to 00b and wraps around at 11b. This will be incremented when a scheduling overrun is detected even if SchedulingOverrun in HcInterruptStatus has already been set. This is used by HCD to monitor any persistent scheduling problems.

Copyright 2007 Cirrus Logic

HcInterruptStatus

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD	OC	RSVD													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									RHSC	FNO	UE	RD	SF	WDH	SO

Address:

0x8002_000C

Default:

0x0000_0000

Definition:

Provides interrupt status information.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

SO: SchedulingOverrun. This bit is set when the USB schedule for the current Frame overruns and after the update of HccaFrameNumber. A scheduling overrun will also cause the SchedulingOverrunCount of HcCommandStatus to be incremented.

WDH: WritebackDoneHead. This bit is set immediately after HC has written HcDoneHead to HccaDoneHead. Further updates of the HccaDoneHead will not occur until this bit has been cleared. HCD should only clear this bit after it has saved the content of HccaDoneHead.

SF: StartofFrame. This bit is set by HC at each start of a frame and after the update of HccaFrameNumber. HC also generates a SOF token at the same time.

RD: ResumeDetected. This bit is set when HC detects that a device on the USB is asserting resume signaling. It is the transition from no resume signaling to resume signaling causing this bit to be set. This bit is not set when HCD sets the USBRESUME state.

UE: UnrecoverableError. This bit is set when HC detects a system error not related to USB. HC should not proceed with any processing nor signaling before the system error has been corrected. HCD clears this bit after HC has been reset.

Copyright 2007 Cirrus Logic

FNO: FrameNumberOverflow. This bit is set when the MSB of HcFmNumber (bit 15) changes value, from 0 to 1 or from 1 to 0, and after HccaFrameNumber has been updated.

RHSC: RootHubStatusChange. This bit is set when the content of HcRhStatus or the content of any of HcRhPortStatus[NumberofDownstreamPort] has changed.

OC: OwnershipChange. This bit is set by HC when HCD sets the OwnershipChangeRequest field in HcCommandStatus. This event, when unmasked, will always generate a System Management Interrupt (SMI) immediately. This bit is tied to 0b when the SMI pin is not implemented.

HcInterruptEnable

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MIE	OC	RSVD													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									RHSC	FNO	UE	RD	SF	WDH	SO

Address:

0x8002_0010

Default:

0x0000_0000

Definition:

Enables interrupt sources.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

SO: SchedulingOverrun. Enable interrupt generation due to Scheduling Overrun.

WDH: WritebackDoneHead. Enable interrupt generation due to HcDoneHead Writeback.

SF: StartofFrame. Enable interrupt generation due to Start of Frame.

RD: ResumeDetected. Enable interrupt generation due to Resume Detect.

UE: UnrecoverableError. Enable interrupt generation due to Unrecoverable Error.

Copyright 2007 Cirrus Logic

FNO: FrameNumberOverflow. Enable interrupt generation due to Frame Number Overflow.

RHSC: RootHubStatusChange. Enable interrupt generation due to Root Hub Status Change.

OC: OwnershipChange. Enable interrupt generation due to Ownership Change.

MIE: Master Interrupt Enable. A zero written to this field is ignored by HC. A one written to this field enables interrupt generation due to events specified in the other bits of this register. This is used by HCD as a Master Interrupt Enable.

HcInterruptDisable

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MIE	OC	RSVD													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									RHSC	FNO	UE	RD	SF	WDH	SO

Address:
0x8002_0014

Default:
0x0000_0000

Definition:
Disables interrupt sources.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

SO: SchedulingOverrun: Disable interrupt generation due to Scheduling Overrun.

WDH: WritebackDoneHead: Disable interrupt generation due to HcDoneHead Writeback.

SF: StartofFrame: Disable interrupt generation due to Start of Frame.

RD: ResumeDetected: Disable interrupt generation due to Resume Detect.

UE: UnrecoverableError: Disable interrupt generation due to Unrecoverable Error.

Copyright 2007 Cirrus Logic

FNO:	FrameNumberOverflow: Disable interrupt generation due to Frame Number Overflow.
RHSC:	RootHubStatusChange: Disable interrupt generation due to Ownership Change.
OC:	OwnershipChange. Enable interrupt generation due to Ownership Change.
MIE:	Master Interrupt Enable: A zero written to this field is ignored by HC. A one written to this field disables interrupt generation due to events specified in the other bits of this register. This field is set after a hardware or software reset.

HcHCCA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AD								RSVD							

Address:

0x8002_0018

Default:

0x0000_0000

Definition:

Base physical address of the Host Controller Communication Area.

Bit Description:

RSVD:	Reserved. Unknown During Read.
AD:	HCCA. Base physical address of the Host Controller Communication Area.

HcPeriodCurrentED

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AD												RSVD			

Address:

0x8002_001C

Copyright 2007 Cirrus Logic

Default:

0x0000_0000

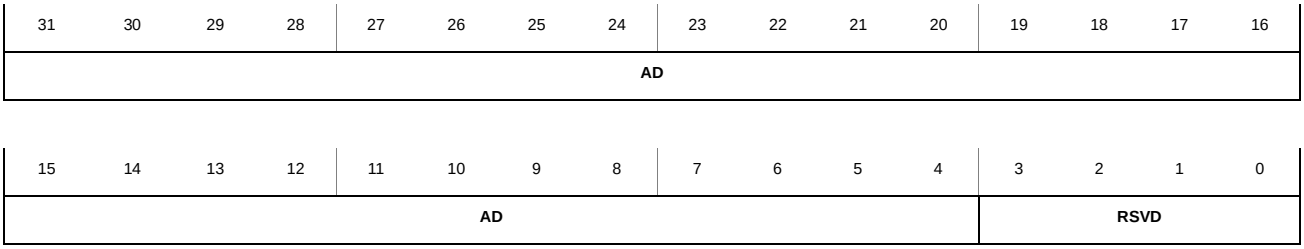
Definition:

Physical address of the current isochronous or interrupt endpoint descriptor.

Bit Description:

RSVD:	Reserved. Unknown During Read.
AD:	PeriodCurrentED. This is used by HC to point to the head of one of the Periodic lists which will be processed in the current Frame. The content of this register is updated by HC after a periodic ED has been processed. HCD may read the content in determining which ED is currently being processed at the time of reading.

HcControlHeadED



Address:

0x8002_0020

Default:

0x0000_0000

Definition:

Physical address of the first endpoint descriptor of the control list.

Bit Description:

RSVD:	Reserved. Unknown During Read.
AD:	ControlHeadED. HC traverses the Control list starting with the HcControlHeadED pointer. The content is loaded from HCCA during the initialization of HC.

Copyright 2007 Cirrus Logic

HcControlCurrentED

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AD												RSVD			

Address:

0x8002_0024

Default:

0x0000_0000

Definition:

Physical address of the current endpoint descriptor of the control list.

Bit Description:

RSVD: Reserved. Unknown During Read.

AD: ControlCurrentED. This pointer is advanced to the next ED after serving the present one. HC will continue processing the list from where it left off in the last Frame. When it reaches the end of the Control list, HC checks the ControlListFilled of HcCommandStatus. If set, it copies the content of HcControlHeadED to HcControlCurrentEDand clears the bit. If not set, it does nothing. HCD is allowed to modify this register only when the ControlListEnable of HcControl is cleared. When set, HCD only reads the instantaneous value of this register. Initially, this is set to zero to indicate the end of the Control list.

HcBulkHeadED

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AD												RSVD			

Address:

0x8002_0028

Default:

0x0000_0000

Definition:

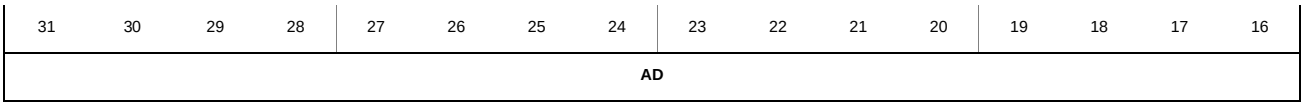
Copyright 2007 Cirrus Logic

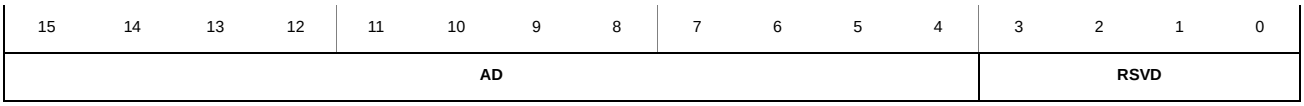
Physical address of the first endpoint descriptor of the bulk list.

Bit Description:

RSVD:	Reserved. Unknown During Read.
AD:	BulkHeadED. HC traverses the Bulk list starting with the HcBulkHeadED pointer. The content is loaded from HCCA during the initialization of HC.

HcBulkCurrentED





Address:

0x8002_002C

Default:

0x0000_0000

Definition:

Physical address of the current endpoint descriptor of the bulk list.

Bit Description:

RSVD:	Reserved. Unknown During Read.
AD:	BulkCurrentED. This is advanced to the next ED after the HC has served the present one. HC continues processing the list from where it left off in the last Frame. When it reaches the end of the Bulk list, HC checks the ControlListFilled of HcControl. If set, it copies the content of HcBulkHeadED to HcBulkCurrentED and clears the bit. If it is not set, it does nothing. HCD is only allowed to modify this register when the BulkListEnable of HcControl is cleared. When set, the HCD only reads the instantaneous value of this register. This is initially set to zero to indicate the end of the Bulk list.

Copyright 2007 Cirrus Logic

HcDoneHead

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AD												RSVD			

Address:

0x8002_0030

Default:

0x0000_0000

Definition:

Physical address of the last completed transfer descriptor that was added to the done list.

Bit Description:

RSVD: Reserved. Unknown During Read.

AD: DoneHead. When a TD is completed, HC writes the content of HcDoneHead to the NextTD field of the TD. HC then overwrites the content of HcDoneHead with the address of this TD. This is set to zero whenever HC writes the content of this register to HCCA. It also sets the WritebackDoneHead of HcInterruptStatus.

HcFmInterval

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIT	FSMPS														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	FI														

Address:

0x8002_0034

Default:

0x0000_2EDF

Definition:

Describes the bit time interval in a frame and the full speed maximum packet size.

Bit Descriptions:

Copyright 2007 Cirrus Logic

RSVD: Reserved. Unknown During Read.

FI: FrameInterval. This specifies the interval between two consecutive SOFs in bit times. The nominal value is set to be 11,999. HCD should store the current value of this field before resetting HC. By setting the HostControllerReset field of HcCommandStatus as this will cause the HC to reset this field to its nominal value. HCD may choose to restore the stored value upon the completion of the Reset sequence.

FSMPS: FSLargestDataPacket. This field specifies a value which is loaded into the Largest Data Packet Counter at the beginning of each frame. The counter value represents the largest amount of data in bits which can be sent or received by the HC in a single transaction at any given time without causing scheduling overrun. The field value is calculated by the HCD.

FIT: FrameIntervalToggle. HCD toggles this bit whenever it loads a new value to FrameInterval.

HcFmRemaining

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FRT	RSVD														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	FR														

Address:

0x8002_0038

Default:

0x0000_0000

Definition:

Contains the time remaining in the current frame.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

FR: FrameRemaining. This counter is decremented at each bit time. When it reaches zero, it is reset by loading the FrameInterval value specified in HcFmInterval at the next bit time boundary. When entering the USBOPERATIONAL state, HC re-loads the content with the FrameInterval of HcFmInterval and uses the updated value from the next SOF.

Copyright 2007 Cirrus Logic

FRT: FrameRemainingToggle. This bit is loaded from the FrameIntervalToggle field of HcFmInterval whenever FrameRemaining reaches 0. This bit is used by HCD for the synchronization between FrameInterval and FrameRemaining.

HcFmNumber

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FN															

Address:

0x8002_003C

Default:

0x0000_0000

Definition:

Contains a 16-bit counter used as a timing reference between the host controller and its driver.

Bit Description:

RSVD: Reserved. Unknown During Read.

FN: FrameNumber. This is incremented when HcFmRemaining is re-loaded. It will be rolled over to 0x0 after 0xFFFF. When entering the USBOPERATIONAL state, this will be incremented automatically. The content will be written to HCCA after HC has incremented the FrameNumber at each frame boundary and sent a SOF but before HC reads the first ED in that Frame. After writing to HCCA, HC will set the StartofFrame in HcInterruptStatus.

HcPeriodicStart

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				PS											

Address:

Copyright 2007 Cirrus Logic

0x8002_0040

Default:

0x0000_0000

Definition:

Defines the earliest time the host controller should start processing the periodic list.

Bit Description:

RSVD: Reserved. Unknown During Read.

PS: PeriodicStart. After a hardware reset, this field is cleared. This is then set by HCD during the HC initialization. The value is calculated roughly as 10% off from HcFmInterval. A typical value will be 0x03E67. When HcFmRemaining reaches the value specified, processing of the periodic lists will have priority over Control/Bulk processing. HC will therefore start processing the Interrupt list after completing the current Control or Bulk transaction that is in progress.

HcLSThreshold

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				LST											

Address:

0x8002_0044

Default:

0x0000_0628

Definition:

Contains a value used by the host controller to determine whether to commit to the transfer of a maximum 8-byte LS packet before EOF.

Bit Description:

RSVD: Reserved. Unknown During Read.

LST: LSThreshold. This field contains a value which is compared to the FrameRemaining field prior to initiating a Low Speed transaction. The transaction is started only if FrameRemaining >= this field. The value is calculated by HCD with the consideration of transmission and setup overhead.

Copyright 2007 Cirrus Logic

HcRhDescriptorA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P								RSVD							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD			NOCP	OCPM	DT	NPS	PSM	NDP							

Address:

0x8002_0048

Default:

0x0200_1203

Definition:

Describes the root hub.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

NDP: NumberDownstreamPorts. These bits specify the number of downstream ports supported by the Root Hub. It is implementation-specific. The minimum number of ports is 1. The maximum number of ports supported by OpenHCI is 15.
0x03 = 3 downstream ports.

PSM: PowerSwitchingMode. This bit is used to specify how the power switching of the Root Hub ports is controlled. It is implementation-specific. This field is only valid if the NoPowerSwitching field is cleared.
0: All ports are powered at the same time.
1: Each port is powered individually.

This mode allows port power to be controlled by either the global switch or per-port switching. If the PortPowerControlMask bit is set, the port responds only to port power commands (Set/ClearPortPower). If the port mask is cleared, the port is controlled only by the global power switch (Set/ClearGlobalPower).

Copyright 2007 Cirrus Logic

NPS: NoPowerSwitching. These bits are used to specify whether power switching is supported or port are always powered. It is implementation-specific. When this bit is cleared, the PowerSwitchingMode specifies global or per-port switching.
0: Ports are power switched
1: Ports are always powered on when the HC is powered on.

DT: DeviceType. This bit specifies that the Root Hub is not a compound device. The Root Hub is not permitted to be a compound device. This field should always read/write 0.

OCPM: OverCurrentProtectionMode. This bit describes how the overcurrent status for the Root Hub ports are reported. At reset, this fields should reflect the same mode as PowerSwitchingMode. This field is valid only if the NoOverCurrentProtection field is cleared.
0: Over-current status is reported collectively for all downstream ports
1: Over-current status is reported on a per-port basis.

NOCP: NoOverCurrentProtection. This bit describes how the overcurrent status for the Root Hub ports are reported. When this bit is cleared, the OverCurrentProtectionMode field specifies global or per-port reporting.
0: Over-current status is reported collectively for all downstream ports
1: No overcurrent protection supported

P: PowerOnToPowerGoodTime. This byte specifies the duration HCD has to wait before accessing a powered-on port of the Root Hub. It is implementation-specific. The unit of time is 2 ms. The duration is calculated as P[7:0] * 2 ms. 0x05 = 10 ms

HcRhDescriptorB

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD												PPCM			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												DR			

Address: 0x8002_004C

Copyright 2007 Cirrus Logic

Default:

0x0000_0000

Definition:

Describes the root hub.

Bit Descriptions

RSVD:	Reserved. Unknown During Read.
DR:	DeviceRemovable. Each bit is dedicated to a port of the Root Hub. When cleared, the attached device is removable. When set, the attached device is not removable. bit 0: Reserved bit 1: Device attached to Port #1 bit 2: Device attached to Port #2 bit 3: Device attached to Port #3
PPCM:	PortPowerControlMask: Each bit indicates if a port is affected by a global power control command when PowerSwitchingMode is set. When set, the port's power state is only affected by per-port power control (Set/ClearPortPower). When cleared, the port is controlled by the global power switch (Set/ClearGlobalPower). If the device is configured to global switching mode (PowerSwitchingMode=0), this field is not valid. bit 0: Reserved bit 1: Ganged-power mask on Port #1 bit 2: Ganged-power mask on Port #2 bit 3: Ganged-power mask on Port #3

HcRhStatus

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRWE	RSVD													CCIC	LPSC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DRWE	RSVD													OCI	LPS

Address:

0x8002_0050

Default:

0x0000_0000

Definition:

Root hub status.

Bit Descriptions:

Copyright 2007 Cirrus Logic

RSVD:	Reserved. Unknown During Read.
LPS:	(READ) LocalPowerStatus. The Root Hub does not support the local power status feature; thus, this bit is always read as "0". (WRITE) ClearGlobalPower: In global power mode (PowerSwitchingMode=0), this bit is written to "1" to turn off power to all ports (clear PortPowerStatus). In per-port power mode, it clears PortPowerStatus only on ports whose PortPowerControlMask bit is not set. Writing a "0" has no effect.
OCI:	OverCurrentIndicator. This bit reports overcurrent conditions when the global reporting is implemented. When set, an overcurrent condition exists. When cleared, all power operations are normal. If per-port overcurrent protection is implemented this bit is always "0"
DRWE:	(READ) DeviceRemoteWakeupEnable. This bit enables a ConnectStatusChange bit as a resume event, causing a USBSUSPEND to USBRESUME state transition and setting the ResumeDetected interrupt. 0 = ConnectStatusChange is not a remote wakeup event. 1 = ConnectStatusChange is a remote wakeup event. (WRITE) SetRemoteWakeupEnable: Writing a '1' sets DeviceRemoveWakeupEnable. Writing a '0' has no effect.
LPSC:	(READ) LocalPowerStatusChange. The Root Hub does not support the local power status feature; thus, this bit is always read as "0". (WRITE) SetGlobalPower. In global power mode (PowerSwitchingMode=0), This bit is written to "1" to turn on power to all ports (clear PortPowerStatus). In per-port power mode, it sets PortPowerStatus only on ports whose PortPowerControlMask bit is not set. Writing a "0" has no effect.
CCIC:	OverCurrentIndicatorChange. This bit is set by hardware when a change has occurred to the OCI field of this register. The HCD clears this bit by writing a "1". Writing a "0" has no effect.
CRWE:	(WRITE) ClearRemoteWakeupEnable. Writing a '1' clears DeviceRemoveWakeupEnable. Writing a '0' has no effect.

Copyright 2007 Cirrus Logic

HcRhPortStatusx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD											PRSC	OCIC	PSSC	PESC	CSC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD						LSDA	PPS	RSVD			PRS	POCI	PSS	PES	CCS

Address:

HcRhPortStatus1 - 0x8002_0054,
HcRhPortStatus2 - 0x8002_0058,
HcRhPortStatus3 - 0x8002_005C

Default:

0x0000_0100

Definition:

Control/status for root hub port 1, 2, and 3 respectively

Bit Descriptions:

CCS: (READ) CurrentConnectStatus: This bit reflects the current state of the downstream port.
0 = no device connected
1 = device connected

(WRITE) ClearPortEnable: The HCD writes a "1" to this bit to clear the PortEnableStatus bit. Writing a "0" has no effect. The CurrentConnectStatus is not affected by any write.

Note: This bit is always read "1" when the attached device is nonremovable (DeviceRemoveable.NDP).

Copyright 2007 Cirrus Logic

PES: (READ) PortEnableStatus. This bit indicates whether the port is enabled or disabled. The Root Hub may clear this bit when an overcurrent condition, disconnect event, switched-off power, or operational bus error such as babble is detected. This change also causes PortEnabledStatusChange to be set. HCD sets this bit by writing SetPortEnable and clears it by writing ClearPortEnable. This bit cannot be set when CurrentConnectStatus is cleared. This bit is also set, if not already, at the completion of a port reset when ResetStatusChange is set or port suspend when SuspendStatusChange is set.
0 = port is disabled
1 = port is enabled

(WRITE) SetPortEnable. The HCD sets PortEnableStatus by writing a "1". Writing a "0" has no effect. If CurrentConnectStatus is cleared, this write does not set PortEnableStatus, but instead sets ConnectStatusChange. This informs the driver that it attempted to enable a disconnected port.

PSS: (READ) PortSuspendStatus. This bit indicates the port is suspended or in the resume sequence. It is set by a SetSuspendState write and cleared when PortSuspendStatusChange is set at the end of the resume interval. This bit cannot be set if CurrentConnectStatus is cleared. This bit is also cleared when PortResetStatusChange is set at the end of the port reset or when the HC is placed in the USBRESUME state. If an upstream resume is in progress, it should propagate to the HC.
0 = port is not suspended
1 = port is suspended

(WRITE) SetPortSuspend. The HCD sets the PortSuspendStatus bit by writing a "1" to this bit. Writing a "0" has no effect. If CurrentConnectStatus is cleared, this write does not set PortSuspendStatus; instead it sets ConnectStatusChange. This informs the driver that it attempted to suspend a disconnected port.

Copyright 2007 Cirrus Logic

POCI: (READ) PortOverCurrentIndicator. This bit is only valid when the Root Hub is configured in such a way that overcurrent conditions are reported on a per-port basis. If per-port overcurrent reporting is not supported, this bit is set to 0. If cleared, all power operations are normal for this port. If set, an overcurrent condition exists on this port. This bit always reflects the overcurrent input signal
0 = no overcurrent condition.
1 = overcurrent condition detected.

(WRITE) ClearSuspendStatus. The HCD writes a "1" to initiate a resume. Writing a "0" has no effect. A resume is initiated only if PortSuspendStatus is set.

PRS: (READ) PortResetStatus. When this bit is set by a write to SetPortReset, port reset signaling is asserted. When reset is completed, this bit is cleared when PortResetStatusChange is set. This bit cannot be set if CurrentConnectStatus is cleared.
0 = port reset signal is not active
1 = port reset signal is active

(WRITE) SetPortReset. The HCD sets the port reset signaling by writing a "1" to this bit. Writing a "0" has no effect. If CurrentConnectStatus is cleared, this write does not set PortResetStatus, but instead sets ConnectStatusChange. This informs the driver that it attempted to reset a disconnected port.

PPS: (READ) PortPowerStatus. This bit reflects the port's power status, regardless of the type of power switching implemented. This bit is cleared if an overcurrent condition is detected. HCD sets this bit by writing SetPortPower or SetGlobalPower. HCD clears this bit by writing ClearPortPower or ClearGlobalPower. Which power control switches are enabled is determined by PowerSwitchingMode and PortPortControlMask[NDP].

In global switching mode (PowerSwitchingMode=0), only Set/ClearGlobalPower controls this bit. In per-port power switching (PowerSwitchingMode=1), if the PortPowerControlMask[NDP] bit for the port is set, only Set/ClearPortPower commands are enabled. If the mask is not set, only Set/ClearGlobalPower commands are enabled. When port power is disabled, CurrentConnectStatus, PortEnableStatus, PortSuspendStatus, and PortResetStatus should be reset.
0 = port power is off
1 = port power is on

Copyright 2007 Cirrus Logic

(WRITE) SetPortPower: The HCD writes a "1" to set the PortPowerStatus bit. Writing a "0" has no effect.

Note: This bit is always reads "1" if power switching is not supported.

LSDA: (READ) LowSpeedDeviceAttached. This bit indicates the speed of the device attached to this port. When set, a Low Speed device is attached to this port. When clear, a Full Speed device is attached to this port. This field is valid only when the CurrentConnectStatus is set.
0 = full speed device attached
1 = low speed device attached

(WRITE) ClearPortPower. The HCD clears the PortPowerStatus bit by writing a "1" to this bit. Writing a "0" has no effect.

CSC: ConnectStatusChange. This bit is set whenever a connect or disconnect event occurs. The HCD writes a "1" to clear this bit. Writing a "0" has no effect. If CurrentConnectStatus is cleared when a SetPortReset, SetPortEnable, or SetPortSuspend write occurs, this bit is set to force the driver to re-evaluate the connection status since these writes should not occur if the port is disconnected.
0 = no change in CurrentConnectStatus
1 = change in CurrentConnectStatus

Note: If the DeviceRemovable.NDP bit is set, this bit is set only after a Root Hub reset to inform the system that the device is attached.

PESC: PortEnableStatusChange. This bit is set when hardware events cause the PortEnableStatus bit to be cleared. Changes from HCD writes do not set this bit. The HCD writes a "1" to clear this bit. Writing a "0" has no effect.
0 = no change in PortEnableStatus
1 = change in PortEnableStatus

PSSC: PortSuspendStatusChange. This bit is set when the full resume sequence has been completed. This sequence includes the 20 ms resume pulse, LS EOP, and 3 ms re-synchronization delay. The HCD writes a "1" to clear this bit. Writing a "0" has no effect. This bit is also cleared when ResetStatusChange is set.
0 = resume is not completed
1 = resume completed

Copyright 2007 Cirrus Logic

OCIC: PortOverCurrentIndicatorChange. This bit is valid only if overcurrent conditions are reported on a per-port basis. This bit is set when Root Hub changes the PortOverCurrentIndicator bit. The HCD writes a "1" to clear this bit. Writing a "0" has no effect.
0 = no change in PortOverCurrentIndicator
1 = PortOverCurrentIndicator has changed

PRSC: PortResetStatusChange. This bit is set at the end of the 10 ms port reset signal. The HCD writes a "1" to clear this bit. Writing a "0" has no effect.
0 = port reset is not complete
1 = port reset is complete

USBCfgCtrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD											TRCS	TPOC			RSVD

Address:
0x8002_0080 - Read/Write

Default:
0x0000_0000

Definition:
Used to implement some input signals to USB host controller for configuration through software.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

TPOC: When asserted by software, the corresponding port will enter DISCONNECT state. These bits must be cleared before the ports can be reused.

TRCS: Inverted internally and sent out as APP_CntSelN signal to uhostc_top. Internally known as TicRegCntSel. APP_CntSelN is used for selecting the counter value for either simulation or real-time for the 1 ms frame duration used internally. It should be usually set to "0". Setting it to "1" will cause the internal counter count to be a partial full count.

Copyright 2007 Cirrus Logic

USBHCISts

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD													RWU	MSN	MBA

Address:

0x8002_0084 - Read/Write

Default:

0x0000_0000

Definition:

Host Controller Interface. Some status bits reporting from USB host controller to software.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

MBA: Host controller buffer access indication. When asserted, it indicates that currently host controller is accessing data buffer. It is a status bit reporting to software and software does not need to take any action.

MSN: Host controller new frame. Software does not need to take any action because it is a status about a new frame that is generated.

RWU: Host controller remote wakeup. Software action when this bit is asserted is implementation specific. It is a status bit reporting a transition of internal state.

Copyright 2007 Cirrus Logic

11

Copyright 2007 Cirrus Logic

Chapter 12

Static Memory Controller

12

12.1 Introduction

Note: In the EP9301 and 9302 processors, the common address/data bus is 16-bits wide and the Static Memory Controller (SMC) supports 8-bit and 16-bit devices.

Note: In the EP9307, EP9312, and EP9315 processors, the common address/data bus is programmable to either 16-bits or 32-bits wide, and the SMC supports 8-bit, 16-bit, and 32-bit devices.

Note: PCMCIA (PC Card) is supported in the EP9315 processor only.

The Static Memory Controller (SMC) operates in little endian mode, and it supports up to six independently configurable memory spaces or banks. Supported memory types are:

- SRAM
- ROM
- NOR FLASH
- External Peripheral that has an SRAM-type interface

Each memory bank can be configured to support:

- Memory devices that have either 8-, 16-, or 32-bit data paths. For example:
  - Two 16-bit devices can be used in parallel to make a 32-bit data path
  - Two 8-bit devices can be used in parallel to make a 16-bit data path
  - One 16-bit device can be used standalone to make an 16-bit data path
  - One 8-bit device can be used standalone to make an 8-bit data path
- One external peripheral that uses the external DMA handshake signals, DREQ0/1, DACK0/1, and optionally DEOT0/1. These signals are multiplexed with EPGIO[12:7] pins.

Note: There are only two external DMA interfaces total on the EP93xx device to control data flow

- Non-burst read and write accesses
- Page Mode (burst-of-four) read and write accesses
- PCMCIA interfacing (EP9315 processor only)

Copyright 2007 Cirrus Logic

The SMC has five main functions:

1. Memory bank selecting
2. Access timing
3. Wait State generation
4. Byte lane write enabling
5. External bus interfacing

12

12.2 Static Memory Controller Operation

The SMC provides access to static memory devices that are attached to the external bus. The SMC can work with a wide variety of external device types, including SRAM, ROM, NOR FLASH, and peripherals that respond to SRAM-type signaling.

Six chip-select output signals, CSn7, CSn6, CSn3, CSn2, CSn1, and CSn0 can be used to access six different memory spaces. However, only one of the six memory banks can be accessed at a given time. The SMC has six independent control registers that configure the six respective chip-select signals. Each control register, "SMCBCR[7:0]" specifies the timing characteristics that are needed to access the memory device(s) in its respective memory space.

As shown in Figure 12-1 and Figure 12-3, the SMC captures read data on the HCLK edge that occurs just prior to the HCLK edge that de-asserts the chip-select output signal on the CSnX pin. The output signal on the CSnX pin and the address outputs on the AD[x] pins are de-asserted on the next HCLK edge.

The SMC can insert wait cycles into its access timing. Wait cycles can be specified by:

- A programmable value, N, where N has the range 1<N<32. When N is used, the SMC holds its bus state for N HCLK cycles. The value for N must be written to the WST2 and/or WST1 fields of the "SMCBCR[7:0]" register(s).
- An asserted wait input signal on the WAITn pin. As shown in Figure 12-3 and Figure 12-4, the WAITn pin can be asserted as needed by an external device to extend access time. When WAITn is asserted, the SMC holds its bus state until WAITn is sampled as being de-asserted. For internal synchronization to occur, WAITn must remain asserted for a minimum of two HCLK cycles.
- When both N and WAITn are used, the SMC holds its bus state for N HCLK cycles or until WAITn is sampled as being de-asserted, whichever occurs last.

Copyright 2007 Cirrus Logic

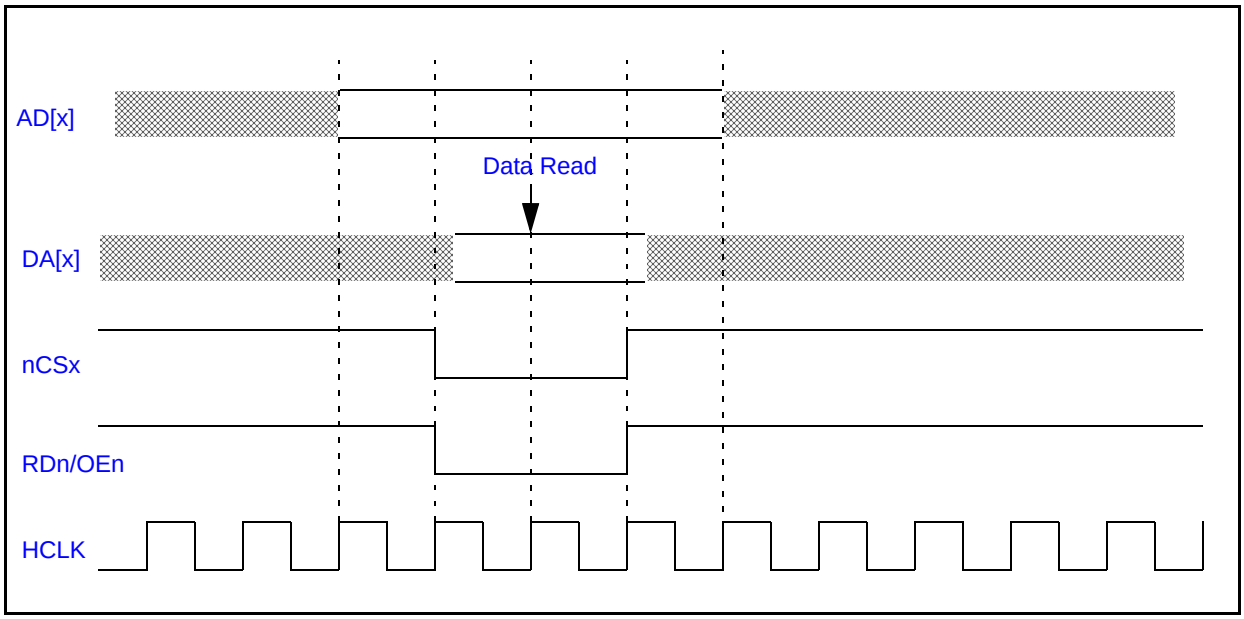


Figure 12-1. 32-bit Read, 32-bit Memory, 0 Wait Cycles, RBLE = 1, WAITn Inactive

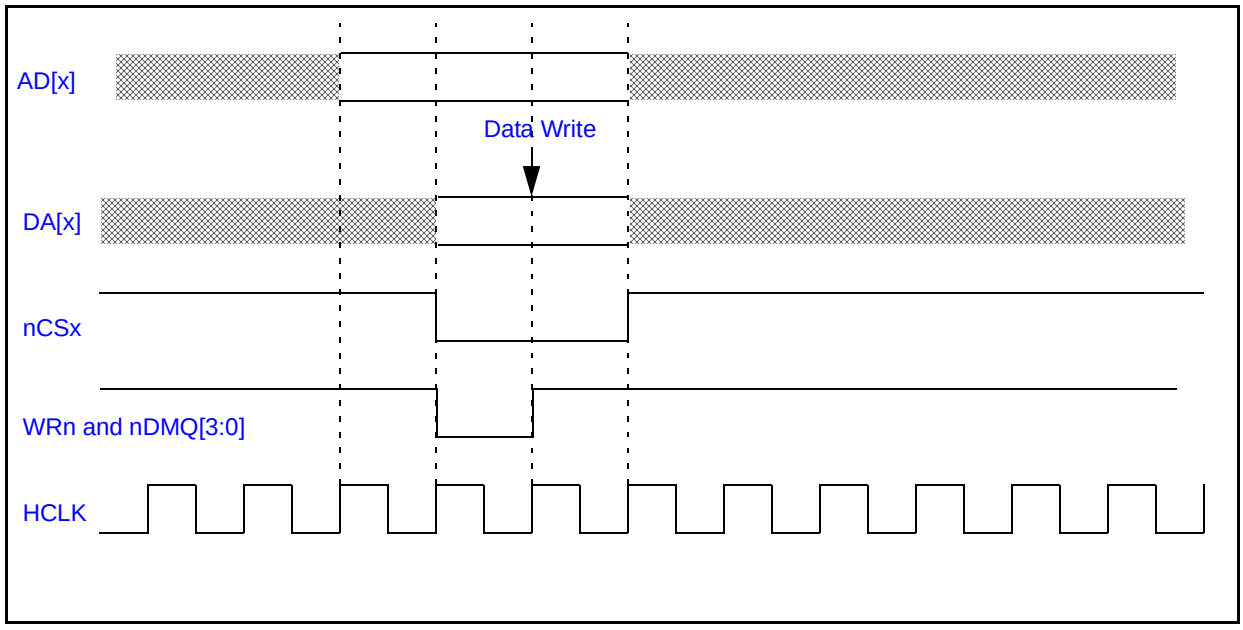


Figure 12-2. 32-bit Write, 32-bit Memory, 0 Wait Cycles, RBLE = 1, WAITn Inactive

Copyright 2007 Cirrus Logic

12

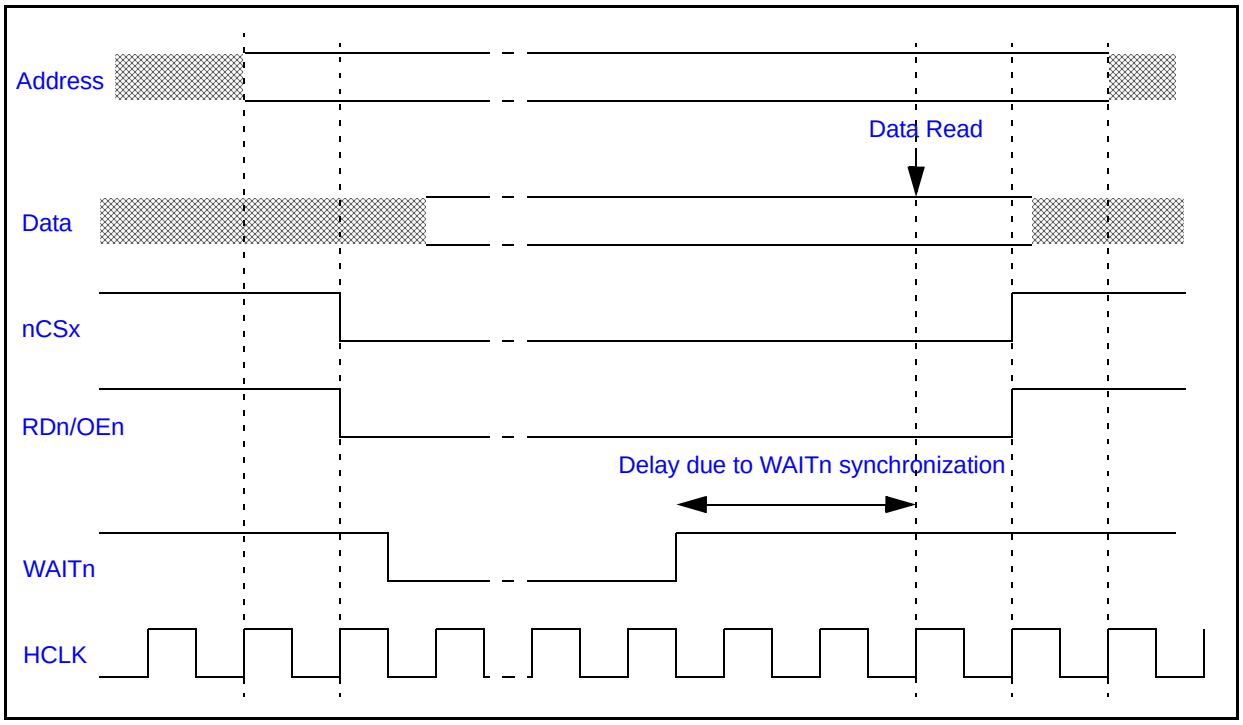


Figure 12-3. 16-bit Read, 16-bit Memory, RBLE = 1, WAITn Active

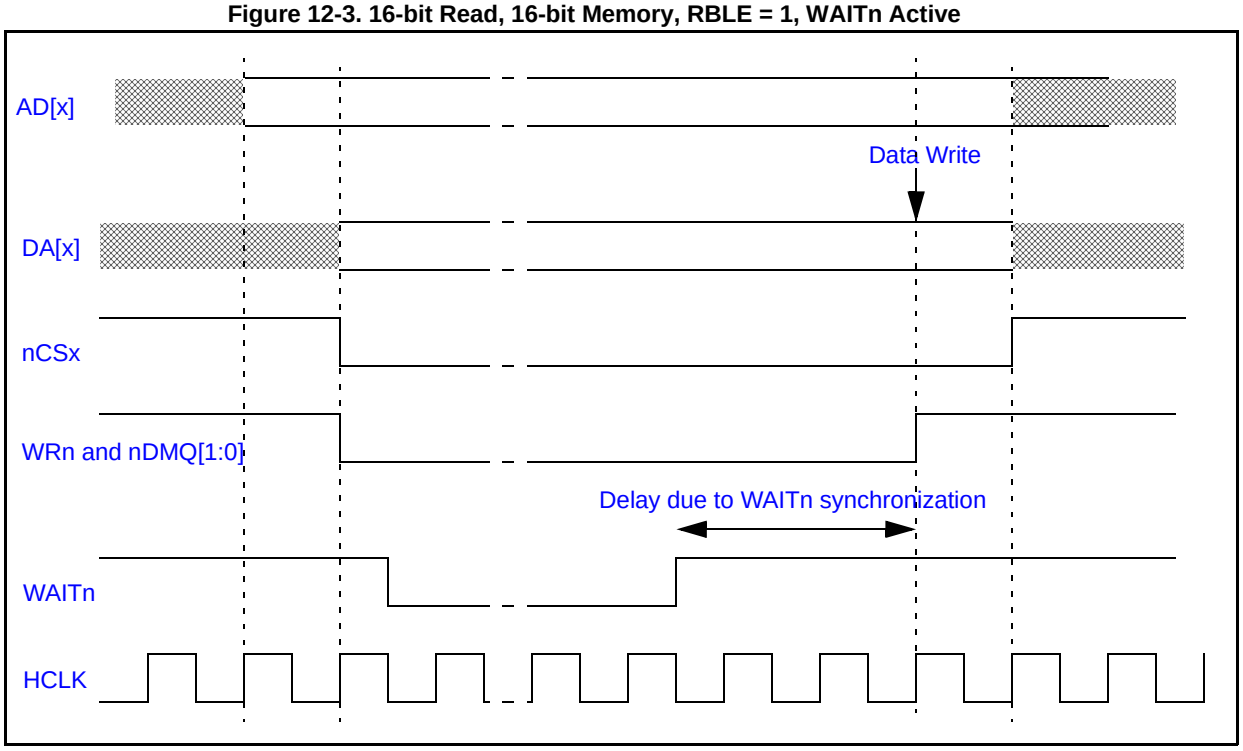


Figure 12-4. 16-bit Write, 16-bit Memory, RBLE = 1, WAITn Active

Copyright 2007 Cirrus Logic

If the bit-width of an internal device that generates a read or write request is larger than the bit-width of the memory device in the target memory space, the SMC will perform multiple successive read or write accesses to the external device. For example, if an internal device generates a 16-bit read request to an 8-bit external memory device, the SMC will perform two successive read accesses to the 8-bit external device. The 8-bit data from the 1st read is stored within the SMC until the 8-bit data from the 2nd read arrives. The SMC then combines the data from the 1st and 2nd 8-bit read to form the requested 16-bit read data. The bus that connects the internal device to the SMC cannot be used for any other purpose until after the requested 16-bit read data is latched into the internal device.

During a write cycle, four byte lane output signals on the DQMn[3:0] pins notify the external memory device of which byte lanes it should accept data from. See Figure 12-2. For example, when the SBC performs an upper half-word (16-bit) write to a 32-bit-wide external memory (32-bit bus), the SMC would output DQMn[3:0] = '0011' to notify the external memory that it should accept write data only from the upper two bytes on the 32-bit bus, and not accept data from the lower two bytes on the 32-bit bus. In other words, the upper two bytes in the 32-bit-wide memory would be written and the lower two bytes would remain as they are (unwritten).

Each memory bank can be specified to operate with either single read and write accesses or with burst-of-four (page mode) read and write accesses. During burst-of-four accesses, the A[3] and A[4] address bits are internally incremented, '00' –> '01' –> '10' –> '11', to access four sequential words. When using burst-of-four accesses, the address of the first access must be on a quad-word address boundary. Burst-of-four or non-burst accesses are specified by the value written to the PME bit in a bank control (SMCBCRx) register.

Note: The external device must support burst-of-four accesses.

12.3 PCMCIA Interface (EP9315 Processor Only)

With external logic, the PCMCIA Interface supports a PC Card in Slot 0 at 0x4000_0000. Table 12-1 shows the memory address ranges. Address, data, and control signals for interfacing to a PC Card are shown in Table 12-2.

Table 12-1. PCMCIA Address Memory Ranges

Memory Space	Bit [27:26]	Address Range
IO	**00**	0x4000_0000 - 0x43FF_FFFF
Undefined	**01**	0x4400_0000 - 0x47FF_FFFF
Attribute	**10**	0x4800_0000 - 0x4BFF_FFFF
Memory	**11**	0x4C00_0000 - 0x4FFF_FFFF

Table 12-2. PCMCIA Pin Usage

Pin Name	Alternate Use If No Card	PCMCIA Signal Name	Note:
MCRDn		nPOE	1
MCWRn		nPWE	1
IORDn		nPIORD	1

Copyright 2007 Cirrus Logic

Table 12-2. PCMCIA Pin Usage (Continued)

Pin Name	Alternate Use If No Card	PCMCIA Signal Name	Note:
IOWRn		nPIOWR	1
MCREGn		nPREG	1
MCELn		nPC_CE1	1
MCEHn		nPC_CE2	1
MCRESETn		RESET_1	1
MCWAIT		nWAIT	1
AD[10:8]		PC_A[10:8]	1
AD[7:0]		PC_A[7:0]	2
DA[15:0]		PC_D[15:0]	2
MCDIR		PC_DIR	2
MCDAENn		-	2
MCADENn		-	2
VS2	GPIO.F[7]	VS2	2
READY	GPIO.F[6]	PC_RDY	2
VS1	GPIO.F[5]	VS1	2
MCBVD2	GPIO.F[4]	MCBVD2	2
MCBVD1	GPIO.F[3]	MCBVD1	2
MCD2	GPIO.F[2]	MCD2	2
MCD1	GPIO.F[1]	MCD1	2
WP	GPIO.F[0]	WP	2
Not Implemented	Not Implemented	nIOIS16	3

1. These signals go directly to the inserted PC card.
2. These signals require external logic to interface to the PC card.
3. The PCMCIA signal, nIOIS16, is not supported. The IO width can be determined by reading the card attribute memory, and programming the IO space registers accordingly.

External logic, as shown in Figure 12-5, is required to connect some PCMCIA card signals to the processor. Other PCMCIA card signals, also shown in Figure 12-5, connect directly to the processor.

Copyright 2007 Cirrus Logic

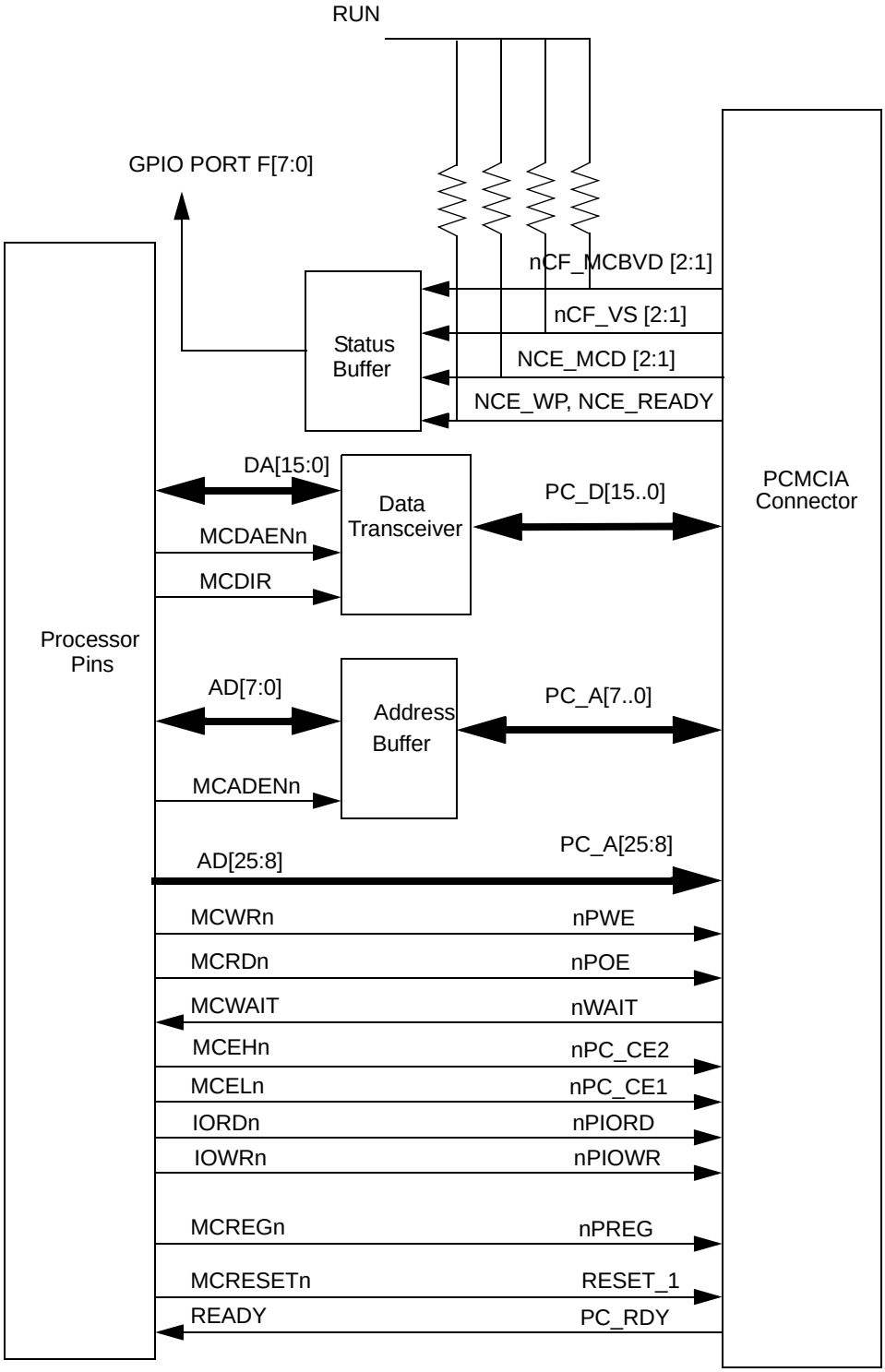


Figure 12-5. Single PC Card Interface

Copyright 2007 Cirrus Logic

12.4 PC Card Memory-Mode Enable Signals

PC Card memory-mode enable signals, nPC_CE1 and nPC_CE2, are output on pin MCELn and pin MCEHn, respectively. Along with the address signal output on pin AD[0] and the data signals input or output on pins DA[15:8] and DA[7:0], the nPC_CE1 and nPC_CE2 signals specify the type of access that is being made to the particular segment of memory in the PC Card, as shown in Table 12-3 and Table 12-4.

Table 12-3. Supported 8-Bit Accesses

Access	nPC_CE2	nPC_CE1	A0	D15-D8	D7-D0
Stand by (no access)	1	1	X	Z	Z
Even Byte Access	0	0	0	Z	Even Byte
Odd Byte Access	1	0	1	Z	Odd Byte

Table 12-4. Supported 16-Bit Accesses

Access	nPC_CE2	nPC_CE1	A0	D15-D8	D7-D0
Stand by (no access)	1	1	X	Z	Z
Even Byte Access	1	0	0	Z	Even Byte
Odd Byte Access	1	0	1	Z	Odd Byte
Both Byte Access	0	0	Z	Odd Byte	Even Byte

Note: Prior to version 8.0 of the PCMCIA specification, two valid types of odd-byte accesses to a 16-bit PC Card were defined as shown in Table 12-5. The SMC does not support Type 1 odd-byte access to 16-bit PC Cards.

Table 12-5. PCMCIA Legacy Usage

Access	nPC_CE2	nPC_CE1	A0	D15-D8	D7-D0
Type 1 - Odd-Byte Access	0	1	1	Odd-Byte	Z
Type 2 - Odd-Byte Access	1	0	1	Z	Odd-Byte

12.5 PC Card Memory Mapping

The address mapping for access to an 8- or 16-bit PC Card is shown in Table 12-6 and Table 12-7, respectively.

Note: It is up to the programmer to provide an even address for all attribute memory access operations (see PCMCIA Spec. 2.1), because the PCMCIA controller will generate the physical address as shown in Table 12-6 and Table 12-7, regardless of whether the least significant address bit is 0b1 or 0b0.

Note: In Table 12-6 and Table 12-7, bit 1 and bit 0 of the address each show a value of 0b1, 0b0, or 0bx. [25:2] refers to bit positions of the address, not address values.

Copyright 2007 Cirrus Logic

Table 12-6. Accesses to 8-Bit Attribute / Common / IO Memory

					Common / IO Memory Access		Attribute Memory Access	
Access	**Byte # In Word**	**PC_A[25:0]**	**nPC_CE2**	**nPC_CE1**	**D15-D8**	**D7-D0**	**D15-D8**	**D7-D0**
Word (4 transfers required)	0	[25:2],x,0	0	0	-	[7:0]	-	[7:0]
	1	[25:2],x,1	1	0	-	[15:8]	-	Invalid
	2	[25:2],x,0	0	0	-	[23:16]	-	[23:16]
	3	[25:2],x,1	1	0	-	[31:24]	-	Invalid
Lower Half-Word (2 transfers required)	0	[25:2],x,0	0	0	-	[7:0]	-	[7:0]
	1	[25:2],x,1	1	0	-	[15:8]	-	Invalid
Upper Half-Word (2 transfers required)	2	[25:2],x,0	0	0	-	[23:16]	-	[23:16]
	3	[25:2],1,1	1	0	-	[31:24]	-	Invalid
Byte	0	[25:2],x,0	0	0	-	[7:0]	-	[7:0]
Byte	1	[25:2],x,1	1	0	-	[15:8]	-	Invalid
Byte	2	[25:2],x,0	0	0	-	[23:16]	-	[23:16]
Byte	3	[25:2],x,1	1	0	-	[31:24]	-	Invalid

Table 12-7. Accesses to 16-Bit Attribute / Common / IO Memory

					Common / IO Memory Access		Attribute Memory Access	
Access	**Half-Word # IN Word**	**Processor Address Bus AD[25:0]**	**nPC_CE 2**	**nPC_CE 1**	**D15-D8**	**D7-D0**	**D15-D8**	**D7-D0**
Word (2 transfers required)	0	AD[25:2],x,x	0	0	[15:8]	[7:0]	Invalid	[7:0]
	1	AD[25:2],x,x	0	0	[31:24]	[23:16]	Invalid	[23:16]
Lower Half-Word	0	AD[25:2],x,x	0	0	[15:8]	[7:0]	Invalid	[7:0]
Upper Half-Word	1	AD[25:2],x,x	0	0	[31:24]	[23:16]	Invalid	[23:16]
Byte 0	0	AD[25:2],x,0	1	0	-	[7:0]	-	[7:0]
Byte 1	0	AD[25:2],0,1	1	0	[15:8]	-	Invalid	-
Byte 2	1	AD[25:2],x,0	1	0	-	[23:16]	-	[23:16]
Byte 3	1	AD[25:2],x,1	1	0	[31:24]	-	Invalid	-

Copyright 2007 Cirrus Logic

12

12.6 Registers

Table 12-8. Static Memory Controller (SMC) Register Map

Address	Name	Description
	"SMCBCR[7:0]"	(See individual bank configuration registers below)
0x8008_0000	"SMCBCR[7:0]"	Bank Configuration Register 0
0x8008_0004	"SMCBCR[7:0]"	Bank Configuration Register 1
0x8008_0008	"SMCBCR[7:0]"	Bank Configuration Register 2
0x8008_000C	"SMCBCR[7:0]"	Bank Configuration Register 3
0x8008_0010	Reserved	Reserved
0x8008_0014	Reserved	Reserved
0x8008_0018	"SMCBCR[7:0]"	Bank Configuration Register 6
0x8008_001C	"SMCBCR[7:0]"	Bank Configuration Register 7
0x8008_0020	"PCAttribute"	Attribute Space Register
0x8008_0024	"PCCommon"	Common Space Register
0x8008_0028	"PCIO"	I/O Space Register
0x8008_002C	Reserved	Reserved
0x8008_0030	Reserved	Reserved
0x8008_0034	Reserved	Reserved
0x8008_0038	Reserved	Reserved
0x8008_003C	Reserved	Reserved
0x8008_0040	"PCMCIACtrl"	Control Register

12.6.1 Bank Configuration Registers

SMCBCR[7:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD	EBIBRK DIS	MW		PME	WP	WPERR	RSVD								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WST2				BLE	WST1						RSVD	IDCY			

Address: **SMCBCR0: 0x8008_0000 - Read/Write**
SMCBCR1: 0x8008_0004 - Read/Write
SMCBCR2: 0x8008_0008 - Read/Write
SMCBCR3: 0x8008_000C - Read/Write
SMCBCR6: 0x8008_0018 - Read/Write
SMCBCR7: 0x8008_001C - Read/Write

Default: **0x2000_FBE0**

Definition: **SMC Bank Configuration registers**
These registers are used to specify the characteristics and timing for each of the memory banks, respectively.

Copyright 2007 Cirrus Logic

Bit Descriptions:

RSVD: Reserved - Unknown During Read

IDCY: Idle Cycle - Read/Write

The value written to this field specifies the memory data bus turnaround time between a Read access and a Write access. The turnaround time is specified by (IDCY + 1) HCLKs. For example, if IDCY = 0xA, the turnaround time is 10 + 1 = 11 cycles of HCLK.

WST1: Wait States1 - Read/Write

The value written to this field specifies the 'number of HCLK cycles, minus 1' that are inserted as wait cycles into the timing for:

- A single Read or Write access, or
- The first Read or Write access of a burst-of-four accesses.

The number of wait cycles is specified by (WST1 + 1) HCLKs. For example, if WST1 = 0x3, 3 + 1 = 4 cycles of HCLK are inserted into the access timing.

On reset, this field defaults to 0x1F (slowest access) to enable booting from ROM or FLASH memory device types.

RBLE: Read Byte Lane Enable - Read/Write

The value written to this bit specifies the output values on the DQMn[3:0] pins during a Read access:

0 - DQMn[3:0] pins are all driven HIGH during memory Reads (default at reset for bank 1-3,6,7)
1 - DQMn[3:0] pins are all driven LOW during memory Reads (default at reset for bank 0)

For memory Writes, this bit must written to '1'.

WST2: Wait States2 - Read/Write

The value in this field specifies the 'number of HCLK cycles, minus 1' that are inserted as wait cycles into the timing for each of the 2nd, 3rd, and 4th accesses of Read or Write burst-of-four accesses.

Copyright 2007 Cirrus Logic

The number of wait cycles for each of the 2nd, 3rd, and 4th accesses is specified by (WST2 + 1) HCLKs. For example, if WST2 = 0x4, 4 + 1 = 5 cycles of HCLK are inserted into the timing for each of the 2nd, 3rd, and 4th accesses.

On reset, this field defaults to 0x1F (slowest access) to enable booting from ROM or FLASH memory device types.

WPERR: Write Protect Error status flag - Read/Write

0 - No Error
1 - Write Protect Error

Writing a '1' to this bit will clear the Write Protect status error.

WP: Write Protect - Read/Write

The value written to this bit specifies that either Writes to the memory device are allowed to occur, or not occur:

0 - Yes (SRAM, FLASH)
1 - No (ROM, SRAM, FLASH)

PME: Page Mode (Burst-of-4) Enable - Read/Write

0 - Page Mode is disabled, non-burst accesses occur

1 - Page Mode is enabled. Page Mode provides fast burst-of-four accesses where the A[3] and A[4] address bits are internally incremented, '00' –> '01' –> '10' –> '11', to access four sequential words.

This bit is reset to '0'

MW: Memory Width - Read/Write

The value written to this field specifies the bus-width of the memory:

00 - 8-bit
01 - 16-bit
10 - 32-bit
11 - 32-bit

To support various bus-width memory devices for booting, the MW field of the "SMCBCR[7:0]" register can be automatically configured with the input values on the nCS7 and nCS6 pins, respectively. This takes place following a power-on reset, but only if the input values on these pins are: ASDO = '0', Boot[1:0] = '00', EEDAT = '1'. and EECLK = '0'.

Copyright 2007 Cirrus Logic

EBIBRKDIS: EBI Break Disable - Read/Write

The value written to this bit specifies the circumstances for when the SMC will release the external memory bus:

0 - The SMC releases the external memory bus at the end of each access to this memory bank
1 - The SMC releases the external memory bus after it has completed all pending accesses to this memory bank

12.6.2 PCMCIA Configuration Registers (EP9315 Processor Only)

The SMC has additional functionality to support a PC-card in Memory Bank 4. Memory Bank 4 has three registers to control wait-states and device width for attribute, common memory and IO address spaces; and a single PCMCIA control register to provide global control for the card.

PCAttribute

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WA	RSVD							AA							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				HA				PA							

Address: 0x8008_0020 - Read/Write

Default: 0x0000_0000

Definition: PC Card Attribute register

Bit Descriptions:

RSVD: Reserved - Unknown During Read

WA: Attribute Space Width - Read/Write

The value written to this bit specifies the bus-width of the Attribute space:

0 - 8-bit wide Attribute space
1 - 16-bit wide Attribute space

AA: Attribute Space Access time - Read/Write

The value written to this field specifies the minimum 'number of HCLK cycles, minus 1' that the data strobe, MCDAENn, is asserted during a Read or Write access.

Copyright 2007 Cirrus Logic

The data strobe assertion time is specified by (AA+1) HCLK cycles. For example, if AA = 0x10, the data strobe assertion time is 16 + 1 = 17 cycles of HCLK

HA: Attribute space Hold time - Read/Write

The value written to this field specifies the minimum 'number of HCLK cycles, minus 1' between de-asserting the data strobe, MCDAENn, and de-asserting the address strobe, MCADENn.

The Hold time is specified by (HA +1) HCLK cycles. For example, if HA = 0xC, the Hold time is 12 + 1 = 13 cycles of HCLK.

PA: Attribute space setup time - Read/Write

The value written to this field specifies the 'number of HCLK cycles, minus 1' that the address strobe, MCADENn, is set up before assertion of the data strobe, MCDAENn.

The Setup time is specified by (PA+1) HCLK cycles. For example, if PA = 0x25, the Setup time is 37 + 1 = 38 cycles of HCLK.

PCCommon

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WC	RSVD							AC							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				HC				PC							

Address: 0x8008_0024 - Read/Write

Default: 0x0000_0000

Definition: PC Card Common register

Bit Descriptions:

RSVD: Reserved - Unknown During Read

WC: Common Space Width - Read/Write

The value written to this bit specifies the bus-width of the Common space:

0 - 8-bit wide Common space
1 - 16-bit wide Common space

AC: Common Space Access time - Read/Write

Copyright 2007 Cirrus Logic

The value written to this field specifies the minimum 'number of HCLK cycles, minus 1' that the data strobe, MCDAENn, is asserted during a Read or Write access.

The data strobe assertion time is specified by (AC+1) HCLK cycles. For example, if AC = 0x10, the data strobe assertion time is 16 + 1 = 17 cycles of HCLK

HC: Common space Hold time - Read/Write

The value written to this field specifies the minimum 'number of HCLK cycles, minus 1' between de-asserting the data strobe, MCDAENn, and de-asserting the address strobe, MCADENn.

The Hold time is specified by (HC +1) HCLK cycles. For example, if HC = 0xC, the Hold time is 12 + 1 = 13 cycles of HCLK.

PC: Common space setup time - Read/Write

The value written to this field specifies the 'number of HCLK cycles, minus 1' that the address strobe, MCADENn, is set up before assertion of the data strobe, MCDAENn.

The Setup time is specified by (PC+1) HCLK cycles. For example, if PC = 0x25, the Setup time is 37 + 1 = 38 cycles of HCLK.

PCIO

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WI	RSVD							AI							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				HI				PI							

Address: 0x8008_0028 - Read/Write

Default: 0x0000_0000

Definition: PC Card IO register

Bit Descriptions:

RSVD: Reserved - Unknown During Read

WI: IO Space Width - Read/Write

The value written to this bit specifies the bus-width of the IO space:

Copyright 2007 Cirrus Logic

0 - 8-bit wide Common space
1 - 16-bit wide Common space

AI: IO Space Access time - Read/Write

The value written to this field specifies the minimum 'number of HCLK cycles, minus 1' that the data strobe, MCDAENn, is asserted during a Read or Write access.

The data strobe assertion time is specified by (AI+1) HCLK cycles. For example, if AI = 0x10, the data strobe assertion time is 16 + 1 = 17 cycles of HCLK

HI: IO space Hold time - Read/Write

The value written to this field specifies the minimum 'number of HCLK cycles, minus 1' between de-asserting the data strobe, MCDAENn, and de-asserting the address strobe, MCADENn.

The Hold time is specified by (HI +1) HCLK cycles. For example, if HI = 0xC, the Hold time is 12 + 1 = 13 cycles of HCLK.

PI: IO space setup time - Read/Write

The value written to this field specifies the 'number of HCLK cycles, minus 1' that the address strobe, MCADENn, is set up before assertion of the data strobe, MCDAENn.

The Setup time is specified by (PI+1) HCLK cycles. For example, if PI = 0x25, the Setup time is 37 + 1 = 38 cycles of HCLK.

PCMCIACtrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD											WEN	RSVD	PCRST	RSVD	PCEN

Address: 0x8008_0040 - Read/Write

Default: 0x0000_0000

Definition: PC Card Control register

Bit Descriptions:

RSVD: Reserved - Unknown During Read

Copyright 2007 Cirrus Logic

PCEN: PC Card Enable - Read/Write

Writing a "1" to this bit enables the PC Card interface.

PCRST: PC Card Reset - Read/Write

Writing a '1' to this bit clears the Configuration Option register in the card. This places the card into an unconfigured (memory only interface) state.

Writing a '0' to this bit allows normal PC Card operation.

WEN: External Wait Enable - Read/Write

Writing a '1' to this bit enables the MCWAIT input pin to be asserted by the card to insert wait cycles into the access timing.

Writing a '0' to this bit disables the MCWAIT input pin from being asserted by the card.

Copyright 2007 Cirrus Logic

12

Copyright 2007 Cirrus Logic

Chapter 13

SDRAM, SyncROM, and SyncFLASH Controller

13.1 Introduction

Note: In the EP9301 and 9302 processors, the common address/data bus is 16-bits wide and the SDRAM, SyncROM, and SyncFLASH synchronous memory controller supports 16-bit and 8-bit devices.

Note: In the EP9307, EP9312, and EP9315 processors, the common address/data bus is programmable to be either 16-bits or 32-bits wide and the SDRAM, SyncROM, and SyncFLASH synchronous memory controller supports 32-bit , 16-bit, and 8-bit devices.

The SDRAM controller provides a high speed memory interface to single-data-rate SDRAMs, Synchronous FLASH, and Synchronous ROMs.

The key features of the SDRAM controller are:

- Raster DMA input port for high-bandwidth display refreshing.
- Up to four synchronous memory banks that can be independently configured
- Special configuration bits for Synchronous ROM operation
- Ability to program Synchronous FLASH devices using write and erase commands
- Data is transferred between the controller and the synchronous memory device in quad-word bursts.
- Programmable for 16 or 32-bit data bus: EP9307, EP9312, and EP9315 processors only
- SDRAM contents are preserved when a “soft” reset is asserted
- Power saving synchronous memory clock enable

13.2 Booting from SyncROM or SyncFLASH

During power-on reset, if the values on the processor pins shown in Table CAUTION: select either a Synchronous ROM device or Synchronous FLASH device to be used for booting up the processor, a short configuration sequence is activated and completed before the processor is released from power-on reset. By default, Synchronous Memory Bank 3, controlled by device configuration register SDRAMDevCfg[3:0], is used for booting.

For a Synchronous ROM device, the configuration sequence writes RAS = 0x2 and CAS = 0x5 to the SDRAMDevCfg[3:0] register and writes RAS = 0x2, CAS = 0x5, and either Burst

Copyright 2007 Cirrus Logic

Length = 0x4 (32-bit wide memory bus) or Burst Length = 0x8 (16-bit wide memory bus) to the Mode register that is inside the SyncROM device.

For a Synchronous FLASH device, the configuration sequence writes RAS = 0x2 and CAS = 0x5 to the SDRAMDevCfg[3:0] register and writes WBM = 0x0, CAS = 0x3, and either Burst Length = 0x4 (32-bit wide memory bus) or Burst Length = 0x8 (16-bit wide memory bus) to the Configuration register that is inside the SyncFLASH device.

CAUTION: Do not attempt to configure the registers of other synchronous memory banks while booting from Synchronous Memory Bank 3. Attempting to do so may cause the system to lock-up. Rather, it is advised that the boot code copy the configuration code for other synchronous memory banks to some non-synchronous memory space, and then later configure the registers of the other synchronous memory banks from that space.

Table 13-1. Boot Device Selection

Boot modes	CSn7	CSn6	ASDO	EECLK
8-bit ROM	0	0	0	0
16-bit ROM	0	1	0	0
32-bit ROM	1	0	0	0
32-bit ROM	1	1	0	0
16-bit SFLASH (Initializes Command Register)	0	0	1	0
16-bit SROM (Initializes Mode Register)	0	1	1	0
32-bit SFLASH (Initializes Command Register	1	0	1	0
32-bit SROM (Initializes Mode Register)	1	1	1	0

The power-up sequence that is executed when the power-on reset becomes asserted is:

1. The SDCLKEN and DQM[3:0] pins are each externally pulled high so that they rise with the VDD and VDDQ power supplies.
2. Following power-up, the ARM Core is held in the reset state with HCLK running. The CKE bit in the Global configuration register, GlConfig, is written to '1' to enable HCLK to be output on the SDCLK pin. Initialize = '1', MRS = '1', and LCR = '0', shown in Table , are written to the GlConfig register to cause a NOP access to be issued. Continuous NOP accesses are issued for 200 µs.
3. Initialize = '0', MRS = '1', and LCR = '0' are written to the GlConfig register to enable access to the Mode register that is inside the synchronous memory device. Default settings are then written to the Mode register by reading the appropriate address, where the value of the address itself is the value of the default setting. For a Synchronous ROM device, the default settings are RAS = 0x2, CAS = 0x5, and either Burst Length = 0x4 (32-bit wide memory bus) or Burst Length = 0x8 (16-bit wide memory bus). For a Synchronous FLASH device, the default settings are WBM = 0x0, CAS = 0x3, and either Burst Length = 0x4 (32-bit wide memory bus) or Burst Length = 0x8 (16-bit wide memory bus).
4. Three SDCLK cycles after the Mode register is written with the appropriate default value, the memory portion of the synchronous memory device is ready for power-up with all of

Copyright 2007 Cirrus Logic

it's data outputs in the high impedance state. If power-on reset has become de-asserted, the ARM Core is released from the reset state.

13.3 Address Pin Usage

Each of the four synchronous memory domains can be fitted with a variety of device types, provided the total capacitance on any address/control/data line does not exceed the specified operating limit. Four pins, SDCSn[3:0], are used to as chip-selects (domain selects) for the four synchronous memory domains, where the configurations of the domains are specified by registers SDRAMDevCfg[3:0], SDRAMDevCfg[3:0], SDRAMDevCfg[3:0], and SDRAMDevCfg[3:0], respectively. For example, SDCSn[2] selects the 3rd of four synchronous memory domains and SDRAMDevCfg[3:0] specifies the configuration of that domain.

Address bits 31:28 are internally decoded to specify an address domain. Table 13-2 shows he values of address bits 31:28 that specify a synchronous memory domain.

Table 13-2. Address Decoding for Synchronous Memory Domains

Value of Address Bits 31:28	SDCSn[3:0]	Synchronous Memory Domain
0xF	3	4
0xE	2	3
0xD	1	2
0xC	0	1
0xB through 0x1	None	Used for other domains
0x0	3	Used during boot from SyncROM or SyncFLASH

Because of the row/column/bank architecture of synchronous memory devices, the mapping of these memories into the processor's memory space is not always obvious, typically because the memory inside a synchronous device does not appear to the processor to be continuous. For example, a 32-Mbyte SDRAM device may be visible as four 4-Mbyte banks. Table 13-3 shows address pin usage. In Table 13-3, external pins are identified as AD[15:0], internal address signals are identified as A[27:1]. The 2nd row of the table shows the address pins, AD[15:0], that may be connected to the synchronous memory device, depending on its address depth. The remaining rows show how the device's linear address space is mapped onto the address pins, AD[15:0]. For each memory device configuration, that is, 16- or 32-bit wide SDRAM or SROM or SFLASH, there is a Row and Bank, and Column, entry in the table that shows the internal linear address bits, A[27:1], that are presented on the external AD[15:0] pins for Row and Bank, and Column, accesses. The shallower the depth of the synchronous memory device, the fewer the number of most-significant address bits that are used for Row and Bank, and Column, addressing. By observing the number of rows and columns in a specific synchronous memory device (see the device's data sheet), the actual number of address bits used for addressing the device can be determined. Because some address bits are not used, the address map of the synchronous memory appears to be non-continuous. The SROMLL should be used when possible to reduce the number of "holes" in

Copyright 2007 Cirrus Logic

the synchronous memory map. Refer to Table 13-11 to compare the memory space with SROMLL=1 and SROMLL=0. bit can be used to reduce the number of memory segments and it is

Table 13-3. Synchronous Memory Address Decoding

Sync Device	Address Pins Muxing	Bank Address Pins		Address Pins													
		AD 15	AD 14	AD 13	AD 12	AD 11	AD 10	AD 9	AD 8	AD 7	AD 6	AD 5	AD 4	AD 3	AD 2	AD 1	AD 0
SDRAM 16 bit data	Row and Bank	A27	A26	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9
	Column	A27	A26	-	-	-	AP[1]	A25	A24	A8	A7	A6	A5	A4	A3	A2	A1
SDRAM 32 bit data	Row and Bank	A27	A26	A23	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10
	Column	A27	A26	-	-	-	AP[1]	A25	A24	A9	A8	A7	A6	A5	A4	A3	A2
SFLASH 2K Page Mode, 32 bit data	Row and Bank	A27	A26	A24	A23	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11
	Column	A27	A26	-	-	-	AP[1]	A25	A10	A9	A8	A7	A6	A5	A4	A3	A2
SROM 512, 32 bit data	Row and Bank	A27	A26	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9
	Column	A27	A26	-	-	-	-	A25	A24	A23	A8	A7	A6	A5	A4	A3	A2
SROM look alike, 16 bit data	Row and Bank	A22	A21	A27	A26	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9
	Column	A22	A21	-	-	-	AP[1]	A25	A24	A8	A7	A6	A5	A4	A3	A2	A1
SROM look alike, 32 bit data	Row and Bank	A23	A22	A27	A26	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10
	Column	A23	A22	-	-	-	AP[1]	A25	A24	A9	A8	A7	A6	A5	A4	A3	A2

1. "AP" means Auto Precharge -- see SDRAM device's data sheet

13.4 SDRAM Initialization

Following power on, each SDRAM device must be initialized before it can be used. Table 13-4 shows a general initialization sequence (refer to the SDRAM device's data sheet to ensure compatibility).

Table 13-4. General SDRAM Initialization Sequence

Step	Action	Reason
1	Wait 100 µs	To allow SDRAM power and clocks to stabilize

Copyright 2007 Cirrus Logic

Table 13-4. General SDRAM Initialization Sequence

Step	Action	Reason
2	Write a '1' or '0' to the External Bus Width bit in the appropriate "SDRAMDevCfg[3:0]" register	'1' specifies 16-bit memory bus width '0' specifies 32-bit memory bus width
3	Write Initialize = '0', MRS = '1', and LCR = '0' to the "GlConfig" register	To allow the Mode register inside the external SDRAM device to be accessed
4	Read from the external SDRAM's Mode register with Row and Bank address = 0x2 or 0x3 (see SDRAM data sheet)	0x2 -- Burst Length = 4 (32-bit wide memory bus) 0x3 -- Burst Length = 8 (16-bit wide memory bus)
5	Write Initialize = '1', MRS = '1', and LCR = '0' to the "GlConfig" register	To issue continuous NOP accesses
6	Wait 200 µs	SDRAM requirement
7	Write Initialize = '1', MRS = '0', and LCR = '0' to the "GlConfig" register	To issue a Pre-Charge All accesses
8	Write Refcnt = 0xB into the "RefrshTimr" register	To provide a refresh every 10 SDCLK cycles
9	Wait for at least 80 SDCLK cycles	To provide 8 refresh cycles to all SDRAMs in "SDRAMDevCfg[3:0]" space
10	Write the normal operating value to the Refcnt field in the "RefrshTimr" register	To establish normal refresh operation
11	Write Initialize = '0', MRS = '1', and LCR = '0' to the "GlConfig" register	To allow the Mode register inside the SDRAM device to be accessed
12	Perform a read from each SDRAM in the "SDRAMDevCfg[3:0]" space. The value of the address that is read defines the value that is written into the Mode register (see SDRAM device datasheet). The address value is dependent on the configuration of the memory system since the actual SDRAM address pins are mapped differently onto the processor's address pins for 16- and 32-bit wide memory systems. (This is the reason for step 2).	To set up the Mode register inside each SDRAM device
13	Write parameters corresponding to those programmed into the SDRAM devices Mode register into the corresponding fields of the "SDRAMDevCfg[3:0]" register. Write other fields in the "SDRAMDevCfg[3:0]"register as appropriate for the given SDRAM usage.	To initialize the SDRAM controller timing
14	Write Initialize = '0', MRS = '0', and LCR = '0' to the "GlConfig" register.	To start normal operation

Copyright 2007 Cirrus Logic

13

13.5 Programming Mode Register: SDRAM Or SyncROM Device

When setting up the Mode register that is inside an SDRAM or SyncROM device, or the Configuration register that is inside a SyncFLASH device, the command word that is placed on the address pins shown in Table 13-5 depends on whether a SROM, SDRAM, or SyncFlash is attached. Once Initialize = '0', MRS = '1', and LCR = '0' are written to the GlConfig register to enable access to the Mode register, the address of a subsequent Read operation is output on AD[12:0]. The internal address, A[23:0], is mapped to external address pins AD[12:0] as shown in Table 13-5.

In Table 13-5, AD[2:0] represents the Burst Length (BL). The Burst Length for 32-bit configurations must be set to four. The Burst Length for 16-bit configurations must be set to eight. See Table 13-8 for Burst Length values.

AD[3] specifies Burst Type (BT). A value of zero specifies Sequential, a value of one specifies Interleaved.

AD[6:4] specifies CAS Latency (CASL). Only values of two or three are supported. See Table 13-6 for CAS Latency values.

AD[8:7] specify Operation Mode (OM). This value must be zero for normal operation.

AD[9] specifies the Write Burst Mode (WBM). This value should be programmed to zero for devices that support burst, such as SDRAM. It should be set to zero for devices that do not support burst mode, such as SyncFlash or SyncROM.

AD[12:10] are reserved, but must be zero for normal operation.

Table 13-5. Mode Register Command Decoding for 32-bit Wide Memory Bus

Address	AD12	AD11	AD10	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
Mapped addr for default 32-bit wide	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10
SDRAM or SFLASH	RFU			Write Burst Mode	Operating Mode		CAS Latency			Burst Type	0	1	0
Example: SDRAM with WBM = 0, OM= 0, CASL = 3, BT = Sequential, BL = 4	0	0	0	0	0	0	0	1	1	0	0	1	0
SROM	RFU					RFU	RAS	CAS			Burst Type	0	1
Example: SROM RAS =2, CAS=2, Sequential, BL=4	0	0	0	0	0	0	1	1	0	0	0	0	1

Note: If using an external bus that is 16 bits wide then the address mapping must be shifted as indicated by Table 13-3 on page 13-4.

Note: For SDRAM, AD[2:0] specify burst length. For SROM, AD[1:0] specify burst length.

Copyright 2007 Cirrus Logic

Note: "RFU" means Reserved for Future Use.

Table 13-6, Table 13-7, and Table 13-8 show the bit field values for CASL, RAS, and Burst Length, respectively.

Table 13-6. Sync Memory CAS

CAS Value	SDRAM	SFLASH	SROM
000	Reserved	Reserved	Reserved
001	Reserved	1	2
010	2	2	3
011	3	3	4
100	Reserved	Reserved	5
101	Reserved	Reserved	6
110	Reserved	Reserved	7
111	Reserved	Reserved	8

Table 13-7. Sync Memory RAS, Burst Type, and Write Burst Length

Value	SDRAM	SFLASH	SROM
RAS = 0	Not applicable	Not applicable	1 clk
RAS = 1	Not applicable	Not applicable	2 clk
BT = 0	Sequential	Sequential	Sequential
BT = 1	Interleaved	Interleaved	Interleaved
WBM = 0	Use BL value	Use BL value	Use BL value
WBM = 1	Write Burst = 1	Write Burst = 1	Not applicable

Table 13-8. Burst Length

Burst Length	SDRAM	SFLASH	SROM
000	Reserved	1	Reserved
001	Reserved	2	4
010	4	4	8
011	8	8	Reserved
100	Reserved	Reserved	---
101	Reserved	Reserved	---
110	Reserved	Reserved	---
111	Reserved	Reserved	

When using a 32-bit wide external memory bus, the following Read addresses *must* be used to set up the specified parameters, where H can be 0x0, 0xC, 0xD, 0xE or 0xF as shown in Table 13-2:

- SDRAM default READ Address: 0xH000_C800 — sets WBM=0, TM=0, CAS=3, Sequential, BL=4
- SFLASH default READ Address: 0xH008_C800 — sets WBM=1, TM=0, CAS=3, Sequential, BL=4
- SROM default READ Address: 0xH001_8400 — sets RAS=2, CAS=5, Sequential, BL=4

Copyright 2007 Cirrus Logic

When using a 16-bit wide external memory bus, the following Read addresses *must* be used to set up the specified parameters, where H can be 0, C, D, E or F as shown in Table 13-2:

- SDRAM default READ Address: 0xH000_6600 — sets WBM=0, TM=0, CAS=3, Sequential, BL=8
- SFLASH default READ Address: 0xH004_6600 — sets WBM=1, TM=0, CAS=3, Sequential, BL=8
- SROM default READ Address: 0xH000_C400 — sets RAS=2, CAS=5, Sequential, BL=8

13.6 SDRAM Self Refresh

13.6.1 Entering Self Refresh Mode

When entering the Standby mode, the following actions are carried out by the Synchronous Memory controller before the processor is stopped:

1. Issue Precharge accesses to all active banks
2. Issue NOP commands
3. SDCLKEN output driven low
4. Issue AUTO REFRESH command
5. Enter SELF REFRESH Mode

13.6.2 Exiting Self Refresh Mode

When coming out of the Standby mode, the following actions are carried out by the synchronous memory controller before the processor is started:

1. Allow clock stabilization
2. SDCLKEN output driven high
3. Issue ten NOP accesses
4. Issue AUTO REFRESH accesses
5. Exit SELF REFRESH Mode

13.7 Programming Registers: SyncFLASH Device

The programmable registers that are inside a SyncFLASH memory device, can be programmed in a manner that is similar to programming the Mode register that is inside of an SDRAM or SyncROM memory device.

The process of programming the SyncFLASH registers begins by writing WBM = '1' to the appropriate SDRAMDevCfg register to specify that burst-of-four reads and burst-of-one writes will be used to access the device. Then, write LCR = '1' to the GlConfig register. Doing so causes the value of a subsequent read address to be used as the data value that is written

Copyright 2007 Cirrus Logic

to the SyncFLASH register and the associated value on the data pins specifies which SyncFLASH register is written. Actually, the value on the data pins specifies a command to the SyncFLASH device such as Write Configuration Register, Lock Block, Block Erase; and the associated value on the address pins specifies either a value that is written to a register or a address location inside the SyncFLASH device.

Synchronous FLASH devices:

- Use the same combination of the CS, RAS, CAS, and WE signals which would normally place an SDRAM device into Auto-Refresh mode
- Cannot be written in bursts, but only one word at a time. Hence the requirement to write WBM = '1' to the appropriate SDRAMDevCfg register. When WBM = '1', no Auto Refresh cycle will occur in the associated synchronous memory domain because the synchronous memory controller will assume that a Synchronous FLASH device is attached.
- Require 100 µs of initialization time after a low-to-high transition occurs on its write protect input pin
- Can be set up by either programming the Synchronous FLASH Configuration register before releasing the processor from reset or by using the contents of it's NonVolatileMODE register (which must have been previously programmed).

13.8 External Synchronous Memory System

The synchronous memory system is decoded from the ARM Core's physical memory map into four independent address domains, each having an address range of 256 Mbytes (64 Mwords). All of the memory devices that are attached to a given domain must be of the same type, but the other domains may use different memory device types and associated timing characteristics.

Since all memory devices, synchronous or static, share a common external memory bus, the total number of devices is limited by the maximum allowable bus capacitance.

13.8.1 Chip Select SDCSN[3:0] Decoding

Each of the four address domains within synchronous memory space have an associated chip select signal that is output on one of the SDCSn[3:0] pins as shown in Table 13-9. These signals are decoded from address bits A31:A28.

The latched value of ASDO determines how SDCSn3 is mapped into synchronous memory space. If the latched value of ASDO=1 then SDCSn3 is mapped to 0x0000_0000 otherwise it is mapped to 0xF000_0000.

Table 13-9. Chip Select Decoding

Boot Option (ASDO)	A31	A30	A29	A28	Chip select
1	0	0	0	0	nSDCS3

Copyright 2007 Cirrus Logic

Table 13-9. Chip Select Decoding

Boot Option (ASDO)	A31	A30	A29	A28	Chip select
0	1	1	1	1	nSDCS3
X	1	1	1	0	nSDCS2
X	1	1	0	1	nSDCS1
X	1	1	0	0	nSDCS0

13

13.8.2 Address/Data/Control Required by Memory System

An independent device configuration register, "SDRAMDevCfg[3:0]", "SDRAMDevCfg[3:0]", "SDRAMDevCfg[3:0]", and "SDRAMDevCfg[3:0]", is provided for each of the four synchronous memory domains. Each domain can be configured for either an SDRAM, SyncROM, or SyncFLASH device type. Only one device type can be configured per domain. However, different domains can be configured for different device types.

Each of the four synchronous memory domains can be configured to be either 16- or 32-bits wide and each will support 32-bit (word), 16-bit (half-word), and 8-bit (byte) accesses to or from the synchronous memory device. If the external memory bus is 16-bits wide, two external bus accesses are automatically made to Read or Write a 32-bit word. This is why a burst-of-eight accesses is used to Read or Write devices that are attached to a 16-bit bus while only a burst-of-four accesses is used to Read or Write devices that are attached to a 32-bit bus.

When writing to external memory, byte lane enable signals are output on the nDQM[3:0] pins, where the DQMn0 pin controls the least-significant byte lane, the DQMn1 pin controls the next to least-significant byte lane, the DQMn2 pin controls the next to most- significant byte lane, and the DQMn3 pin controls the most-significant byte lane. The memory device uses the byte lane enable signals on the DQMn pins to determine which byte lane data it should accept during a Write operation. For example, if a 32-bit word is to be written to a memory device on a 32-bit memory bus, DQMn[3:0] = '0000' is output to alert the memory device that it should accept Write data from all four byte lanes. However, if an 8-bit byte is to be written to the next to least-significant byte of a memory device on a 32-bit bus, DQMn[3:0] = '1101' is output to alert the memory device that it should accept Write data from only the next to least-significant byte lane and reject Write data from the other byte lanes. As another example, if a 32-bit word is to be written to a memory device on a 16-bit data bus, two 16-bit writes are automatically performed to the memory device. For each 16-bit Write, DQM[1:0] = '00' and DQM[3:2] are not used.

Table 13-10 shows a memory addressing example for a 256 Mbit synchronous memory device with 13-row x 9-column x 2-bank addressing attached to a 16-bit memory bus. Note

Copyright 2007 Cirrus Logic

that AD23 is not used (needed) in either the row or column address, and this demonstrates why the memory map for synchronous memory devices may be non-continuous.

Table 13-10. Memory Addressing Example

Muxing Scheme		B1	B0	AD 13	AD 12	AD 11	AD 10	AD 9	AD 8	AD 7	AD 6	AD 5	AD 4	AD 3	AD 2	AD 1	AD 0
16-Bits Data	ROW / BANK	A27	A26	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9
	COLUMN	-	-	-	-	-	AP	A25	A24	A8	A7	A6	A5	A4	A3	A2	A1

Table 13-11 shows the continuous address ranges used by a variety of different synchronous memory configurations. Note that in the "Continuous Address Range Per Segment" column, the value N can be 0x0, 0xC, 0xD, 0xE or 0xF as shown in Table 13-12.

Copyright 2007 Cirrus Logic

Table 13-11. EP93xx SDRAM Address Ranges (16-Bit Wide Data Systems)

Organization	Device Size, Type System	Address Matrix	Total Bank Size	SROMLL = 0		SROMLL = 1	
				Continuous Address Range (see Note)	Size of Segment	Continuous Address Range (see Note)	Size of Segment
16-Bit Wide Data Systems	64 Mbit (16-bit wide device)	12 x 8 x 4 banks	8 Mbytes	0xN000_0000 - 0xN01F_FFFF 0xN400_0000 - 0xN41F_FFFF 0xN800_0000 - 0xN81F_FFFF 0xNC00_0000 - 0xNC1F_FFFF	2 Mbytes	0xN000_0000 - 0xN07F_FFFF	8 Mbytes
	128 Mbit (16-bit wide device)	12 x 9 x 4 banks	16 Mbytes	0xN000_0000 - 0xN01F_FFFF 0xN100_0000 - 0xN11F_FFFF 0xN400_0000 - 0xN41F_FFFF 0xN500_0000 - 0xN51F_FFFF 0xN800_0000 - 0xN81F_FFFF 0xN900_0000 - 0xN91F_FFFF 0xNC00_0000 - 0xNC1F_FFFF 0xND00_0000 - 0xND1F_FFFF	2 Mbytes	0xN000_0000 - 0xN07F_FFFF 0xN100_0000 - 0xN17F_FFFF	8 Mbytes
	256 Mbit (16-bit wide device)	13 x 9 x 4 banks	32 Mbytes	0xN000_0000 - 0xN03F_FFFF 0xN100_0000 - 0xN13F_FFFF 0xN400_0000 - 0xN43F_FFFF 0xN500_0000 - 0xN53F_FFFF 0xN800_0000 - 0xN83F_FFFF 0xN900_0000 - 0xN93F_FFFF 0xNC00_0000 - 0xNC3F_FFFF 0xND00_0000 - 0xND3F_FFFF	4 Mbytes	0xN000_0000 - 0xN07F_FFFF 0xN100_0000 - 0xN17F_FFFF 0xN400_0000 - 0xN47F_FFFF 0xN500_0000 - 0xN57F_FFFF	8 Mbytes
				0xN000_0000 - 0xN03F_FFFF 0xN100_0000 - 0xN13F_FFFF 0xN200_0000 - 0xN23F_FFFF			

Copyright 2007 Cirrus Logic

Table 13-11. EP93xx SDRAM Address Ranges (16-Bit Wide Data Systems) (Continued)

Organization	Device Size, Type System	Address Matrix	Total Bank Size	SROMLL = 0		SROMLL = 1	
				Continuous Address Range (see Note)	Size of Segment	Continuous Address Range (see Note)	Size of Segment
16-Bit Wide Data Systems (Continued)	512 Mbit (16-bit wide device)	13 x 10 x 4 banks	64 Mbytes	0xN300_0000 - 0xN33F_FFFF 0xN400_0000 - 0xN43F_FFFF 0xN500_0000 - 0xN53F_FFFF 0xN600_0000 - 0xN63F_FFFF 0xN700_0000 - 0xN73F_FFFF 0xN800_0000 - 0xN83F_FFFF 0xN900_0000 - 0xN93F_FFFF 0xNA00_0000 - 0xNA3F_FFFF 0xNB00_0000 - 0xNB3F_FFFF 0xNC00_0000 - 0xNC3F_FFFF 0xND00_0000 - 0xND3F_FFFF 0xNE00_0000 - 0xNE3F_FFFF 0xNF00_0000 - 0xNF3F_FFFF	4 Mbytes	0xN000_0000 - 0xN07F_FFFF 0xN100_0000 - 0xN17F_FFFF 0xN200_0000 - 0xN27F_FFFF 0xN300_0000 - 0xN37F_FFFF 0xN400_0000 - 0xN47F_FFFF 0xN500_0000 - 0xN57F_FFFF 0xN600_0000 - 0xN67F_FFFF 0xN700_0000 - 0xN77F_FFFF	8 Mbytes

Table 13-11. EP93xx SDRAM Address Ranges (16-Bit Wide Data Systems) (Continued)

Organization	Device Size, Type System	Address Matrix	Total Bank Size	SROMLL = 0		SROMLL = 1	
				Continuous Address Range (see Note)	Size of Segment	Continuous Address Range (see Note)	Size of Segment
32-Bit Wide Data Systems	64 Mbit (32-bit wide device)	12 x 8 x 2 banks	8 Mbytes	0xN000_0000 - 0xN03F_FFFF 0xN400_0000 - 0xN43F_FFFF	4 Mbytes	0xN000_0000 - 0xN07F_FFFF	8 Mbytes
	64 Mbit (32-bit wide device)	11 x 8 x 4 banks	8 Mbytes	0xN000_0000 - 0xN01F_FFFF 0xN400_0000 - 0xN41F_FFFF 0xN800_0000 - 0xN81F_FFFF 0xNC00_0000 - 0xNC1F_FFFF	2 Mbytes	0xN000_0000 - 0xN01F_FFFF 0xN040_0000 - 0xN05F_FFFF 0xN080_0000 - 0xN09F_FFFF 0xN0C0_0000 - 0xN0DF_FFFF	2 Mbytes
	64 Mbit (2 x 16-bit wide device)	12 x 8 x 4 banks	16 Mbytes	0xN000_0000 - 0xN03F_FFFF 0xN400_0000 - 0xN43F_FFFF 0xN800_0000 - 0xN83F_FFFF 0xNC00_0000 - 0xNC3F_FFFF	4 Mbytes	0xN000_0000 - 0xN0FF_FFFF	16 Mbytes
	128 Mbit (32-bit wide device)	12 x 8 x 4 banks	16 Mbytes	0xN000_0000 - 0xN03F_FFFF 0xN400_0000 - 0xN43F_FFFF 0xN800_0000 - 0xN83F_FFFF 0xNC00_0000 - 0xNC3F_FFFF	4 Mbytes	0xN000_0000 - 0xN0FF_FFFF	16 Mbytes
	128 Mbit (2 x 16-bit wide device)	12 x 9 x 4 banks	32 Mbytes	0xN000_0000 - 0xN03F_FFFF 0xN100_0000 - 0xN13F_FFFF 0xN400_0000 - 0xN43F_FFFF 0xN500_0000 - 0xN53F_FFFF 0xN800_0000 - 0xN83F_FFFF 0xN900_0000 - 0xN93F_FFFF 0xNC00_0000 - 0xNC3F_FFFF 0xND00_0000 - 0xND3F_FFFF	4 Mbytes	0xN000_0000 - 0xN1FF_FFFF	32 Mbytes
				0xN000_0000 - 0xN07F_FFFF			

Copyright 2007 Cirrus Logic

Table 13-11. EP93xx SDRAM Address Ranges (16-Bit Wide Data Systems) (Continued)

Organization	Device Size, Type System	Address Matrix	Total Bank Size	SROMLL = 0		SROMLL = 1	
				Continuous Address Range (see Note)	Size of Segment	Continuous Address Range (see Note)	Size of Segment
32-Bit Wide Data Systems (Continued)	256 Mbit (32-bit wide device)	13 x 8 x 4 banks	32 Mbytes	0xN400_0000 - 0xN47F_FFFF 0xN800_0000 - 0xN87F_FFFF 0xNC00_0000 - 0xNC7F_FFFF	8 Mbytes	0xN000_0000 - 0xN0FF_FFFF 0xN400_0000 - 0xN4FF_FFFF	16 Mbytes
	256 Mbit (2 x 16-bit wide device)	13 x 9 x 4 banks	64 Mbytes	0xN000_0000 - 0xN07F_FFFF 0xN100_0000 - 0xN17F_FFFF 0xN400_0000 - 0xN47F_FFFF 0xN500_0000 - 0xN57F_FFFF 0xN800_0000 - 0xN87F_FFFF 0xN900_0000 - 0xN97F_FFFF 0xNC00_0000 - 0xNC7F_FFFF 0xND00_0000 - 0xND7F_FFFF	8 Mbytes	0xN000_0000 - 0xN1FF_FFFF 0xN400_0000 - 0xN5FF_FFFF	32 Mbytes
				0xN000_0000 - 0xN07F_FFFF 0xN100_0000 - 0xN17F_FFFF 0xN200_0000 - 0xN27F_FFFF			

Copyright 2007 Cirrus Logic

Table 13-11. EP93xx SDRAM Address Ranges (16-Bit Wide Data Systems) (Continued)

Organization	Device Size, Type System	Address Matrix	Total Bank Size	SROMLL = 0		SROMLL = 1	
				Continuous Address Range (see Note)	Size of Segment	Continuous Address Range (see Note)	Size of Segment
32-Bit Wide Data Systems (Continued)	512 Mbit (2 x 16-bit wide device)	13 x 10 x 4 banks	128 Mbytes	0xN300_0000 - 0xN37F_FFFF 0xN400_0000 - 0xN47F_FFFF 0xN500_0000 - 0xN57F_FFFF 0xN600_0000 - 0xN67F_FFFF 0xN700_0000 - 0xN77F_FFFF 0xN800_0000 - 0xN87F_FFFF 0xN900_0000 - 0xN97F_FFFF 0xNA00_0000 - 0xNA7F_FFFF 0xNB00_0000 - 0xNB7F_FFFF 0xNC00_0000 - 0xNC7F_FFFF 0xND00_0000 - 0xND7F_FFFF 0xNE00_0000 - 0xNE7F_FFFF 0xNF00_0000 - 0xNF7F_FFFF	8 Mbytes	0xN000_0000 - 0xN7FF_FFFF	128 Mbytes

Note: , the letter "N" represents four additional address bits used for chip select. See Table 13-12.

Copyright 2007 Cirrus Logic

Table 13-12. Address Bits Used for Chip Select

Boot Option (ASDO)	A31	A30	A29	A28	Chip select
1	0	0	0	0	nSDCS3
0	1	1	1	1	nSDCS3
X	1	1	1	0	nSDCS2
X	1	1	0	1	nSDCS1
X	1	1	0	0	nSDCS0

13.9 Registers

The Synchronous Memory controller has seven registers as shown in Table 13-13. The Configuration registers allow software to specify the operating parameters of the Synchronous Memory controller according to the memory device types being used. The Refresh Timer register allows software to specify the time period between successive synchronous memory refresh commands. The Boot Status allows software to determine the state of the boot configuration pins.

Table 13-13. Synchronous Memory Controller Registers

Address	Name	Description
0x8006_0000	Reserved	
0x8006_0004	"GlConfig"	Global Configuration
0x8006_0008	"RefrshTimr"	Refresh Timer
0x8006_000C	"BootSts"	Boot Configuration Pins Status
	"SDRAMDevCfg[3:0]" (See Below)	
0x8006_0010	SDRAMDevCfg[3:0]	Synchronous Device Configuration 0
0x8006_0014	SDRAMDevCfg[3:0]	Synchronous Device Configuration 1
0x8006_0018	SDRAMDevCfg[3:0]	Synchronous Device Configuration 2
0x8006_001C	SDRAMDevCfg[3:0]	Synchronous Device Configuration 3

Copyright 2007 Cirrus Logic

Register Descriptions

GlConfig

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CKE	Clk Shutdown	RSVD													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								ReArb En	LCR	SMEM Bust	RSVD			MRS	Initialize

Address: 0x8006_0004 - Read/Write

Default: 0x0000_0000

Definition:

The Global configuration register contains general control and status bits. The least-significant two bits, MRS and Initialize, are used in combination as shown in Table to allow access to otherwise unavailable synchronous memory commands that are required during memory initialization. The Synchronous Memory Busy Status bit, SMEMBust, provides the state of the Synchronous Memory controller, and it can be monitored to determine when a change of device configuration has taken effect.

Bit Descriptions:

RSVD: Reserved - Unknown During Read

CKE: Synchronous memory Clock Enable - Read/Write

Writing a value to this bit specifies if the enable signal that is output on the SDCLKEN is asserted, or not:

0 - SDCLKEN is de-asserted to save power only when there is no current access to any synchronous memory device
1 - SDCLKEN is continuously asserted (especially useful when booting from SyncROM or SyncFLASH device types)

ClkShutdown: Synchronous memory Clock Shutdown - Read/Write

Writing a value to this bit specifies if the HCLK output on the SDCLK pin is free-running or gated off:

0 - SDCLK is free-running
1 - SDCLK is gated off only when there is no current access to any synchronous memory device

Copyright 2007 Cirrus Logic

The CKE bit must be written to '0' before the ClkShutdown bit is written to '1'.

ReArbEn: Re-arbitration controller Enable - Read/Write

Writing a '1' to this bit allows the SDRAM Arbiter to stop the current burst accesses to the external synchronous memory, allow burst accesses from another requester to begin, and later resume the stopped burst accesses. This can suspend burst accesses from the Raster engine long enough to deprive the display from being adequately refreshed, and thereby cause undesired affects to appear on the display. So, by default, this bit is '0'.

Writing a '0' to this bit specifies that the SDRAM Arbiter must wait for current burst accesses to complete before it allows burst accesses from another requester to begin.

LCR: Load FLASH Command Register - Read/Write

When Initialize = '0' and MRS = '1', writing a '1' to this bit allows commands to be issued to the Synchronous FLASH device as described in "Programming Registers: SyncFLASH Device" on page 13-8:

0 - See Table 13-10
1 - See Table 13-10

SMEMBust: Synchronous Memory Busy Status - Read/Write

This status bit shows that the Synchronous Memory controller is either busy or idle:

0 - Idle
1 - Busy

When this bit is a '1', writing a '1' to it will clear it to '0'.

MRS: Synchronous Memory Mode Register - Read/Write

When Initialize = '0' and LCR = '0', writing a '1' to this bit allows setup commands to be written to the Mode register that is inside a synchronous memory device. When this bit is written to a '1', subsequent Read accesses to the synchronous device cause commands on the AD[13:0] pins to be written to the Mode register.

0 - See Table 13-14
1 - See Table 13-14

Initialize: Initialize bit - Read/Write

Copyright 2007 Cirrus Logic

Writing a '1' to this bit, in combination with the values of the MRS and LCR bits, cause the Synchronous Memory controller to issue either NOP or PreALL accesses to SDRAM devices as shown in Table 13-4.

0 - See Table 13-14
1 - See Table 13-14

13

Table 13-14. Synchronous Memory Command Encoding

Initialize	MRS	LCR	Synchronous Memory Command
1	1	0	Issue NOP to Synchronous Memory
1	0	0	Issue PreALL (Pre-charge All) to SDRAM
0	1	0	Enable access to Synchronous Memory device mode register
0	1	1	Issue command to Synchronous FLASH Memory devices
0	0	1	UNDEFINED. Do not use.
1	0	1	UNDEFINED. Do not use.
1	1	1	UNDEFINED. Do not use.
0	0	0	Normal operation

RefrshTimr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Refcnt															

Address: 0x8006_0008 - Read/Write

Default: 0x0000_0080

Copyright 2007 Cirrus Logic

Definition:

The Refresh Timer register is used to specify the period between refresh cycles.

Bit Descriptions:

RSVD: Reserved. - Unknown During Read

Refcnt: Refresh Count - Read/Write

The value written to this field specifies, in multiples of the period of HCLK, the time period between refresh cycles. For example, if the period of HCLK is 20 ns, this field should be written to 0x320 (decimal 800) to generate a 16 ms refresh period. On reset, this field defaults to 0x0080 (decimal 128) to generate a 2.56 ms refresh period, but it must be written during the SDRAM initialization routine to the appropriate value for the SDRAM devices. If this field is written to 0x0000, no refresh cycles are issued.

BootSts

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD													Latched ASDO	Width	

Address: 0x8006_000C - Read Only

Default: 0x0000_0000

Definition:

When power on reset is asserted, the values of the boot mode option pins shown in Table 13-1 are latched. The Boot Status register reflects those latched values. This register can be read to determine which memory configuration was used during the boot process.

Bit Descriptions:

RSVD: Reserved - Unknown During Read

ASDO: Latched ASDO pin value - Read Only

Boot Media:

1 - SyncROM or SyncFLASH
0 - Asynchronous ROM

Width: Boot memory bus Width - Read Only

Copyright 2007 Cirrus Logic

13

Latched nCS[7:6] pins values:

Asynchronous (ASDO = '0')

11 - 32-bit
10 - 32-bit
01 - 16-bit
00 - 8-bit

Synchronous (ASDO = '1')

11 - 32-bit SROM (RAS=2, CAS=5, BL=4)
10 - 32-bit SFLASH (WBM=1, CAS=3, BL=4)
01 - 16-bit SROM (RAS=2, CAS=5, BL=8)
00 - 16-bit SFLASH (WBM=1, CAS=3, BL=4)

Note: 8-bit wide bus is not supported for SyncROM or SyncFLASH.

If booting from Asynchronous ROM, asynchronous memory bank zero (nCS0) is mapped to address 0x0000_0000. If booting from SyncROM or SyncFLASH, Synchronous Memory Domain 3 (nSDCS3) is re-mapped to address 0x0000_0000. This re-mapping of nSDCS3 does not change until after the boot process is completed and the processor is reset (not power-on reset). At that time, nSDCS3 is mapped back to address 0xF000_0000, the beginning address of Synchronous Memory Domain 3.

SDRAMDevCfg[3:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD							Auto Precharge	RSVD		RasToCas		WBM	CasLat		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								SFConfig Addr	2K PAGE	SROMLL	SROM512	Bank Count	External Bus Width	RSVD	

Address: **SDRAMDevCfg0: 0x8006_0010 - Read/Write**
SDRAMDevCfg1: 0x8006_0014 - Read/Write
SDRAMDevCfg2: 0x8006_0018 - Read/Write
SDRAMDevCfg3: 0x8006_001C - Read/Write

Default: **0x0122_0008**

Copyright 2007 Cirrus Logic

Definition:

The four device configuration registers, SDRAMDevCfg[3:0], specify the characteristics of the external synchronous memory device types that are attached to each of the four Synchronous Memory Domains. Only one device type, SDRAM, SyncROM, or SyncFLASH, can be attached to a given domain, but the other domains can have different device types attached.

For correct operation, the values written to these configuration registers must correspond with the values that are programmed into the Mode register that is inside an SDRAM or SyncROM device.

Changes written to these configuration registers are applied only when the Synchronous Memory controller is idle or when it becomes idle. This assures that the Synchronous Memory controller remains synchronized to the state of the respective synchronous memory device. To assure correct programming results, these registers should only be written when interrupts, and DMA operations, are disabled.

Bit Descriptions:

RSVD: Reserved - Unknown During Read

AutoPrecharge: SDRAM Automatic Precharge - Read/Write

During SDRAM initialization, the value written to this bit specifies if the Synchronous Memory controller should issue an automatic precharge access to the SDRAM device, or not:

0 - No automatic precharge access
1 - Issue automatic precharge access

RasToCas: Synchronous memory RAS-to-CAS latency - Read/Write

The value written to this field specifies the RAS-to-CAS latency that the Synchronous Memory controller uses for Read or Write accesses to SDRAM or SyncROM devices:

00 - Reserved
01 - Reserved
10 - RAS Latency = 2 (also default value used when booting from a SyncROM device)
11 - RAS Latency = 3

When performing a Write access, the Synchronous Memory controller automatically adds one SDCLK cycle to the RasToCas value. When performing a Read access, the Synchronous Memory controller uses the RasToCas value as it is.

WBM: Write Burst Mode - Read/Write

Copyright 2007 Cirrus Logic

13

When writing to a SyncFLASH device, only single writes (burst-of-one) are allowed. The value that is written to this bit specifies that a burst length of either one or four will be used for Write accesses:

0 - Burst-of-four accesses for both Reads and Writes
1 - Burst-of-one accesses for Writes (SyncFLASH support) and burst-of-four accesses for Reads

When WBM = '1', the Synchronous Memory controller will not issue refresh cycles to this domain.

A single word write occurs when the ARM assembly instruction, 'str', is executed. Writing WBM = '1' will not prevent burst-of-four writes from occurring when the ARM assembly instruction, 'stm', is executed. So, **only use ARM assembly "str" instructions for Write accesses to SyncFLASH devices.**

CasLat: Synchronous memory CAS Latency - Read/Write

The value written to this field specifies the CAS latency that the Synchronous Memory controller uses for Read or Write accesses to SDRAM or SyncROM devices:

000 - Reserved
001 - CAS Latency = 2
010 - CAS Latency = 3 (also normal default)
011 - CAS Latency =4
100 - CAS Latency =5 (also default when booting from a SyncROM device)
101 - CAS Latency =6
110 - CAS Latency = 7
111 - CAS Latency =8

SFConfigAddr: SyncFLASH Configuration register read - Read/Write

The value written to this bit specifies either normal operation or that the Synchronous Memory controller is caused to perform a Read access to the Configuration register that is inside a SyncFLASH device:

0 - Normal operation
1 - Read SyncFLASH Configuration register

The AutoPrecharge bit must be '0' before the SFConfigAddr bit is written to '1'.

2KPAGE: Synchronous memory 2K byte Page - Read/Write

Copyright 2007 Cirrus Logic

The value written to this bit specifies a synchronous memory page size of 2 KBytes, or not:

0 - Page size is not 2 KByte
1 - Page size is 2 KByte

Only one of the SROM512, SROMLL, and 2KPAGE bits can be '1' at any time. With the exception of SROMLL, these bits always operate in 32-bit memory bus width mode regardless of the setting of External Bus Width bit.

SROMLL: SROM Look-a-Like - Read/Write

The value written to this bit specifies if a SyncFLASH device is operated in a manner that mimics a SycnROM device, or not:

0 - SyncFLASH device does not mimic SycnROM device
1 - SyncFLASH device mimics SycnROM device (16-bit wide memory bus only as specified by External Bus Width = '0')

If this bit is written to '1', the signals on the BA0 and BA1 pins are exchanged with the signals on the AD12 and AD13 pins, respectively.

Only one of the SROM512, SROMLL, and 2KPAGE bits can be '1' at any time. With the exception of SROMLL, these bits always operate in 32-bit memory bus width mode regardless of the setting of External Bus Width bit.

SROM512: Synchronous ROM 512 byte page - Read/Write

The value written to this bit specifies if a SyncROM device has a page size of 512 bytes, or not:

0 - Page size is not 512 bytes
1 - Page size is 512 bytes

Only one of the SROM512, SROMLL, and 2KPAGE bits can be '1' at any time. With the exception of SROMLL, these bits always operate in 32-bit memory bus width mode regardless of the setting of External Bus Width bit.

BankCount: Bank Count - Read/Write

The value written to this bit specifies the number of banks that are inside an SDRAM device:

1 - Four banks
0 - Two banks

External Bus Width:External Bus Width - Read/Write

Copyright 2007 Cirrus Logic

The value written to this bit specifies the width of the memory bus:

0 - Width is 32-bits
1 - Width is 16-bits

Copyright 2007 Cirrus Logic

Chapter 14

UART1 With HDLC and Modem Control Signals

14.1 Introduction

UART1 is the collection of a UART block along with a block to support a 9 pin modem interface and a block to support synchronous and asynchronous HDLC protocol support for full duplex transmit and receive. The following sections address each of these blocks.

14.2 UART Overview

Transmit and Receive data transfers through UART1 can either be managed by the DMA, interrupt driven, or CPU polled operations. A loopback control bit is available to enable system testing by routing the transmit data stream into the receiver.

The UART performs:

- Serial-to-parallel conversion on data received from a peripheral device.
- Parallel-to-serial conversion on data transmitted to the peripheral device.

The CPU reads and writes data and control/status information via the AMBA APB interface. The transmit and receive paths are buffered with internal FIFO memories allowing up to 16 bytes to be stored independently in both transmit and receive modes.

The UART:

- Includes a programmable baud rate generator which generates a common transmit and receive internal clock from the UART internal reference clock input, UARTCLK.
- Offers similar functionality to the industry-standard 16C550 UART device.
- Supports baud rates of up to 115.2 Kbps and beyond, subject to UARTCLK reference clock frequency.

The UART operation and baud rate values are controlled by the line control register (UART1LinCtrl).

The UART can generate:

- Four individually-maskable interrupts from the receive, transmit and modem status logic blocks.
- A single combined interrupt so that the output is asserted if any of the individual interrupts are asserted and unmasked.

If a framing, parity or break error occurs during reception, the appropriate error bit is set, and is stored in the FIFO. If an overrun condition occurs, the overrun register bit is set immediately and FIFO data is prevented from being overwritten.

Copyright 2007 Cirrus Logic

The FIFOs can be programmed to be 1 byte deep providing a conventional double-buffered UART interface.

The modem status input signals Clear To Send (**CTS**), Data Carrier Detect (**DCD**) and Data Set Ready (**DSR**) are supported. The additional modem status input Ring Indicator (**RI**) is not supported. Output modem control lines, such as Request To Send (**RTS**) and Data Terminal Ready (**DTR**), are not explicitly supported. Note that the separate modem block described later in this chapter does provide support for **RI**, **RTS**, and **DTR**.

14.2.1 UART Functional Description

A block diagram of the UART is shown in Figure 14-1.

14.2.1.1 AMBA APB Interface

The AMBA APB interface generates read and write decodes for accesses to status and control registers and transmit and receive FIFO memories.

The AMBA APB is a local secondary bus which provides a low-power extension to the higher bandwidth Advanced High-performance Bus (AHB) within the AMBA system hierarchy. The AMBA APB groups narrow-bus peripherals to avoid loading the system bus and provides an interface using memory-mapped registers which are accessed under program control.

14.2.1.2 DMA Block

The DMA interface passes data between the UART FIFOs and an external DMA engine as an alternative to AMBA APB accesses. (See Chapter 10,, "DMA Controller" on page 10-1 for additional details.) It may be configured to automatically move characters from the DMA engine to the transmit FIFO and from the receive FIFO to the DMA engine. The DMA engine may also indicate certain error conditions in the receive data to the DMA engine. Note that the DMA interface only supports 8-bit accesses to the FIFOs; status information in the receive FIFO is not passed to the DMA engine.

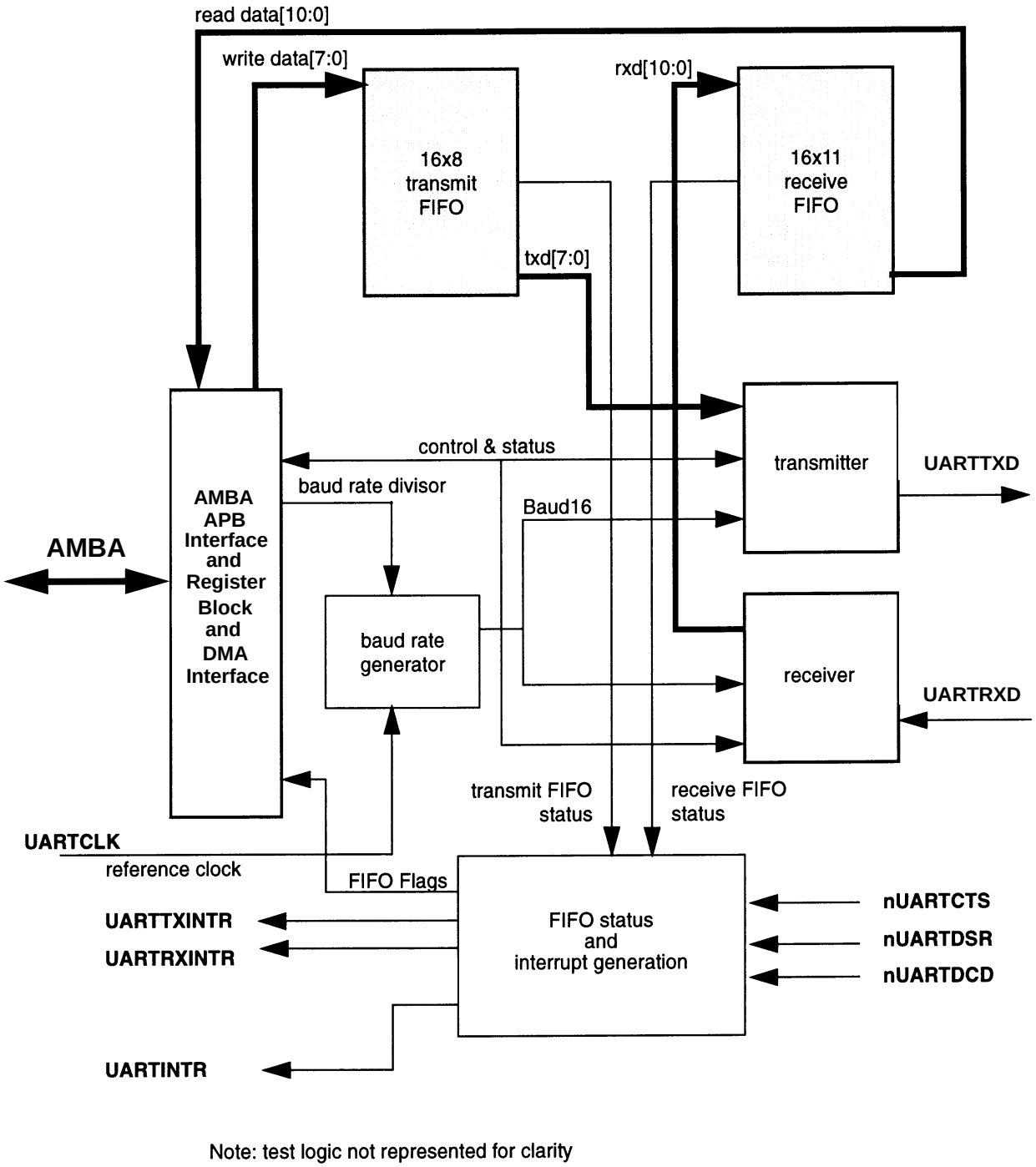
The UART1DMACtrl register controls the private interface between the DMA engine and the UART. Setting bit TXDMAE enables the transmit channel, while setting bit RXDMAE enables the receive channel. Setting bit DMAERR allows the UART to communicate certain error conditions to the DMA engine via RxEnd on the DMA channel. These conditions include receiving a break, a parity error, or a framing error. Note that configuration of the DMA channels in the DMA engine is also required for DMA operation with the UART.

14.2.1.3 Register Block

The register block stores data written or to be read across the AMBA APB interface.

Copyright 2007 Cirrus Logic

14



Note: test logic not represented for clarity

Figure 14-1. UART Block Diagram

Copyright 2007 Cirrus Logic

14.2.1.4 Baud Rate Generator

The baud rate generator contains free-running counters which generate the internal x16 clocks and the Baud16 signal. Baud16 provides timing information for UART transmit and receive control. Baud16 is a stream of pulses with a width of one UARTCLK clock period and a frequency of sixteen times the baud rate.

14.2.1.5 Transmit FIFO

The transmit FIFO is an 8-bit wide, 16-entry deep, first-in, first-out memory buffer. CPU data written across the APB interface and data written across the DMA interface is stored in the FIFO until read out by the transmit logic. The transmit FIFO can be disabled to act as a one-byte holding register.

14.2.1.6 Receive FIFO

The receive FIFO is an 11 bit wide, 16-entry deep, FIFO memory buffer. Received data, and corresponding error bits, are stored in the receive FIFO by the receive logic until read out by the CPU across the APB interface or across the DMA interface. The FIFO can be disabled to act as a one-byte holding register.

14.2.1.7 Transmit Logic

The transmit logic performs parallel-to-serial conversion on the data read from the transmit FIFO. Control logic outputs the serial bit stream beginning with a start bit, data bits, least significant bit (LSB) first, followed by parity bit, and then stop bits according to the programmed configuration in control registers.

14.2.1.8 Receive Logic

The receive logic performs serial-to-parallel conversion on the received bit stream after a valid start pulse has been detected. Parity, frame error checking and line break detection are also performed, and the data with associated parity, framing and break error bits is written to the receive FIFO.

14.2.1.9 Interrupt Generation Logic

Four individual maskable active HIGH interrupts are generated by the UART, and a combined interrupt output is also generated as an OR function of the individual interrupt requests.

The single combined UART interrupt (**UARTINTR**) is routed to the system interrupt controller. In addition, a separate receive FIFO interrupt **UARTRXINTR** and a transmit FIFO interrupt **UARTTXINTR** are routed to the system interrupt controller. (See Chapter 6,, "Vectored Interrupt Controller" on page 6-1 for additional details.) Separate receive and transmit FIFO status signals indicate to the DMA interface when there is room in the transmit FIFO for more data and when there is data in the receive FIFO.

Copyright 2007 Cirrus Logic

14.2.1.10 Synchronizing Registers and Logic

The UART supports both asynchronous and synchronous operation of the clocks, PCLK and UARTCLK. Synchronization registers and handshaking logic have been implemented, and are active at all times. This has a minimal impact on performance or area. Synchronization of control signals is performed on both directions of data flow, that is, from the PCLK to the UARTCLK domain and from the UARTCLK domain to the PCLK.

14.2.2 UART Operation

Control data is written to the UART line control register, UARTLCR. This register is 23 bits wide internally, but is externally accessed through the AMBA APB bus by three 8-bit wide register locations, UARTLCR_H, UARTLCR_M and UARTLCR.L.

UARTLCR defines the baud rate divisor and transmission parameters, word length, buffer mode, number of transmitted stop bits, parity mode and break generation.

The baud rate divisor is a 16-bit number used by the baud rate generator to determine the bit period. The baud rate generator contains a 16-bit down counter, clocked by the UART reference clock. When the value of the baud rate divisor has decremented to zero, the value of the baud rate divisor is reloaded into the down counter, and an internal clock enable signal, Baud16, is generated. This signal is then divided by 16 to give the transmit clock. A low number in the baud rate divisor gives a short bit period and vice versa.

Data received or transmitted is stored in two 16-byte FIFOs, though the receive FIFO has an extra three bits per character for status information.

For transmission, data is written into the transmit FIFO. This causes a data frame to start transmitting with the parameters indicated in UARTLCR. Data continues to be transmitted until there is no data left in the transmit FIFO. The **BUSY** signal goes HIGH as soon as data is written to the transmit FIFO (that is, the FIFO is non-empty) and remains asserted HIGH while data is being transmitted. **BUSY** is negated only when the transmit FIFO is empty, and the last character has been transmitted from the shift register, including the stop bits. **BUSY** can be asserted HIGH even though the UART may no longer be enabled.

When the receiver is idle (**UARTRXD** continuously 1, in the marking state) and a LOW is detected on the data input (a start bit has been received), the receive counter, with the clock enabled by Baud16, begins running and data is sampled on the eighth cycle of that counter (half way through a bit period).

The start bit is valid if **UARTRXD** is still LOW on the eighth cycle of Baud16, otherwise a false start bit is detected and it is ignored.

If the start bit was valid, successive data bits are sampled on every 16th cycle of Baud16 (that is, one bit period later) according to the programmed length of the data characters. The parity bit is then checked if parity mode was enabled.

Lastly, a valid stop bit is confirmed if **UARTRXD** is HIGH, otherwise a framing error has occurred. When a full word has been received, the data is stored in the receive FIFO, with any error bits associated with that word (see Table 14-1).

Copyright 2007 Cirrus Logic

14.2.2.1 Error Bits

Three error bits are stored in bits [10:8] of the receive FIFO, and are associated with a particular character. See Table 14-1. There is an additional error which indicates an overrun error but it is not associated with a particular character in the receive FIFO. The overrun error is set when the FIFO is full and the next character has been completely received in the shift register. The data in the shift register is overwritten but it is not written into the FIFO.

Table 14-1. Receive FIFO Bit Functions

FIFO bit	Function
10	Break error
9	Parity error
8	Framing error
7:0	Received data

14.2.2.2 Disabling the FIFOs

Additionally, it is possible to disable the FIFOs. In this case, the transmit and receive sides of the UART have 1-byte holding registers (the bottom entry of the FIFOs). The overrun bit is set when a word has been received and the previous one was not yet read. In this implementation, the FIFOs are not physically disabled, but the flags are manipulated to give the illusion of a 1-byte register.

14.2.2.3 System/diagnostic Loopback Testing

It is possible to perform loopback testing for UART data by setting the Loop Back Enable (LBE) bit to 1 in the control register UARTxCtrl (bit 7).

Data transmitted on **UARTTXD** output will be received on the **UARTRXD** input.

14.2.2.4 UART Character Frame

The UART character frame is shown in Figure 14-2:

Figure 14-2. UART Character Frame

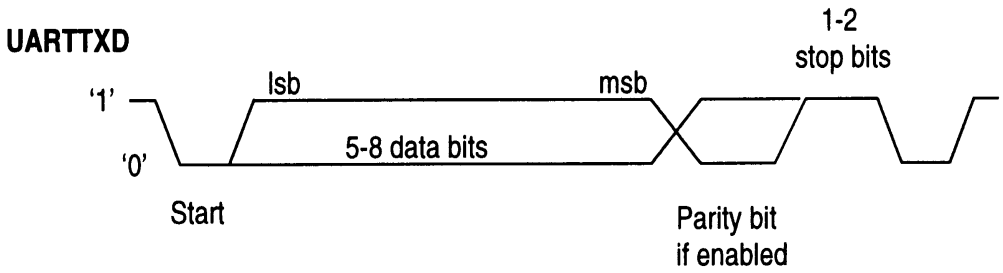


Figure 14-3. UART Character Frame

Copyright 2007 Cirrus Logic

14.2.3 Interrupts

There are five interrupts generated by the UART. Four of these are individual maskable active HIGH interrupts:

- **UARTMSINTR**
- **UARTRXINTR**
- **UARTRTINTR**
- **UARTTXINTR**

The interrupts are also output as a combined single interrupt **UARTINTR**.

Each of the four individual maskable interrupts is enabled or disabled by changing the mask bits in UARTCR. Setting the appropriate mask bit HIGH enables the interrupt.

The transmit and receive dataflow interrupts **UARTRXINTR** and **UARTTXINTR** have been separated from the status interrupts. This allows **UARTRXINTR** and **UARTTXINTR** to be used in a DMA controller, so that data can be read or written in response to just the FIFO trigger levels. The status of the individual interrupt sources can be read from UARTIIR.

14.2.3.1 UARTMSINTR

The modem status interrupt is asserted if any of the modem status lines (**nUARTCTS, nUARTDCD** and **nUARTDSR**) change. It is cleared by writing to the UART1IntIDIntClr register.

This interrupt is not independently connected to the system interrupt controller.

14.2.3.2 UARTRXINTR

The receive interrupt changes state when one of the following events occurs:

If the FIFOs are enabled and the receive FIFO is half or more full (it contains eight or more words), then the receive interrupt is asserted HIGH. The receive interrupt is cleared by reading data from the receive FIFO until it becomes less than half full.

If the FIFOs are disabled (have a depth of one location) and data is received thereby filling the location, the receive interrupt is asserted HIGH. The receive interrupt is cleared by performing a single read of the receive FIFO.

This interrupt is connected to the system interrupt controller.

14.2.3.3 UARTTXINTR

The transmit interrupt changes state when one of the following events occurs:

- If the FIFOs are enabled and the transmit FIFO is at least half empty (it has space for eight or more words), then the transmit interrupt is asserted HIGH. It is cleared by filling the transmit FIFO to more than half full.

Copyright 2007 Cirrus Logic

- If the FIFOs are disabled (have a depth of one location) and there is no data present in the transmitters single location, the transmit FIFO is asserted HIGH. It is cleared by performing a single write to the transmitter FIFO.

The transmit interrupt **UARTTXINTR** is not qualified with the UART Enable signal, which allows operation in one of two ways. Data can be written to the transmit FIFO prior to enabling the UART and the interrupts. Alternatively, the UART and interrupts can be enabled so that data can be written to the transmit FIFO by an interrupt service routine.

This interrupt is connected to the system interrupt controller.

14.2.3.4 UARTRTINTR

The receive timeout interrupt is asserted when the receive FIFO is not empty and no further data is received over a 32-bit period. The receive timeout interrupt is cleared when the FIFO becomes empty through reading all the data (or by reading the holding register).

This interrupt is not independently connected to the system interrupt controller.

14.2.3.5 UARTINTR

The interrupts are also combined into a single output which is an OR function of the individual masked sources. This output is connected to the system interrupt controller to provide another level of masking on a individual peripheral basis. The combined UART interrupt is asserted if any of the four individual interrupts above are asserted and enabled.

14.3 Modem

The modem hardware adds modem control signals **RTSn**, **DTRn**, and **RI**. Two modem support registers provide a 16550 compatible modem control interface.

14.4 HDLC

The HDLC receiver handles framing, address matching, CRC checking, control-octet transparency or bit-stuffing, and optionally passes the CRC to the CPU at the end of the packet. The HDLC transmitter handles framing, CRC generation, and control-octet transparency or bit-stuffing. The CPU must assemble the frame in memory before transmission. The HDLC receiver and transmitter use the UART FIFOs to buffer the data streams.

When entering HDLC mode, always enable HDLC transmit and/or receive first by setting the TXE and/or RXE bit in the UART1HDLCCtrl, and then enable the UART. When leaving HDLC mode, disable the UART first, and then disable HDLC transmit and/or receive by clearing the TXE and/or RXE bit. This insures that no bytes are sent by the UART transmitter without proper HDLC framing, and that no bytes are received via the UART receiver without proper HDLC decoding. In HDLC mode, the UART should be configured to use 8-bit characters and no parity bit.

Copyright 2007 Cirrus Logic

14.4.1 Overview of HDLC Modes

HDLC may operate in one of two basic modes, synchronous or asynchronous. Most configuration options affect both modes identically. Setting the UART1HDLCCtrl.SYNC bit selects synchronous mode and clearing it selects asynchronous mode. In asynchronous mode, each byte is transmitted using standard UART protocol framing (that is, start bit, data, parity, stop bit(s)). In synchronous mode, UART framing is bypassed.

The synchronous HDLC bit stream may be either a NRZ or Manchester encoded. In NRZ mode, both the transmitter and receiver may be synchronized to either an external or internal clock running at one cycle per bit period. The transmitter and receiver may operate independently in any of the four modes:

- Simple NRZ mode
- Manchester encoded
- NRZ mode with an internal clock
- NRZ mode with an external clock

In the first NRZ mode, the data stream does not contain an explicit or implicit clock, so synchronization between an HDLC transmitter and receiver cannot be guaranteed. A data bit value of "1" is encoded as a one in the bit stream, and a value of "0" as a zero.

The second mode, Manchester encoding, combines the HDLC data and clock into a single bit stream. In Manchester encoding, a transition always occurs in the middle of a transmitted bit and the value after this transition is the actual value of the bit. That is, a "0" bit is represented by a transition from high to low, and a "1" bit by a transition from low to high. Because a transition always occurs in the middle of a bit, the receiver can always extract the proper data after a suitable period of synchronization, provided the signal quality is good.

The third and fourth modes utilize NRZ encoding of the data accompanied by a separate clock signal. The period of the clock signal is one bit period. When using an internal clock, the HDLC transmitter generates a clock such that the data is stable at the clock's rising edge. Hence, an external receiver may sample each data bit at the rising edge of the clock. The internal receiver will also use the same clock to sample input data if programmed to do so.

The internal transmitter and/or receiver may also synchronize to an external, rather than internal, clock. The internal receiver gets this clock along with the incoming HDLC data, allowing it to always sample bits at the right time. In addition, the internal transmitter will synchronize the data it transmits to this clock if programmed to do so. The transmitter will insure that its data is valid before the rising edge of the clock, and the receiver expects the same of the incoming bit stream.

14.4.2 Selecting HDLC Modes

By default, HDLC is NRZ-encoded. Set bit UART1HDLCCtrl.TXENC to force Manchester encoding in the transmitter, and set bit UART1HDLCCtrl.RXENC to make the receiver expect Manchester encoding.

Copyright 2007 Cirrus Logic

The receiver utilizes a digital PLL to synchronize to the incoming encoded bit stream. The digital PLL should always successfully lock on to an incoming data stream within two bytes provided that the first two bits of the first byte are either "01" or "10". Hence, at a minimum, two bytes must precede the final opening flag to insure that the HDLC receiver sees the packet. To meet this requirement, the simplest approach is to insure that at least three opening flags are received if the packet is Manchester encoded. (Note that to meet this requirement when transmitting, field HDLC1Ctrl.FLAG should be set to 0010b.)

Three bits in various combination determine how an external or internal clock may be used along with NRZ data. The clock will have a period equal to the bit period of the data stream, and it is expected that the internal or external receiver will sample the bit at or near the rising edge of this clock.

To generate an internal clock suitable for sending along with the transmitted data, set UART1HDLCCtrl.TXCM and UART1HDLCCtrl.CMAS. To make the receiver use the same internal clock, set UART1HDLCCtrl.RXCM. To make the receiver use an externally generated clock, clear UART1HDLCCtrl.CMAS, but set UART1HDLCCtrl.RXCM.

To force the transmitter to use the same external clock, also set UART1HDLCCtrl.TXCM. The clock is either internal or external, that is, the receiver cannot use an external clock while the transmitter generates and sends an internal one. Refer to the documentation for the DeviceCfg register in Syscon for the use and routing of HDLC clocks to or from external pins on the device.

The internal clock is generated by the transmitter only while it is sending data or flags; the clock is not generated while the transmitter is idle. For this reason, another transmitter which expects to use this clock to at any time send its own packets cannot reliably do so. To insure that a clock is continuously generated, the IDLE bit in the UART1HDLCCtrl register may be set, which causes this transmitter to continuously send flags between packets instead of going idle.

Table 14-2 summarizes the legal HDLC mode configurations.

Table 14-2. Legal HDLC Mode Configurations

UART1HDLCCtrl Bits Set						Transmit Mode	Receive Mode
CMAS	TXCM	RXCM	TXENC	RXENC	SYNC		
-	-	-	-	-	-	Asynchronous NRZ	Asynchronous NRZ
-	-	-	-	-	1	Synchronous NRZ	Synchronous NRZ
-	-	-	-	1	1	Synchronous NRZ	Manchester
-	-	-	1	-	1	Manchester	Synchronous NRZ
-	-	-	1	1	1	Manchester	Manchester
-	-	1	-	-	1	Synchronous NRZ	External clock
-	-	1	1	-	1	Manchester	External clock
-	1	-	-	-	1	External clock	Synchronous NRZ
-	1	-	-	1	1	External clock	Manchester
1	1	-	-	-	1	Internal clock	Synchronous NRZ

Copyright 2007 Cirrus Logic

Table 14-2. Legal HDLC Mode Configurations (Continued)

UART1HDLCCtrl Bits Set						Transmit Mode	Receive Mode
CMAS	TXCM	RXCM	TXENC	RXENC	SYNC		
1	1	-	-	1	1	Internal clock	Manchester
-	1	1	-	-	1	External clock	External clock
1	1	1	-	-	1	Internal clock	Internal clock

14.4.3 HDLC Transmit

In normal operation, the HDLC transmitter either continuously sends flags or holds the transmit pin in a marking state, depending on the setting of the UART1HDLCCtrl.IDLE bit. When data appears in the transmit FIFO, it begins sending a packet. If in the marking state, it sends from 1 to 16 opening flags, as specified by the UART1HDLCCtrl.FLAG field. If already sending flags, it ensures that at least the specified number have been sent. It then begins sending the bytes in the FIFO, inserting and modifying the data depending on the HDLC mode.

In asynchronous HDLC, the transmitter enforces control-octet transparency. Whenever a flag byte (01111110b) or an escape byte (01111101b) appears in the data, the transmitter inverts the fifth bit and precedes it with an escape byte.

In synchronous HDLC, the transmitter performs bit-stuffing (except for flags). Whenever five consecutive "1" bits appear in the transmitted bit stream, a "0" bit is inserted, preventing six ones from appearing consecutively.

When the transmit FIFO under-runs, the HDLC transmitter does one of two things (depending on the setting of the UART1HDLCCtrl.TUS bit). If the TUS bit is zero, the transmitter first sends the CRC (if CRC is enabled) and then sends from 1 to 16 closing flags, as specified in the UART1HDLCCtrl.FLAG field, terminating the packet.

If TUS is one, the transmitter aborts the packet. In synchronous HDLC, it sends a byte of all ones (since seven consecutive ones signifies an abort), following by at least one closing flag. In asynchronous HDLC, it sends an escape and then at least one closing flag. The number of closing flags is from 1 to 16, as specified in the UART1HDLCCtrl.FLAG field.

When a packet ends, the UART1HDLCSts.TFC bit is set, and if UART1HDLCCtrl.TFCEN is set, an interrupt is generated. When a packet is aborted, the UART1HDLCCtrl.TAB bit is set, also generating an interrupt if UART1HDLCCtrl.TABEN is set.

14.4.4 HDLC Receive

The HDLC receiver continuously reads bytes from the UART receiver until it finds a flag followed by a byte other than a flag. Then, if in asynchronous mode, it processes the incoming bytes (including the first after the flag), reversing control-octet transparency, or, if in synchronous mode, it reverses bit-stuffing. Processed bytes are placed in the receive FIFO. When programmed to receive a Manchester encoded bit stream, UART1HDLCSts.PLLCS indicates whether the DPLL in the receiver has locked on to the carrier.

Copyright 2007 Cirrus Logic

When the last byte of data for a packet is read from the receive FIFO, the HDLC logic sets a number of bits in the UART1HDLCSts depending on the state of the system and the way the packet was terminated. In all cases, the RFC bit and EOF bit are set. If the receive FIFO overflowed while the packet was being received, the ROR bit is also set. If CRC is enabled and the received CRC does not match the calculated one, the CRE bit is set. The RFC bit is set and, if UART1HDLCCtrl.RFCEN is set, an interrupt is generated. If the packet was aborted, the RAB bit is set, and an interrupt generated if the UART1HDLCCtrl.RABEN bit is set. If using Manchester encoding and the packet was aborted due to losing synchronization with the encoded clock, the UART1HDLCCtrl.PLLE bit is set.

Besides setting bits in the UART1HDLCSts and possibly causing interrupts, reading the last byte of a packet also loads the UART1HDLCRXInfoBuf register with data describing the packet. BRAB, BCRE, BROR, and BPLLE are copied from RAB, CRE, ROR, and PLLE in the UART1HDLCSts. BFRE is copied from the FE bit in the UART1RXSts. BC is set to the number of bytes in the packet that were read from the FIFO. Whenever this register is written by the receiver and has not been read since previously it was previously written, the UART1HDLCSts.RIL bit is set, and, if UART1HDLCSts.RILEN is set, an interrupt is generated.

If a new packet is received and the first byte of that packet cannot be written into the receive FIFO because it has overflowed, the UART1HDLCSts.RFL bit is set and the packet is discarded. An interrupt is generated if the UART1HDLCCtrl.RFLEN bit is also set.

14.4.5 CRCs

Several bits in the UART1HDLCCtrl determine how CRCs are generated by the transmitter and processed by the receiver. By setting the CRCE bit, the HDLC transmitter will calculate and append a CRC to each packet. The CRC may be either 16-bit or 32-bit, depending on the CRCS bit. Furthermore, it will be inverted prior to transmission if the CRCN bit is set. If CRCs are enabled, the receiver will expect the same type of CRC that the transmitter sends. It will automatically calculate the CRC for the received packet in the fly, and if the calculated CRC does not match the received one, the UART1RXSts.CRE bit will be set when the last byte of the received packet is read from the UART1Data. The receiver does not pass the CRC to the CPU unless the CRCApd bit is set.

14.4.6 Address Matching

When address matching is enabled, the HDLC receiver will ignore any packet whose address does not match the programmed configuration. Address matching is enabled and address size specified by the UART1HDLCCtrl.AME bits. The UART1HDLCAddMtchVal specifies the addresses that are compared while the UART1HDLCAddMask controls which bits in each address are compared If one-byte addressing is used, each byte in UART1HDLCAddMtchVal specifies an address to match, while the corresponding byte in UART1HDLCAddMask specifies which bits of each address must match. If two-byte addressing is used, each half-word in UART1HDLCAddMtchVal specifies an address to match and the corresponding half-word in UART1HDLCAddMask specifies which bits of each address to match. Hence, up to four different one-byte addresses and two different two-byte addresses may be specified. An

Copyright 2007 Cirrus Logic

incoming address consisting entirely of "1"s, that is, 0xFF or 0xFFFF, will always match, as it is expected to be the broadcast address. For packets whose addresses do not match, the HDLC receiver will generate no interrupts, modify no status bits, and place no data in the receive FIFO.

Table 14-3. HDLC Receive Address Matching Modes

AME	Match Function	Address Match Test
00	No matching	
01	One byte address	***NOT***((AMV[31:24] ***XOR*** ADDR) ***AND*** AMSK[31:24]) ***OR*** ***NOT***((AMV[23:16] ***XOR*** ADDR) ***AND*** AMSK[23:16]) ***OR*** ***NOT***((AMV[15:8] ***XOR*** ADDR) ***AND*** AMSK[15:8]) ***OR*** ***NOT***((AMV[7:0] ***XOR*** ADDR) ***AND*** AMSK[7:0]) ***OR*** ADDR = 0xFF
10	Two byte address	***NOT***((AMV[31:16] ***XOR*** ADDR) ***AND*** AMSK[31:16]) ***OR*** ***NOT***((AMV[15:0] ***XOR*** ADDR) ***AND*** AMSK[15:0]) ***OR*** ADDR = 0xFFFF
11	Undefined	

14.4.7 Aborts

If a packet is aborted or is too short, or if using Manchester encoding and the receiver DPLL loses the carrier signal, the CPU will see at least some part of the packet in the receive FIFO. In all cases, reading the last byte of the packet from the receive FIFO will set the EOF and RAB bits in the UART1HDLCSts (and possibly generate an interrupt). In the case of an abort indicated by an HDLC transmitter, that is, an escape-closing flag sequence in asynchronous mode or an all "1"s byte in synchronous mode, all bytes received in the frame will appear in the receive FIFO.

In asynchronous mode, if the abort is caused by a framing error (a missing stop bit), all bytes up to and including the misframed byte will appear in the receive FIFO. Reading the last byte will also set the UART1HDLCSts.FRE bit.

In synchronous mode, if the abort is caused by a misaligned flag or a series of seven consecutive "1"s, all bytes except the one containing the bit after the sixth "1" will appear in the receive FIFO. If the abort is caused by the receiver DPLL losing synchronization with a Manchester encoded bit stream, the UART1HDLCSts.DPLLE bit is set.

Finally, if the packet is too short, that is, there are not enough received bytes to hold the specified number of address and CRC bytes, the entire packet will appear in the receive FIFO. In all cases, the packet is illegal and will be ignored by the CPU.

Copyright 2007 Cirrus Logic

14.4.8 DMA

The DMA engine may be used with the UART when transmitting and receiving HDLC packets. The transmit and receive channels may operate completely independently.

When receiving data in HDLC mode, the DMA channel reads the packet data byte by byte from the RX FIFO. When it reads the final byte, the HDLC RFC interrupt will occur if enabled. However, the DMA channel, which buffers the data, may not write all of the data to memory. To insure that the DMA channel dumps the data, the interrupt handling routine must do the following:

1. Note the values in the MAXCNTx and REMAIN registers for the DMA channel. The difference is the number of bytes read from the UART, which is the size of the HDLC packet. Call this difference N. Note that the BC field of the UART1HDLCRXInfoBuf register should also be N.
2. Temporarily disable the UART DMA RX interface by clearing the RXDMAE bit in the UART1DMACtrl register.
3. Wait until the difference between the CURRENTx and BASEx registers in the DMA channel is equal to N + 1.

An extra byte will be read from the UART by the DMA channel. It should be ignored.

Note that if the DMAERR bit in the UART1DMACtrl register is set and the HDLC receiver is in asynchronous mode, if the receiver sees a break, parity, or framing error, it will indicate an error condition via RxEnd on the DMA channel.

14.4.9 Writing Configuration Registers

It is assumed that various configuration registers for the UART/HDLC are not written more than once in quick succession, in order to insure proper synchronization of configuration information across the implementation. Such registers include UART1Ctrl and UART1LinCtrlHigh as well as UART1HDLCCtrl, UART1HDLCAddMtchVal, UART1HDLCAddMask. These registers should not change often in typical use.

The simplest way to fulfill this requirement with respect to writing the UART1Ctrl and UART1HDLCCtrl registers is to insure that the HDLC transmitter is enabled before the UART transmit logic. This will ensure that the UART does not transmit incorrect characters or unexpectedly transmit characters with UART framing,

First the UART1HDLCCtrl register should be written, setting the TXE bit. Then the UART1Ctrl register should be written, setting the UARTE bit. In between the two writes, at least two UARTCLK periods must occur. Under worst case conditions, at least 55 HCLK periods must separate the two writes. The simplest way to due this is separate the two writes by 55 NOPs.

14.5 UART1 Package Dependency

UART1 uses package pins **RXD0**, **TXD0**, **CTSn**, **DSRn**, **DTRn**, **RTSn**, **EGPIO[3]**, and **EGPIO[0]**, which are described in Table 14-4.

Copyright 2007 Cirrus Logic

Table 14-4. UART1 Pin Functionality

PIN	Description
RXD0	UART1 input pin
TXD0	UART1 output pin
CTSn	Modem input: Clear To Send
DSRn	Modem input: Data Set Ready (also used for DCDn Data Carrier Detect)
EGPIO[0]	Modem input RIn: Ring Indicator if Syscon register DeviceCfg[25] MODonGPIO is set. Otherwise, RIn is driven low.
DTRn	Modem output Data Terminal Ready if Syscon register TESTCR[27] RTConGPIO is clear.
RTSn	Modem output: Ready To Send
EGPIO[3]	HDLC clock

The use of **EGPIO[3]** is determined by several bits in Syscon register DeviceCfg. See Table 14-5.

Table 14-5. DeviceCfg Register Bit Functions

bit 14 HC3EN	bit 13 HC1IN	bit 12 HC1EN	Function
x	0	x	External HDLC clock input is driven low.
0	1	1	External HDLC clock input is driven by EGPIO[3].
0	0	1	Internal HDLC clock output drives EGPIO[3].

14.5.1 Clocking Requirements

There are two clocks, PCLK and UARTCLK.

UARTCLK frequency must accommodate the desired range of baud rates:

$$F_{UARTCLK_{MIN}} \geq 32 \times baudrate_{MAX}$$

$$F_{UARTCLK_{MAX}} \leq 32 \times 65536 \times baudrate_{MIN}$$

The frequency of UARTCLK must also be within the required error limits for all baud rates to be used.

To allow sufficient time to write the received data to the receive FIFO, UARTCLK must be less than or equal to four times the frequency of PCLK:

$$F_{UARTCLK} \leq 4 \times F_{PCLK}$$

Copyright 2007 Cirrus Logic

14.5.2 Bus Bandwidth Requirements

There are two basic ways of moving data to and from the UART FIFOs:

- Direct DMA interface - This permits byte-wide access to the UART without using the APB. The DMA block will pack or unpack individual bytes so that it reads or writes full 32-bit words rather than individual bytes.
- Accessing the UART via the APB - This requires APB/AHB bus bandwidth. Then, both a read and write are required for each 8-bit data byte.

Bandwidth requirements also depend on the selected baud rate, character size, parity selection, number of stop bits, and spacing between characters (if receiving).

For example, assume transmission protocols of 115,200 baud, 8-bit characters, even parity, one stop bit, no space between characters. There are 11 bits per character, so 115,200 / 11 = 10,473 characters per second. If both transmitting and receiving, 20,945 characters per second pass through the UART. Accessing the UART through the DMA interface requires one access per 32-bits, implying only 20,945 / 4 = 5,236 AHB accesses per second. Accessing the UART through the APB requires two accesses per byte, implying 20,945 APB bus accesses.

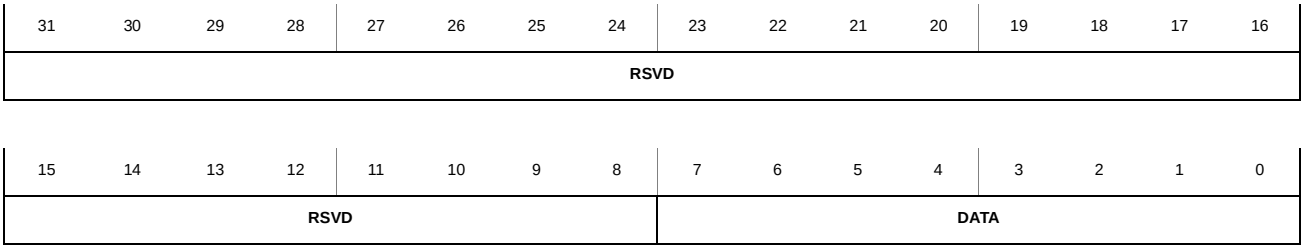
As another example, assume 230,400 baud (the maximum with a UARTCLK equal to 7.3728 Mhz), 5-bit characters, no parity, one stop bit, and no space between characters. There are 7 bits per character, so 230,400 / 7 = 32,914 characters per second. Simultaneous transmitting and receiving implies 65,829 characters per second. Using the DMA interface would result in 16,457 AHB accesses per second, while using the APB to access the UART leads to 65,829 bus accesses per second.

Copyright 2007 Cirrus Logic

14.1 Registers

UART Register Descriptions

UART1Data



Address:
0x808C_0000 - Read/Write

Default:
0x0000_0000

Definition:
UART Data Register

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
DATA:	UART Data: read for receive data, write for transmit data

For words to be transmitted:

- • if the FIFOs are enabled, data written to this location is pushed onto the transmit FIFO
  • if the FIFOs are not enabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO). The write operation initiates transmission from the UART. The data is prefixed with a start bit, appended with the appropriate parity bit (if parity is enabled), and a stop bit. The resultant word is then transmitted.

  For received words:
  • if the FIFOs are enabled, the data byte is extracted, and a 3-bit status (break, frame and parity) is pushed onto the 11-bit wide receive FIFO
  • if the FIFOs are not enabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO).

  The received data byte is read by performing reads from the UART1Data register while the corresponding status information can be read by a successive read of the UART1RXSts register.

Copyright 2007 Cirrus Logic

UART1RXSts

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												OE	BE	PE	FE

Address:

0x808C_0004 - Read/Write

Default:

0x0000_0000

Definition:

UART1 Receive Status Register/Error Clear Register. Provides receive status of the data value last read from the UART1Data. A write to this register clears the framing, parity, break and overrun errors. The data value is not important. Note that BE, PE and FE are not used for synchronous HDLC.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

OE: Overrun Error. This bit is set to "1" if data is received and the FIFO is already full. This bit is cleared to "0" by a write to UART1RXSts. The FIFO contents remain valid since no further data is written when the FIFO is full. Only the contents of the shift register are overwritten. The data must be read in order to empty the FIFO.

BE: Break Error. This bit is set to 1 if a break condition was detected, indicating that the received data input was held LOW for longer than a full-word transmission time (defined as start, data, parity and stop bits). This bit is cleared to 0 after a write to UART1RXSts. In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a 1 (marking state) and the next valid start bit is received.

PE: Parity Error. When this bit is set to 1, it indicates that the parity of the received data character does not match the parity selected in UART1LinCtrlHigh (bit 2). This bit is cleared to 0 by a write to UART1RXSts. In FIFO mode, this error is associated with the character at the top of the FIFO.

Copyright 2007 Cirrus Logic

FE: Framing Error. When this bit is set to 1, it indicates that the received character did not have a valid stop bit (a valid stop bit is "1"). This bit is cleared to 0 by a write to UART1RXSts. In FIFO mode, this error is associated with the character at the top of the FIFO.

UART1LinCtrlHigh

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									WLEN		FEN	STP2	EPS	PEN	BRK

Address:

0x808C_0008 - Read/Write

Default:

0x0000_0000

Definition:

UART1 Line Control Register High. UART1LinCtrlHigh, UART1LinCtrlMid and UART1LinCtrlLow form a single 23-bit wide register (UART1LinCtrl) which is updated on a single write strobe generated by an UART1LinCtrlHigh write. In order to internally update the contents of UART1LinCtrlMid or UART1LinCtrlLow, a UART1LinCtrlHigh write must always be performed at the end.

To update the three registers there are two possible sequences:

- UART1LinCtrlLow write, UART1LinCtrlMid write and UART1LinCtrlHigh write
- UART1LinCtrlMid write, UART1LinCtrlLow write and UART1LinCtrlHigh write.

To update UART1LinCtrlLow or UART1LinCtrlMid only:

- UART1LinCtrlLow write (or UART1LinCtrlMid write) and UART1LinCtrlHigh write.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

WLEN: Number of bits per frame:
11 = 8 bits
10 = 7 bits
01 = 6 bits
00 = 5 bits

Copyright 2007 Cirrus Logic

FEN: FIFO Enable.
1 - Transmit and receive FIFO buffers are enabled (FIFO mode).
0 - The FIFOs are disabled (character mode) that is, the FIFOs become 1-byte-deep holding registers.

STP2: Two Stop Bits Select.
1 - Two stop bits are transmitted at the end of the frame.
0 - One stop bit is transmitted at the end of the frame.
The receive logic does not check for two stop bits being received.

EPS: Even Parity Select.
1 - Even parity generation and checking is performed during transmission and reception, which checks for an even number of "1"s in data and parity bits.
0 - Odd parity checking is performed, which checks for an odd number of "1"s.
This bit has no effect when parity is disabled by Parity Enable (bit 1) being cleared to 0.

PEN: Parity Enable.
1 - Parity checking and generation is enabled,
0 - Parity checking and generation is disabled and no parity bit is added to the data frame.

BRK: Send Break.
1 - A low level is continually output on the **UARTTXD** output, after completing transmission of the current character. This bit must be asserted for at least one complete frame transmission time in order to generate a break condition. The transmit FIFO contents remain unaffected during a break condition.
0 - For normal use, this bit must be cleared.

UART1LinCtrlMid

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								BR							

Address: 0x808C_000C - Read/Write

Default: 0x0000_0000

Copyright 2007 Cirrus Logic

Definition:

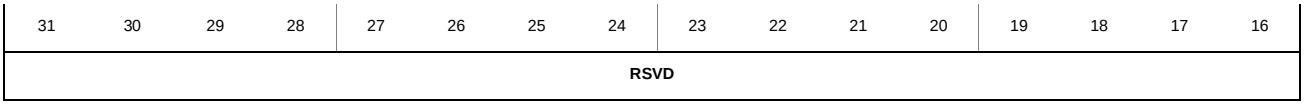
UART Line Control Register Middle.

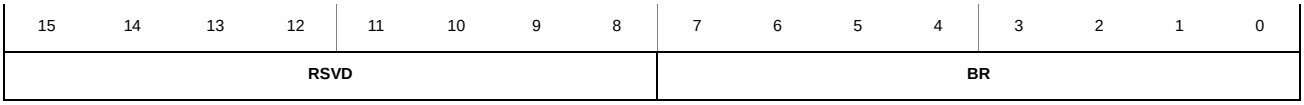
Bit Descriptions:

RSVD: Reserved. Unknown During Read.

BR: Baud Rate Divisor bits [15:8]. Most significant byte of baud rate divisor. These bits are cleared to 0 on reset.

UART1LinCtrlLow





Address:

0x808C_0010 - Read/Write

Default:

0x0000_0000

Definition:

UART Line Control Register Low.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

BR: Baud Rate Divisor bits [7:0]. Least significant byte of baud rate divisor. These bits are cleared to 0 on reset. The baud rate divisor is calculated as follows:

Baud rate divisor
BAUDDIV = ($F_{UARTCLK}$ / 16 * Baud rate)) – 1

where $F_{UARTCLK}$ is the UART reference clock frequency. A baud rate divisor of zero is not allowed and will result in no data transfer.

Copyright 2007 Cirrus Logic

UART1Ctrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								LBE	RTIE	TIE	RIE	MSIE	RSVD		UARTE

Address:

0x808C_0014 - Read/Write

Default:

0x0000_0000

Definition:

UART1 Control Register

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
LBE:	Loopback Enable. If this bit is set to 1, data sent to **TXD** is received on **RXD**. This bit is cleared to 0 on reset, which disables the loopback mode.
RTIE:	Receive Timeout Enable. If this bit is set to 1, the receive timeout interrupt is enabled.
TIE:	Transmit Interrupt Enable. If this bit is set to 1, the transmit interrupt is enabled.
RIE:	Receive Interrupt Enable. If this bit is set to 1, the receive interrupt is enabled.
MSIE:	Modem Status Interrupt Enable. If this bit is set to 1, the modem status interrupt is enabled.
UARTE:	UART Enable. If this bit is set to 1, the UART is enabled. Data transmission and reception occurs for UART signals.

UART1Flag

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								TXFE	RXFF	TXFF	RXFE	BUSY	DCD	DSR	CTS

Copyright 2007 Cirrus Logic

Address:

0x808C_0018 - Read Only

Default:

0x0000_0000

Definition:

UART Flag Register

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
TXFE:	Transmit FIFO Empty. The meaning of this bit depends on the state of the FEN bit in the UART1LinCtrlHigh register. If the FIFO is disabled, this bit is set when the transmit holding register is empty. If the FIFO is enabled, the TXFE bit is set when the transmit FIFO is empty.
RXFF:	Receive FIFO Full. The meaning of this bit depends on the state of the FEN bit in the UART1LinCtrlHigh register. If the FIFO is disabled, this bit is set when the receive holding register is full. If the FIFO is enabled, the RXFF bit is set when the receive FIFO is full.
TXFF:	Transmit FIFO Full. The meaning of this bit depends on the state of the FEN bit in the UART1LinCtrlHigh register. If the FIFO is disabled, this bit is set when the transmit holding register is full. If the FIFO is enabled, the TXFF bit is set when the transmit FIFO is full.
RXFE:	Receive FIFO Empty. The meaning of this bit depends on the state of the FEN bit in the UART1LinCtrlHigh register. If the FIFO is disabled, this bit is set when the receive holding register is empty. If the FIFO is enabled, the RXFE bit is set when the receive FIFO is empty.
BUSY:	UART Busy. If this bit is set to 1, the UART is busy transmitting data. This bit remains set until the complete byte, including all the stop bits, has been sent from the shift register. This bit is set as soon as the transmit FIFO becomes non-empty (regardless of whether the UART is enabled or not).
DCD:	Data Carrier Detect status. This bit is the complement of the UART data carrier detect (**nUARTDCD**) modem status input. That is, the bit is 1 when the modem status input is 0.
DSR:	Data Set Ready status. This bit is the complement of the UART data set ready (**nUARTDSR**) modem status input. That is, the bit is 1 when the modem status input is 0.

Copyright 2007 Cirrus Logic

CTS: Clear To Send status. This bit is the complement of the UART clear to send (**nUARTCTS**) modem status input. That is, the bit is 1 when the modem status input is 0.

UART1IntIDIntClr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												RTIS	TIS	RIS	MIS

Address:

0x808C_001C - Read/Write

Default:

0x0000_0000

Definition:

UART Interrupt Identification and Interrupt Clear Register.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RTIS: Receive Timeout Interrupt Status. This bit is set to 1 if the **UARTRTINTR** receive timeout interrupt is asserted. This bit is cleared when the receive FIFO is empty or the receive line goes active.

TIS: Transmit Interrupt Status.
1 - The **UARTTXINTR** transmit interrupt is asserted, which occurs when the transmit FIFO is not full.
0 - The transmit FIFO is full.

RIS: Receive Interrupt Status.
1 - The **UARTRXINTR** receive interrupt is asserted, which occurs when the receive FIFO is not empty.
0 - The receive FIFO is empty.

MIS: Modem Interrupt Status. This bit is set to 1 if the **UARTMSINTR** modem status interrupt is asserted. This bit is cleared by writing any value to this register.

Copyright 2007 Cirrus Logic

UART1DMACtrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD													DMAERR	TXDMAE	RXDMAE

Address:

0x808C_0028 - Read/Write

Default:

0x0000_0000

Definition:

UART DMA Control Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

DMAERR: RX DMA error handing enable. If 0, the RX DMA interface ignores error conditions in the UART receive section. If 1, the DMA interface stops and notifies the DMA block when an error occurs. Errors include break errors, parity errors, and framing errors.

TXDMAE: TX DMA interface enable. Setting to 1 enables the private DMA interface to the transmit FIFO.

RXDMAE: RX DMA interface enable. Setting to 1 enables the private DMA interface to the receive FIFO.

Modem Register Descriptions

UART1ModemCtrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								0	0	0	LOOP	OUT2	OUT1	RTS	DTR

Address:

0x808C_0100 - Read/Write

Copyright 2007 Cirrus Logic

Default:

0x0000_0000

Definition:

Modem Control Register

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
0:	Must be written as "0".
LOOP:	Activate internal modem control loopback function. This internal loopback only affects the hardware handshake signals. Use the UART1Ctrl LBE bit to loopback the serial data. When high, modem control outputs **RTSn** and **DTRn** are forced high (inactive), and modem control inputs are driven by outputs: DSR = DTR CTS = RTS RI2 = OUT1 DCD = OUT2
OUT2:	OUT2 function. Used for internal loopback.
OUT1:	OUT1 function. Used for internal loopback.
RTS:	RTS output signal: 1 - **RTSn** pin low 0 - **RTSn** pin high
DTR:	DTR output signal: 1 - **DTRn** pin low 0 - **DTRn** pin high

UART1ModemSts

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								DCD	RI	DSR	CTS	DDCD	TERI	DDSR	DCTS

Address:

0x808C_0104 - Read Only

Default:

0x0000_0000

Copyright 2007 Cirrus Logic

Definition:

Modem Status Register

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
DCD:	Inverse of **DCDn** input pin. Note that this is identical to the **DSR** device pin.
RI:	Inverse of **RI** input pin.
DSR:	Inverse of the **DSRn** pin. Note that this is identical to the **DCD** device pin
CTS:	Inverse **CTSn** input pin.
DDCD:	Delta **DCD** - **DCDn** pin changed state since last read.
TERI:	Trailing Edge Ring Indicator. **RI** input pin has changed from low to high.
DDSR:	Delta **DSR** - **DSRn** pin has changed state since last read.
DCTS:	Delta **CTS** - **CTSn** pin has changed state since last read.

HDLC Register Descriptions

UART1HDLCCtrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				CMAS	TXCM	RXCM	TXENC	RXENC	SYNC	TFCEN	TABEN	RFCEN	RILEN	RFLEN	RTOEN

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLAG				CRCN	CRCApd	IDLE	AME		IDLSpc	CRCZ	RXE	TXE	TUS	CRCE	CRCS

Address:

0x808C_020C - Read/Write

Default:

0x0000_0000

Definition:

HDLC Control Register

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.

Copyright 2007 Cirrus Logic

CMAS: Clock Master:
1 - Transmitter and/or receiver use 1x clock generated by the internal transmitter.
0 - Transmitter and/or receiver use 1x clock generated externally.

TXCM: Transmit Clock Mode.
1 - Generate 1x clock when in synchronous HDLC mode using NRZ encoding.
0 - Do not generate clock.
This bit has no effect unless TXENC is clear and synchronous HDLC is enabled.

RXCM: Receive Clock Mode.
1 - Use external 1x clock when in synchronous HDLC mode using NRZ encoding.
0 - Do not use external clock.
This bit has no effect unless RXENC is clear and synchronous HDLC is enabled.

TXENC: Transmit Encoding method.
1 - Use Manchester bit encoding.
0 - Use NRZ bit encoding.
This bit has no effect unless synchronous HDLC is enabled

RXENC: Receive Encoding method.
1 - Use Manchester bit encoding.
0 - Use NRZ bit encoding.
This bit has no effect unless synchronous HDLC is enabled.

SYNC: Synchronous / Asynchronous HDLC Enable.
0 - Select asynchronous HDLC for TX and RX.
1 - Select synchronous HDLC for TX and RX.

TFCEN: Transmit Frame Complete Interrupt Enable.
0 - TFC interrupt will not occur.
1 - TFC interrupt will occur whenever TFC bit is set.

TABEN: Transmit Frame Abort Interrupt Enable.
0 - TAB interrupt will not occur.
1 - TAB interrupt will occur whenever TAB bit is set.

RFCEN: Receive Frame Complete Interrupt Enable.
0 - RFC interrupt will not occur.
1 - RFC interrupt will occur whenever RAB bit or EOF bit is set.

Copyright 2007 Cirrus Logic

RILEN: Receive Information Lost Interrupt Enable.
0 - RIL interrupt will not occur.
1 - RIL interrupt will occur whenever RIL bit is set.

RFLEN: Receive Frame Lost Interrupt Enable.
0 - RFL interrupt will not occur.
1 - RFL interrupt will occur whenever RFL bit is set.

RTOEN: Receiver Time Out Interrupt Enable.
0 - RTO interrupt will not occur.
1 - RTO interrupt will occur whenever RTO bit is set.

FLAG: Minimum number of opening and closing flags for HDLC TX. The minimum number of flags between packets is this 4-bit value plus one. Hence, 0000b forces at least one opening flag and one closing flag for each packet, and 1111b forces at least 16 opening and closing flags. The closing flags of one packet may also be the opening flags of the next one if the transmit line does not go idle in between. Note that HDLC RX does not count flags; only one is necessary (or three in Manchester mode).

CRCN: CRC polarity control.
0 - CRC transmitted not inverted.
1 - CRC transmitted inverted.

CRCApd: CRC pass through.
0 - Do not pass received CRC to CPU.
1 - Pass received CRC to CPU.

IDLE: Idle mode.
0 - Idle-in Mark mode - When HDLC is idle (not transmitting starting/stop flags or packets), hold the transmit data pin high.
1 - Idle-in Flag mode - When HDLC is idle, transmit continuous flags.

AME: Address Match Enable. Activates address matching on received frames.
00 - No address matching
01 - 4 x 1 byte matching
10 - 2 x 2 byte matching
11 - Undefined, no matching

IDLSpc: Idle in space
0 - TX idle in mark (normal)
1 - TX idle in space
RX will receive Manchester encoded data whether it idles in mark or space.

Copyright 2007 Cirrus Logic

CRCZ: CRC zero seed
0 - Seed CRC calculations with all ones; that is, 0xFFFF for 16 bit words and 0xFFFF_FFFF for 32 bit words.
1 - Seed CRC calculations with all zeros.
Applies to both RX and TX.

RXE: HDLC Receive Enable.
0 - Disable HDLC RX. If UART is still enabled, UART may still receive normally.
1 - Enable HDLC RX.

TXE: HDLC Transmit Enable.
0 - Disable HDLC TX. If UART is still enabled, UART may still transmit normally.
1 - Enable HDLC TX.

TUS: Transmit FIFO Underrun Select
0 - TX FIFO underrun causes CRC (if enabled) and stop flag to be transmitted.
1 - TX FIFO underrun causes abort (escape-flag) to be transmitted.

CRCE: CRC enable.
0 - No CRC is generated by TX or expected by RX.
1 - HDLC TX automatically generates and sends a CRC at the end of a packet, and HDLC RX expects a CRC at the end of a packet.

CRCS: CRC size.
0 - CRC-CCITT (16 bits): $x^{16} + x^{12} + x^5 + 1$
1 - CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
If inverted (see CRCN bit) the CRC-16 check value is 0x1D0F and the CRC-32 check value is 0xC704_DD7B. Otherwise the check value is zero.

UART1HDLCAddMtchVal

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AMV															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AMV															

Address:

0x808C_0210 - Read/Write

Copyright 2007 Cirrus Logic

Default:

0x0000_0000

Definition:

HDLC Address Match Value

Bit Descriptions:

AMV: Address match value. Supports 8-bit and 16-bit address matching. If UART1HDLCCtrl.AME[1:0] is 00b or 11b, this register is not used.

UART1HDLCAddMask

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AMSK															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AMSK															

Address:

0x808C_0214 - Read/Write

Default:

0x0000_0000

Definition:

HDLC Address Mask

Bit Descriptions:

AMSK: Address mask value. Supports 8-bit and 16-bit address masking. If UART1HDLCCtrl.AME[1:0] is 00b or 11b, this register is not used.

UART1HDLCRXInfoBuf

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD														BPLLE	RSVD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	BC											BFRE	BROR	BCRE	BRAB

Address:

0x808C_0218 - Read Only

Default:

0x0000_0000

Copyright 2007 Cirrus Logic

Definition:

HDLC Receive Information Buffer Register. This register is loaded when the last data byte in a received frame is read from the receive FIFO. The CPU has until the end of the next frame to read this register, or the RIL bit in the HDLC Status Register is set.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

BPLLE: Buffered Digital PLL Error.
1 - Receiver aborted last frame because DPLL lost the carrier.
0 - Receiver did not abort because DPLL lost the carrier.
This bit is only valid when receiving Manchester-encoded synchronous HDLC.

BC: Received frame Byte Count.
The total number of valid bytes read from the RX FIFO during the last HDLC frame.

BFRE: Buffered Framing Error.
0 - No framing errors were encountered in the last frame.
1 - A framing error occurred during the last frame, causing the remainder of the frame to be discarded.

BROR: Buffered Receiver Over Run.
0 - The RX buffer did not overrun during the last frame.
1 - The receive FIFO did overrun during the last frame. The remainder of the frame was discarded.

BCRE: Buffered CRC Error.
0 - No CRC check errors occurred in the last frame.
1 - The CRC calculated on the incoming data did not match the CRC value contained in the last frame.

BRAB: Buffered Receiver Abort.
0 - No abort occurred in the last frame.
1 - The last frame was aborted.

UART1HDLCSts

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD														PLLE	PLLCC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LNKIDL	CRE	ROR	TBY	RIF	RSVD	RAB	RTO	EOF	RFL	RIL	RFC	RFS	TAB	TFC	TFS

Copyright 2007 Cirrus Logic

Address:

0x808C_021C - Read/Write

Default:

0x0000_0000

Definition:

HDLC Status Register. The TFS and RFS bits in this register are replicas of bits in the UART status register.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

PLLE: Digital PLL Error. (Read Only)
1 - A frame receive was aborted because the DPLL lost synchronization with the carrier.
0 - DPLL has not lost carrier during frame reception.
This bit is only valid when set up to receive Manchester-encoded synchronous HDLC.

Note: This bit reflects the status associated with the last character read from the RX FIFO. It changes with reads from the RX FIFO.

PLLCC: Digital PLL Carrier Sense. (Read Only)
1 - DPLL tacked onto a carrier.
0 - DPLL does not sense a carrier.

LNKIDL: Link Idle. (Read Only)
0 - RX data signal has changed within two bit periods
1 - RX data signal has not changed within two bit periods.
This bit is only valid when set up to receive Manchester-encoded synchronous HDLC.

CRE: CRC Error. (Read Only)
0 - No CRC check errors encountered in incoming frame.
1 - CRC calculated on the incoming data does not match CRC value contained within the received frame. This bit is set with the last data in the incoming frame along with EOF.

Note: This bit reflects the status associated with the last character read from the RX FIFO. It changes with reads from the RX FIFO.

ROR: Receive FIFO Overrun. (Read Only)
0 - RX FIFO has not overrun.
1 - RX logic attempted to place data in the RX FIFO while it was full. The most recently read data is the last valid data before the overrun. The rest of the incoming frame is dropped. EOF is also set.

Note: This bit reflects the status associated with the last character read from the RX FIFO. It changes with reads from the RX FIFO.

Copyright 2007 Cirrus Logic

TBY: Transmitter Busy. (Read Only)
0 - TX is idle, disabled, or transmitting an abort.
1 - TX is currently sending a frame (address, control, data, CRC or start/stop flag).

RIF: Receiver In Frame. (Read Only)
0 - RX is idle, disabled, or receiving start flags.
1 - RX is receiving a frame.

RAB: Receiver Abort. (Read Only)
0 - No abort has been detected for the incoming frame.
1 - Abort detected during receipt of incoming frame. The most recently read data is the last valid data before the abort. EOF is also set.

Note: This bit reflects the status associated with the last character read from the RX FIFO. It changes with reads from the RX FIFO.

RTO: Receiver Time Out.
Set to "1" whenever the HDLC RX has received four consecutive flags, or four character times of idle or space. Cleared by writing a "1" to this bit.

EOF: End of Frame (read only).
0 - Current frame has not been received completely.
1 - The data most recently read from the RX FIFO is the last byte of data within the frame.

Note: This bit reflects the status associated with the last character read from the RX FIFO. It changes with reads from the RX FIFO.

RFL: Receive Frame Lost. (Read/Write)
Set to "1" when an ROR occurred at the start of a new frame, before any data for the frame could be put into the RX FIFO. Cleared by writing a "1" to this bit.

RIL: Receive Information buffer Lost. (Read/Write)
Set to "1" when the last data for a frame is read from the RX FIFO and the UART1HDLCRXInfoBuf has not been read since the last data of the previous frame was read. That is, the information loaded into the UART1HDLCRXInfoBuf about the previous frame was never read and has been overwritten. Cleared by writing a "1" to this bit.

RFC: Received Frame Complete. (Read/Write)
Set to "1" when the last data byte for the frame is read from the RX FIFO (this also triggers an update of the UART1HDLCRXInfoBuf). Cleared by writing to a "1" to this bit.

Copyright 2007 Cirrus Logic

RFS: Receive FIFO Service request. (Read Only)
This bit is a copy of the RIS bit in the UART interrupt identification register.
0 - RX FIFO is empty or RX is disabled.
1 - RX FIFO not empty and RX enabled.
May generate an interrupt and signal a DMA service request.

TAB: Transmitted Frame Aborted. (Read/Write)
Set "1" when a transmitted frame is terminated with an abort. Cleared by writing to a "1" to this bit.

TFC: Transmit Frame Complete. (Read/Write)
Set to "1" whenever a transmitted frame completes, whether terminated normally or aborted. Cleared by writing to a "1" to this bit.

TFS: Transmit FIFO Service request. (Read Only)
This bit is a copy of the TIS bit in the UART interrupt identification register.
0 - TX FIFO is full or TX disabled.
1 - TX FIFO not full and TX enabled. May generate an interrupt and signal a DMA service request.

Copyright 2007 Cirrus Logic

14

Copyright 2007 Cirrus Logic

Chapter 15
UART2

15

15.1 Introduction

UART2 implements a UART interface identical to that of UART1. UART2 does *not* implement a modem or HDLC interface. For additional details about UART1, refer to Chapter 14, "UART1 With HDLC and Modem Control Signals" on page 14-1.

UART2 and the IrDA blocks cooperatively implement a Slow Infrared (SIR) interface. The register interface for each block is separate. The UART2 control registers are at base address 0x808D_0000 and the IrDA controller registers are at base address 0x808B_0000. For additional details about IrDA, refer to Chapter 17, "IrDA" on page 17-1. The UART SIR interface is described below.

15.2 IrDA SIR Block

The IrDA SIR block contains an IrDA SIR protocol Encoder/decoder. The SIR protocol Encoder/decoder can be enabled for serial communication via signals **nSIROUT** and **SIRIN** to an infrared transducer instead of using the UART signals **UARTTXD** and **UARTRXD**.

If the SIR protocol Encoder/decoder is enabled, the **UARTTXD** line is held in the passive state (HIGH) and transitions of the modem status or the **UARTRXD** line will have no effect. The SIR protocol Encoder/decoder can both receive and transmit, but it is half-duplex only, so it cannot receive while transmitting, or vice versa.

The IrDA SIR physical layer specifies a minimum 10 ms delay between transmission and reception.

15.2.1 IrDA SIR Encoder/decoder Functional Description

The IrDA SIR Encoder/decoder comprises:

- IrDA SIR transmit encoder
- IrDA SIR receive decoder

This is shown in Figure 15-1:

Copyright 2007 Cirrus Logic

15

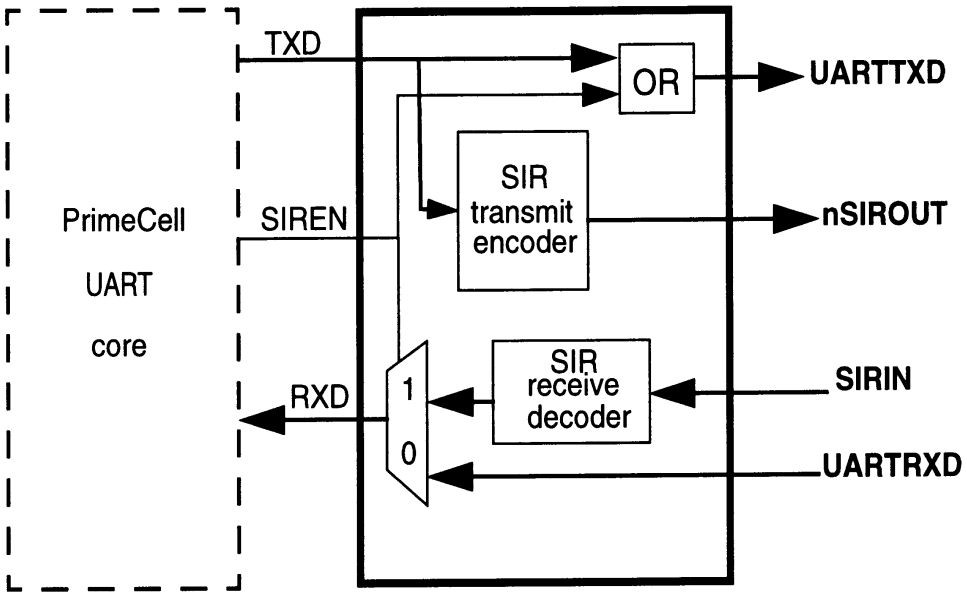


Figure 15-1. IrDA SIR Encoder/decoder Block Diagram

15.2.1.1 IrDA SIR Transmit Encoder

The SIR transmit encoder modulates the Non Return-to-Zero (NRZ) transmit bit stream output from the UART. The IrDA SIR physical layer specifies use of a Return To Zero, Inverted (RZI) modulation scheme which represents logic 0 as an infrared light pulse. The modulated output pulse stream is transmitted to an external output driver and infrared Light Emitting Diode (LED).

In normal mode, the transmitted pulse width is specified as three times the period of the internal x16 clock (Baud16), that is, 3/16 of a bit period.

In low-power mode, the transmit pulse width is specified as 3/16 of a 115.2 Kbps bit period. This is implemented as three times the period of a nominal 1.8432 MHz clock (IrLPBaud16) derived by dividing down the UARTCLK clock. The frequency of IrLPBaud16 is set up by writing the appropriate divisor value to UARTILPR. The active low encoder output is normally LOW for the marking state (no light pulse). The encoder outputs a high pulse to generate a infrared light pulse representing a logic "0" or spacing state.

15.2.1.2 IrDA SIR Receive Decoder

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the UART received data input. The decoder input is normally HIGH (marking state) in the idle state (the transmit encoder output has the opposite polarity to the decoder input).

Copyright 2007 Cirrus Logic

A start bit is detected when the decoder input is LOW.

Regardless of being in normal or low-power mode, a start bit is deemed valid if the decoder is still LOW, one period of IrLPBaud16 after the LOW was first detected. This allows a normal-mode UART to receive data from a low-power mode UART, which may transmit pulses as small as 1.41 µsec.

15.2.2 IrDA SIR Operation

The IrDA SIR Encoder/decoder provides functionality which converts between an asynchronous UART data stream and half-duplex serial SIR interface. No analog processing is performed on-chip. The role of the SIR encoder/decoder is only to provide a digital encoded output and decoded input to the UART. There are two modes of operation:

- In normal IrDA mode, a zero logic level is transmitted as high pulse of 3/16th duration of the selected baud rate bit period on the **nSIROUT** signal, while logic one levels are transmitted as a static LOW signal. These levels control the driver of an infrared transmitter, sending a pulse of light for each zero. On the reception side, the incoming light pulses energize the photo transistor base of the receiver, pulling its output LOW. This then drives the **SIRIN** signal LOW.
- In low-power IrDA mode, the width of the transmitted infrared pulse is set to 3 times the period of the internally generated IrLPBaud16 signal (1.63 ns assuming a nominal 1.8432MHz frequency) by changing the appropriate bit in UARTCR.

In both normal and low-power IrDA modes, during transmission, the UART data bit is used as the base for encoding, while during reception the decoded bits are transferred to the UART receive logic.

The IrDA SIR physical layer specifies a half duplex communication link with a minimum 10ms delay between transmission and reception. This delay must be generated by software since it is not supported by the UART. The delay is required since the Infrared receiver electronics may become biased or even saturated from the optical power coupled from the adjacent transmitter LED. This delay is known as latency or receiver setup time. Shorter delays may be able to be used when the link first starts up.

The IrLPBaud16 signal is generated by dividing down the UARTCLK signal according to the low-power divisor value written to UARTILPR.

The low-power divisor value is calculated as:

$$\text{Low-power divisor} = (\text{FUARTCLK} / \text{FirLPBaud16}) - 1$$

where FirLPBaud16 is nominally 1.8432 MHz.

The divisor must be chosen so that 1.42 MHz < IrLPBaud16 < 2.12 MHz.

Copyright 2007 Cirrus Logic

15.2.2.1 System/diagnostic Loopback Testing

It is possible to perform loopback testing for SIR data by setting the Loop Back Enable (LBE) bit to 1 in the control register UARTCR (bit 7), and setting the SIRTEST bit to 1 in the test register UARTTMR (bit 1).

Data transmitted on nSIROUT will be received on the SIRIN input.

Note: UART2TMR is the only occasion that a test register needs to be accessed during normal operation.

15

15.2.3 IrDA Data Modulation

The effect of IrDA 3/16 data modulation can be seen in Figure 15-2.

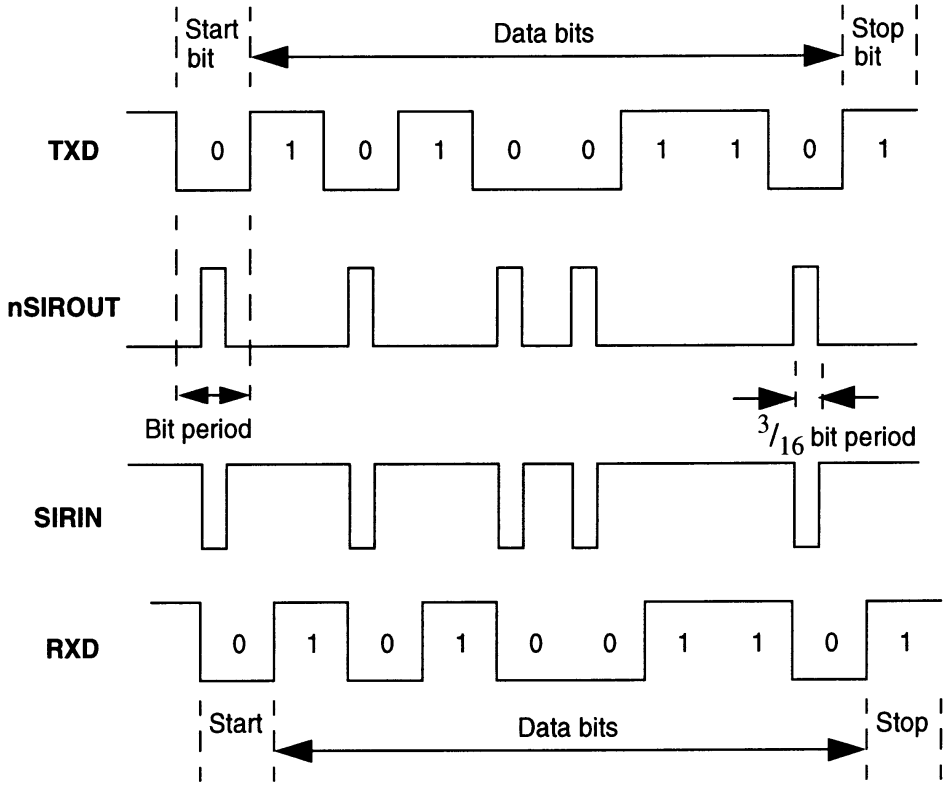


Figure 15-2. IrDA Data Modulation (3/16)

Copyright 2007 Cirrus Logic

15.2.4 Enabling Infrared (Ir) Modes

Table 15-1. UART2 / IrDA Modes

Mode	DeviceCfg Register		UART2Ctrl Register		IrEnable Register	
	U2EN	IonU2	SirEn	UARTE	EN[1]	EN[0]
Disabled	0	x	0	0	0	0
UART2	1	0	0	1	0	0
SIR	1	1	1	1	0	1
MIR	x	1	0	0	1	0
FIR	x	1	0	0	1	1

15.3 UART2 Package Dependency

UART2 uses package pins **RXD1** and **TXD1**. Pin **RXD1** drives both the UART2 UART input and the UART2 SIR input.

However, Syscon register DeviceCfg[28] (IonU2) controls what drives pin **TXD1**. See Table 15-2.

Table 15-2. IonU2 Pin Function

IonU2	Pin TXD1 Function
0	UART2 UART is the output signal
1	Logical OR of IrDA output signal and UART2 SIR output signal

Therefore, to use any IrDA mode, FIR, MIR or SIR, set IonU2. To use UART2 as a UART, clear IonU2.

15.3.1 Clocking Requirements

There are two clocks, PCLK and UARTCLK.

UARTCLK frequency must accommodate the desired range of baud rates:

$$Fuartclk(min) >= 32 \times baud_rate(max)$$

$$Fuartclk(max) <= 32 \times 65{,}536 \times baud_rate(min)$$

The frequency of UARTCLK must also be within the required error limits for all baud rates to be used.

To allow sufficient time to write the received data to the receive FIFO, UARTCLK must be less than or equal to four times the frequency of PCLK:

Copyright 2007 Cirrus Logic

Fuartclk <= 4 x Fpclk

If the IrDA SIR functionality is required, UARTCLK must have a frequency between 2.7 MHz and 542.7 MHz to ensure that the low-power mode transmit pulse duration complies with the IrDA SIR specification.

15.3.2 Bus Bandwidth Requirements

There are two basic ways of moving data to and from the UART FIFOs:

- Direct DMA interface - this permits byte-wide access to the UART without using the APB. The DMA block will pack/unpack individual bytes so that it reads or writes full 32-bit words rather than individual bytes.
- Accessing the UART via the APB - this requires APB/AHB bus bandwidth. Then, both a read and write are required for each 8-bit data byte.

Bandwidth requirements also depend on the selected baud rate, character size, parity selection, number of stop bits, and spacing between characters (if receiving).

For example, assume 115,200 baud, 8-bit characters, even parity, one stop bit, no space between characters. There are 11 bits per character, so 115,200 / 11 = 10473 characters per second. If both transmitting and receiving, 20,945 characters per second pass through the UART. Accessing the UART through the DMA interface requires one access per 32 bits, implying only 20,945 / 4 = 5,236 AHB accesses per second. Accessing the UART through the APB requires two accesses per byte, implying 20,945 APB bus accesses.

As another example, assume 230,400 baud (the maximum with a UARTCLK equal to 7.3728 Mhz), 5-bit characters, no parity, one stop bit, and no space between characters. There are 7 bits per character, so 230400 / 7 = 32,914 characters per second. Simultaneous transmitting and receiving implies 65829 APB characters per second. Using the DMA interface would result in 16457 AHB accesses per second, while using the APB to access the UART leads to 65829 bus accesses per second.

Copyright 2007 Cirrus Logic

15.4 Registers

Register Descriptions

UART2Data

Address:
0x808D_0000 - Read/Write

Default:
0x0000_0000

Definition:
UART Data Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

DATA: UART Data, read for receive data, write for transmit data
For words to be transmitted:
• if the FIFOs are enabled, data written to this location is pushed onto the transmit FIFO
• if the FIFOs are not enabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO). The write operation initiates transmission from the UART. The data is prefixed with a start bit, appended with the appropriate parity bit (if parity is enabled), and a stop bit. The resultant word is then transmitted.
For received words:
• if the FIFOs are enabled, the data byte is extracted, and a 3-bit status (break, frame and parity) is pushed onto the 11-bit wide receive FIFO
• if the FIFOs are not enabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO).

Copyright 2007 Cirrus Logic

UART2RXSts

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												OE	BE	PE	FE

15

Address:

0x808D_0004 - Read/Write

Default:

0x0000_0000

Definition:

UART Receive Status Register and Error Clear Register. Provides receive status of the data value last read from the UART2Data. A write to this register clears the framing, parity, break and overrun errors. The data value is not important.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

OE: Overrun Error. This bit is set to "1" if data is received and the FIFO is already full. This bit is cleared to 0 by a write to UART2RXSts. The FIFO contents remain valid since no further data is written when the FIFO is full, only the contents of the shift register are overwritten. The CPU must now read the data in order to empty the FIFO.

BE: Break Error. This bit is set to "1" if a break condition was detected, indicating that the received data input was held LOW for longer than a full-word transmission time (defined as start, data, parity and stop bits). This bit is cleared to 0 after a write to UART2RXSts. In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a "1" (marking state) and the next valid start bit is received.

PE: Parity Error. When this bit is set to "1", it indicates that the parity of the received data character does not match the parity selected in UART2LinCtrlHigh (bit 2). This bit is cleared to 0 by a write to UART2RXSts. In FIFO mode, this error is associated with the character at the top of the FIFO.

Copyright 2007 Cirrus Logic

FE: Framing Error. When this bit is set to "1", it indicates that the received character did not have a valid stop bit (a valid stop bit is "1"). This bit is cleared to 0 by a write to UART2RXSts. In FIFO mode, this error is associated with the character at the top of the FIFO.

UART2LinCtrlHigh

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									WLEN		FEN	STP2	EPS	PEN	BRK

Address:

0x808D_0008 - Read/Write

Default:

0x0000_0000

Definition:

UART - High. UART2LinCtrlHigh, UART2LinCtrlMid and UART2LinCtrlLow form a single 23-bit wide register (UART2LinCtrl) which is updated on a single write strobe generated by an UART2LinCtrlHigh write. So, in order to internally update the contents of UART2LinCtrlMid or UART2LinCtrlLow, a UART2LinCtrlHigh write must always be performed at the end.

To update the three registers there are two possible sequences:
• UART2LinCtrlLow write, UART2LinCtrlMid write and UART2LinCtrlHigh write
• UART2LinCtrlMid write, UART2LinCtrlLow write and UART2LinCtrlHigh write.

To update UART2LinCtrlLow or UART2LinCtrlMid only:
• UART2LinCtrlLow write (or UART2LinCtrlMid write) and UART2LinCtrlHigh write.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

WLEN: Number of bits per frame:
11 = 8 bits
10 = 7 bits
01 = 6 bits
00 = 5 bits

Copyright 2007 Cirrus Logic

15

FEN: FIFO Enable.
1 - Transmit and receive FIFO buffers are enabled (FIFO mode).
0 - The FIFOs are disabled (character mode). (That is, the FIFOs become 1-byte-deep holding registers.)

STP2: Two Stop Bits Select.
1 - Two stop bits are transmitted at the end of the frame.
0 - One stop bit is transmitted at the end of the frame.
The receive logic does not check for two stop bits being received.

EPS: Even Parity Select.
1 - Even parity generation and checking is performed during transmission and reception (this checks for an even number of "1"s in data and parity bits).
0 - Odd parity is performed (this checks for an odd number of "1"s).
This bit has no effect when parity is disabled by Parity Enable (bit 1) being cleared to 0.

PEN: Parity Enable.
1 - Parity checking and generation is enabled,
0 - Parity checking is disabled and no parity bit added to the data frame.

BRK: Send Break.
1 - A low level is continually output on the **UARTTXD** output, after completing transmission of the current character. This bit must be asserted for at least one complete frame transmission time in order to generate a break condition. The transmit FIFO contents remain unaffected during a break condition.
0 - For normal use, this bit must be cleared.

UART2LinCtrlMid

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								BR							

Address: 0x808D_000C - Read/Write

Default: 0x0000_0000

Copyright 2007 Cirrus Logic

Definition:

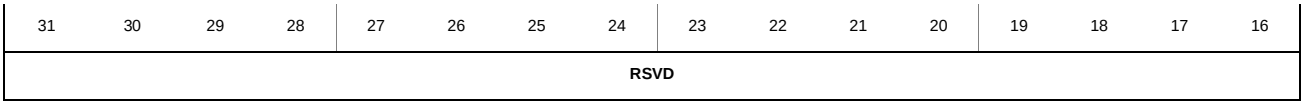
UART Line Control Register Middle.

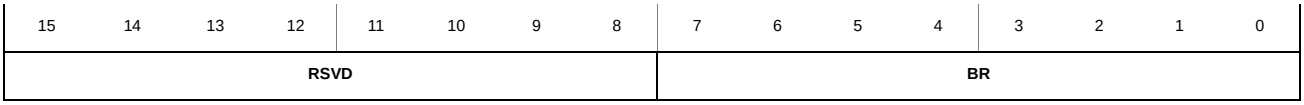
Bit Descriptions:

RSVD: Reserved. Unknown During Read.

BR: Baud Rate Divisor bits [15:8]. Most significant byte of baud rate divisor. These bits are cleared to 0 on reset.

UART2LinCtrlLow





Address:

0x808D_0010 - Read/Write

Default:

0x0000_0000

Definition:

UART Line Control Register Low.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

BR: Baud Rate Divisor bits [7:0]. Least significant byte of baud rate divisor. These bits are cleared to 0 on reset. The baud rate divisor is calculated as follows:

Baud rate divisor BAUDDIV = ($F_{UARTCLK}$ / (16 * Baud rate)) –1

where $F_{UARTCLK}$ is the UART reference clock frequency. A baud rate divisor of zero is not allowed and will result in no data transfer.

Copyright 2007 Cirrus Logic

UART2Ctrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								LBE	RTIE	TIE	RIE	MSIE	SIRLP	SIREN	UARTE

Address:

0x808D_0014 - Read/Write

Default:

0x0000_0000

Definition:

UART Control Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

LBE: Loopback Enable, for SIR and UART only.
1 - If the SIR Enable bit is also set to "1", and register UART2TMR bit 1 (SIRTEST) is set to "1", the SIR output path is inverted and fed through to the SIR input path. The SIRTEST bit in the test register must be set to "1" to override the normal half-duplex SIR operation. This should be the requirement for accessing the test registers during normal operation, and SIRTEST must be cleared to "0" when loopback testing is finished. This feature reduces the amount of external coupling required during system test.
0 - This bit is cleared to "0" on reset, which disables the loopback mode.

RTIE: Receive Timeout Enable. If this bit is set to "1", the receive timeout interrupt is enabled.

TIE: Transmit Interrupt Enable. If this bit is set to "1", the transmit interrupt is enabled.

RIE: Receive Interrupt Enable. If this bit is set to "1", the receive interrupt is enabled.

MSIE: Modem Status Interrupt Enable. If this bit is set to "1", the modem status interrupt is enabled.

Copyright 2007 Cirrus Logic

15

SIRLP: SIR Low Power Mode. This bit selects the IrDA encoding mode. If this bit is cleared to 0, low level bits are transmitted as an active high pulse with a width of 3/16th of the bit period. If this bit is set to "1", low level bits are transmitted with a pulse width which is 3 times the period of the IrLPBaud16 input signal, regardless of the selected bit rate. Setting this bit uses less power, but may reduce transmission distances.

SIREN: SIR Enable. If this bit is set to "1", the IrDA SIR encoder/decoder is enabled. This bit has no effect if the UART is not enabled by bit 0 being set to "1". When the IrDA SIR encoder/decoder is enabled, data is transmitted and received on **nSIROUT** and **SIRIN**. **UARTTXD** remains in the marking state (set to "1"). Signal transitions on **UARTRXD** or modem status inputs will have no effect. When the IrDA SIR encoder/decoder is disabled, **nSIROUT** remains cleared to 0 (no light pulse generated), and signal transitions on **SIRIN** will have no effect.

UARTE: UART Enable. If this bit is set to "1", the UART is enabled. Data transmission and reception occurs for UART signals.

UART2Flag

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								TXFE	RXFF	TXFF	RXFE	BUSY	DCD	DSR	CTS

Address:
0x808D_0018 - Read/Write

Default:
0x0000_0000

Definition:
UART Flag Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

TXFE: Transmit FIFO Empty. The meaning of this bit depends on the state of the FEN bit in the UART2LinCtrlHigh register. If the FIFO is disabled, this bit is set when the transmit holding register is empty. If the FIFO is enabled, the TXFE bit is set when the transmit FIFO is empty.

Copyright 2007 Cirrus Logic

15

RXFF: Receive FIFO Full. The meaning of this bit depends on the state of the FEN bit in the UART2LinCtrlHigh register. If the FIFO is disabled, this bit is set when the receive holding register is full. If the FIFO is enabled, the RXFF bit is set when the receive FIFO is full.

TXFF: Transmit FIFO Full. The meaning of this bit depends on the state of the FEN bit in the UART2LinCtrlHigh register. If the FIFO is disabled, this bit is set when the transmit holding register is full. If the FIFO is enabled, the TXFF bit is set when the transmit FIFO is full.

RXFE: Receive FIFO Empty. The meaning of this bit depends on the state of the FEN bit in the UART2LinCtrlHigh register. If the FIFO is disabled, this bit is set when the receive holding register is empty. If the FIFO is enabled, the RXFE bit is set when the receive FIFO is empty.

BUSY: UART Busy. If this bit is set to "1", the UART is busy transmitting data. This bit remains set until the complete byte, including all the stop bits, has been sent from the shift register. This bit is set as soon as the transmit FIFO becomes non-empty (regardless of whether the UART is enabled or not).

DCD: Data Carrier Detect status. This bit is the complement of the UART data carrier detect (**nUARTDCD**) modem status input. That is, the bit is "1" when the modem status input is 0.

DSR: Data Set Ready status. This bit is the complement of the UART data set ready (**nUARTDSR**) modem status input. That is, the bit is "1" when the modem status input is 0.

CTS: Clear To Send status. This bit is the complement of the UART clear to send (**nUARTCTS**) modem status input. That is, the bit is "1" when the modem status input is 0.

UART2IntIDIntClr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												RTIS	TIS	RIS	MIS

Address:

0x808D_001C

Copyright 2007 Cirrus Logic

Default:

0x0000_0000

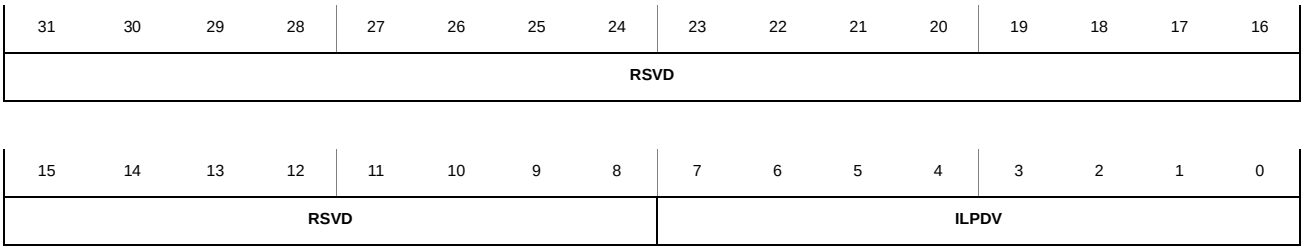
Definition:

UART Interrupt Identification and Interrupt Clear Register. Interrupt status is read from UART2IntIDIntClr. A write to UART2IntIDIntClr clears the modem status interrupt. All the bits are cleared to 0 when reset.

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
RTIS:	Receive Timeout Interrupt Status. This bit is set to "1" if the receive timeout interrupt is asserted.
TIS:	Transmit Interrupt Status. This bit is set to "1" if the transmit interrupt is asserted.
RIS:	Receive Interrupt Status. This bit is set to "1" if the receive interrupt is asserted.
MIS:	Modem Interrupt Status. This bit is set to "1" if the modem status interrupt is asserted.

UART2IrLowPwrCntr



Address:

0x808D_0020 - Read/Write

Default:

0x0000_0000

Definition:

UART IrDA Low Power Divisor Register. This is an 8-bit read/write register that stores the low-power counter divisor value used to generate the **IrLPBaud16** signal by dividing down of UARTCLK. All the bits are cleared to 0 when reset.

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.

Copyright 2007 Cirrus Logic

15

ILPDV: IrDA Low Power Divisor bits [7:0]. 8-bit low-power divisor value. These bits are cleared to 0 at reset. The divisor must be chosen so that the relationship 1.42 MHz < **IrLPBaud16** < 2.12 MHz is maintained, which results in a low power pulse duration of 1.41–2.11 µs (three times the period of **IrLPBaud16**). The minimum frequency of **IrLPBaud16** ensures that pulses less than one period of **IrLPBaud16** are rejected, but that pulses greater than 1.4 µs are accepted as valid pulses. **Zero is an illegal value. Programming a zero value will result in no IrLPBaud16 pulses being generated.**

UART2DMACtrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD													DMAERR	TXDMAE	RXDMAE

Address:
0x808D_0028 - Read/Write

Default:
0x0000_0000

Definition:
UART DMA Control Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

DMAERR: RX DMA error handing enable. If 0, the RX DMA interface ignores error conditions in the UART receive section. If "1", the DMA interface stops and notifies the DMA block when an error occurs. Errors include break errors, parity errors, and framing errors.

TXDMAE: TX DMA interface enable. Setting to "1" enables the private DMA interface to the transmit FIFO.

RXDMAE: RX DMA interface enable. Setting to "1" enables the private DMA interface to the receive FIFO.

Copyright 2007 Cirrus Logic

UART2TMR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								0						SIRTEST	0

Address:

0x808D_0084 - Read/Write

Default:

0x0000_0000

Definition:

UART SIR Loopback Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

0: Must be written as "0". Unknown During Read.

SIRTEST: SIR test enable. Setting this bit to "1" enables the receive data path during IrDA transmission (testing requires SIR to be configured in full-duplex mode). This bit must be set to "1" to enable SIR system loopback testing, when the normal mode control register UART2Ctrl bit 7, Loop Back Enable (LBE), has been set to "1". Clearing this bit to 0 disabled the receive logic when the SIR is transmitting (normal operation). This bit defaults to 0 for normal (half-duplex) operation.

Copyright 2007 Cirrus Logic

15

.

Copyright 2007 Cirrus Logic

Chapter 16

UART3 With HDLC Encoder

16.1 Introduction

Note: This chapter applies only to the EP9307, EP9312, and EP9315 processors.

UART3 implements both a UART and an HDLC interface identical to that of UART1; it does not implement the modem interface. An additional output signal, **TENn**, is provided to support RS-485 operation by providing direction control of external data transceivers. The **OUT1** and **OUT2** signals in the MCR register define the **TENn** operating mode. **TENn** can be configured to assert whenever the UART transmit buffer has data to send, or to operate under software control.

For additional details about UART1, refer to Chapter 14, “UART1 With HDLC and Modem Control Signals” on page 14-1.

16.2 Implementation Details

16.2.1 UART3 Package Dependency

UART3 uses package pins **RXD2, TXD2** and **EGPIO[3]**. See Table 16-1 for details.

Table 16-1. UART3 Pin Functionality

PIN	Description
RXD2	UART2 input pin
TXD2	UART2 output pin
EGPIO[3]	HDLC clock or TENn

The use of **EGPIO[3]** is determined by several bits in Syscon register DeviceCfg. See Table 16-2 for details.

Copyright 2007 Cirrus Logic

16

Table 16-2. DeviceCfg Register Bit Functions

bit 26 TonG	bit 15 HC3IN	bit 14 HC3EN	bit 12 HC1EN	Function
x	x	0	x	External HDLC clock input is driven low.
x	0	1	0	External HDLC clock input is driven by EGPIO[3].
x	1	1	0	Internal HDLC clock output drives EGPIO[3].
1	0	0	0	TENn output drives EGPIO[3].

16.2.2 Clocking Requirements

There are two clocks, PCLK and UARTCLK.

UARTCLK frequency must accommodate the desired range of baud rates:

$$F_{UARTCLK}(min) >= 32 \times baud_rate(max)$$

$$F_{UARTCLK}(max) <= 32 \times 65536 \times baud_rate(min)$$

The frequency of UARTCLK must also be within the required error limits for all baud rates to be used.

To allow sufficient time to write the received data to the receive FIFO, UARTCLK must be less than or equal to four times the frequency of PCLK:

$$F_{UARTCLK} <= 4 \times Fpclk$$

16.2.3 Bus Bandwidth Requirements

There are two basic ways of moving data to and from the UART FIFOs:

- Direct DMA interface - this permits byte-wide access to the UART without using the APB. The DMA block will pack or unpack individual bytes so that it reads or writes full 32-bit words rather than individual bytes.
- Accessing the UART via the APB - this requires APB/AHB bus bandwidth. Then, both a read and write are required for each 8-bit data byte.

Bandwidth requirements also depend on the selected baud rate, character size, parity selection, number of stop bits, and spacing between characters (if receiving).

For example, assume 115,200 baud, 8-bit characters, even parity, one stop bit, no space between characters. There are 11 bits per character, so 115,200 / 11 = 10,473 characters per second. If both transmitting and receiving, 20,945 characters per second pass through the UART. Accessing the UART through the DMA interface requires one access per 32-bits, implying only 20,945 / 4 = 5,236 AHB accesses per second. Accessing the UART through the APB requires two accesses per byte, implying 20,945 APB buss accesses.

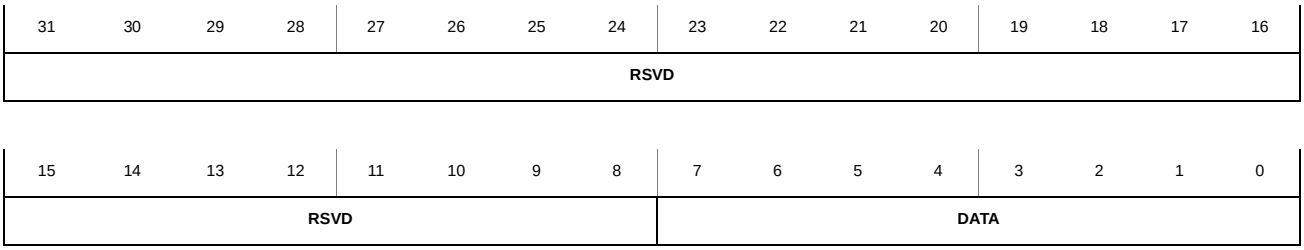
Copyright 2007 Cirrus Logic

As another example, assume 230,400 baud (the maximum with a UARTCLK equal to 7.3728 Mhz), 5-bit characters, no parity, one stop bit, and no space between characters. There are 7 bits per character, so 230,400 / 7 = 32,914 characters per second. Simultaneous transmitting and receiving implies 65,829 APB characters per second. Using the DMA interface would result in 16,457 AHB accesses per second, while using the APB to access the UART leads to 65,829 bus accesses per second.

16.3 Registers

Register Descriptions

UART3Data



Address:

0x808E_0000 - Read/Write

Default:

0x0000_0000

Definition:

UART3 Data Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

DATA: UART Data, read for receive data, write for transmit data

For words to be transmitted:
• if the FIFOs are enabled, data written to this location is pushed onto the transmit FIFO
• if the FIFOs are not enabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO). The write operation initiates transmission from the UART. The data is prefixed with a start bit, appended with the appropriate parity bit (if parity is enabled), and a stop bit. The resultant word is then transmitted.

For received words:
• if the FIFOs are enabled, the data byte is extracted, and

Copyright 2007 Cirrus Logic

a 3-bit status (break, frame and parity) is pushed onto the 11-bit wide receive FIFO
• if the FIFOs are not enabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO).

The received data byte is read by performing reads from the UART3Data register, while the corresponding status information can be read by a successive read of the UART3RXSts register.

UART3RXSts

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												OE	BE	PE	FE

Address:

0x808E_0004 - Read/Write

Default:

0x0000_0000

Definition:

UART3 Receive Status Register and Error Clear Register. Provides receive status of the data value last read from the UART3Data. A write to this register clears the framing, parity, break and overrun errors. The data value is not important.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

OE: Overrun Error.
1 - when data is received and the FIFO is already full.
0 - Cleared by a write to UART3RXSts.
The FIFO contents remain valid since no further data is written when the FIFO is full. Only the contents of the shift register are overwritten. The data must be read in order to empty the FIFO.

Copyright 2007 Cirrus Logic

BE: Break Error. This bit is set to 1 if a break condition was detected, indicating that the received data input was held LOW for longer than a full-word transmission time (defined as start, data, parity and stop bits). This bit is cleared to 0 after a write to UART3RXSts. In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a 1 (marking state) and the next valid start bit is received.

PE: Parity Error. When this bit is set to 1, it indicates that the parity of the received data character does not match the parity selected in UART3LinCtrlHigh (bit 2). This bit is cleared to 0 by a write to UART3RXSts. In FIFO mode, this error is associated with the character at the top of the FIFO.

FE: Framing Error. When this bit is set to 1, it indicates that the received character did not have a valid stop bit (a valid stop bit is 1). This bit is cleared to 0 by a write to UART3RXSts. In FIFO mode, this error is associated with the character at the top of the FIFO.

UART3LinCtrlHigh

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									WLEN		FEN	STP2	EPS	PEN	BRK

Address:

0x808E_0008 - Read/Write

Default:

0x0000_0000

Definition:

UART3 Line Control Register High. UART3LinCtrlHigh, UART3LinCtrlMid and UART3LinCtrlLow form a single 23-bit wide register (UART3LinCtrl) which is updated on a single write strobe generated by an UART3LinCtrlHigh write. So, in order to internally update the contents of UART3LinCtrlMid or UARTBLCR_L, a UART3LinCtrlHigh write must always be performed at the end.

To update the three registers there are two possible sequences:
• UART3LinCtrlLow write, UART3LinCtrlMid write and UART3LinCtrlHigh write

Copyright 2007 Cirrus Logic

- UART3LinCtrlMid write, UART3LinCtrlLow write and UART3LinCtrlHigh write.

To update UART3LinCtrlLow or UART3LinCtrlMid only:

- UART3LinCtrlLow write (or UART3LinCtrlMid write) and UART3LinCtrlHigh write.

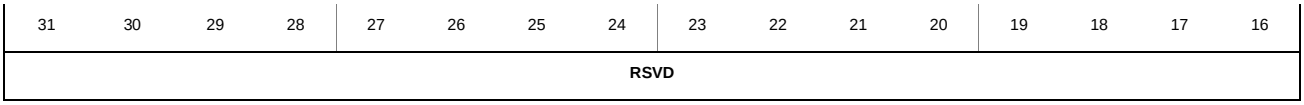
Bit Descriptions:

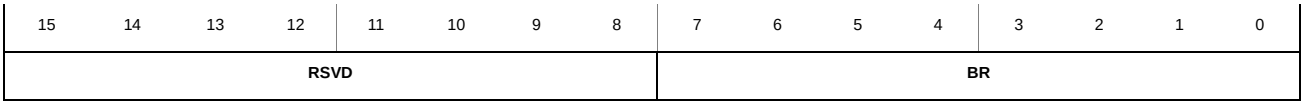
16

RSVD:	Reserved. Unknown During Read.
WLEN:	Number of bits per frame: 11 = 8 bits 10 = 7 bits 01 = 6 bits 00 = 5 bits
FEN:	FIFO Enable. 1 - Transmit and receive FIFO buffers are enabled (FIFO mode). 0 - The FIFOs are disabled (character mode). (That is, the FIFOs become 1-byte-deep holding registers.)
STP2:	Two Stop Bits Select. 1 - Two stop bits are transmitted at the end of the frame. 0 - One stop bit is transmitted at the end of the frame. The receive logic does not check for two stop bits being received.
EPS:	Even Parity Select. 1 - Even parity generation and checking is performed during transmission and reception, which checks for an even number of 1s in data and parity bits. 0 - Odd parity generation and checking is performed during transmission and reception, which checks for an odd number of 1s. This bit has no effect when parity is disabled by Parity Enable (bit 1) being cleared to 0.
PEN:	Parity Enable. 1 - Parity checking and generation is enabled 0 - Parity checking is disabled and no parity bit is added to the data frame.
BRK:	Send Break. 1 - A low level is continually output on the **UARTTXD** output, after completing transmission of the current character. This bit must be asserted for at least one complete frame transmission time in order to generate a break condition. The transmit FIFO contents remain unaffected during a break condition. 0 - For normal use, this bit must be cleared.

Copyright 2007 Cirrus Logic

UART3LinCtrlMid





Address:

0x808E_000C - Read/Write

Default:

0x0000_0000

Definition:

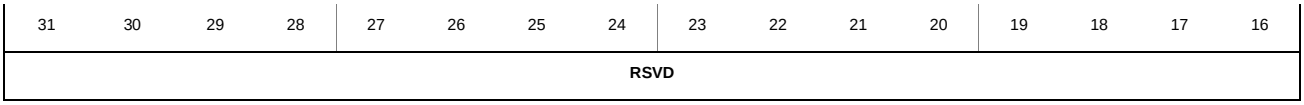
UART3 Line Control Register Middle

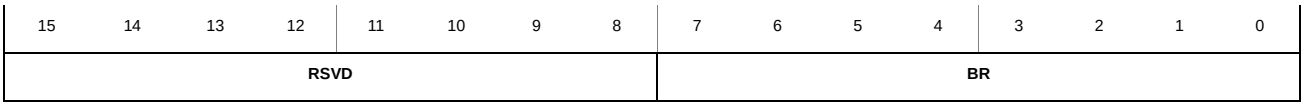
Bit Descriptions:

RSVD: Reserved. Unknown During Read.

BR: Baud Rate Divisor bits [15:8]. Most significant byte of baud rate divisor. These bits are cleared to 0 on reset.

UART3LinCtrlLow





Address:

0x808E_0010 - Read/Write

Default:

0x0000_0000

Definition:

UART3 Line Control Register Low.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

Copyright 2007 Cirrus Logic

BR: Baud Rate Divisor bits [7:0]. Least significant byte of baud rate divisor. These bits are cleared to 0 on reset. The baud rate divisor is calculated as follows:

Baud rate divisor BAUDDIV = ($F_{UARTCLK}$ / (16 * Baud rate)) –1

where $F_{UARTCLK}$ is the UART reference clock frequency. A baud rate divisor of zero is not allowed and will result in no data transfer.

UART3Ctrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								LBE	RTIE	TIE	RIE	MSIE	RSVD		UARTE

Address:

0x808E_0014 - Read/Write

Default:

0x0000_0000

Definition:

UART3 Control Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

LBE: Loopback Enable. If this bit is set to 1, data sent to TXD is received on RXD. This bit is cleared to 0 on reset, which disables the loopback mode.

RTIE: Receive Timeout Enable. If this bit is set to 1, the receive timeout interrupt is enabled.

TIE: Transmit Interrupt Enable. If this bit is set to 1, the transmit interrupt is enabled.

RIE: Receive Interrupt Enable. If this bit is set to 1, the receive interrupt is enabled.

MSIE: Modem Status Interrupt Enable. If this bit is set to 1, the modem status interrupt is enabled.

UARTE: UART Enable. If this bit is set to 1, the UART is enabled. Data transmission and reception occurs for UART signals.

Copyright 2007 Cirrus Logic

UART3Flag

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								TXFE	RXFF	TXFF	RXFE	BUSY	DCD	DSR	CTS

16

Address:

0x808E_0018 - Read/Write

Default:

0x0000_0000

Definition:

UART3 Flag Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

TXFE: Transmit FIFO Empty. The meaning of this bit depends on the state of the FEN bit in the UART3LinCtrlHigh register. If the FIFO is disabled, this bit is set when the transmit holding register is empty. If the FIFO is enabled, the TXFE bit is set when the transmit FIFO is empty.

RXFF: Receive FIFO Full. The meaning of this bit depends on the state of the FEN bit in the UART3LinCtrlHigh register. If the FIFO is disabled, this bit is set when the receive holding register is full. If the FIFO is enabled, the RXFF bit is set when the receive FIFO is full.

TXFF: Transmit FIFO Full. The meaning of this bit depends on the state of the FEN bit in the UART3LinCtrlHigh register. If the FIFO is disabled, this bit is set when the transmit holding register is full. If the FIFO is enabled, the TXFF bit is set when the transmit FIFO is full.

RXFE: Receive FIFO Empty. The meaning of this bit depends on the state of the FEN bit in the UART3LinCtrlHigh register. If the FIFO is disabled, this bit is set when the receive holding register is empty. If the FIFO is enabled, the RXFE bit is set when the receive FIFO is empty.

Copyright 2007 Cirrus Logic

BUSY: UART Busy. If this bit is set to 1, the UART is busy transmitting data. This bit remains set until the complete byte, including all the stop bits, has been sent from the shift register. This bit is set as soon as the transmit FIFO becomes non-empty, regardless of whether the UART is enabled or not.

DCD: Data Carrier Detect status. This bit is the complement of the UART data carrier detect (**nUARTDCD**) modem status input. That is, the bit is 1 when the modem status input is 0.

DSR: Data Set Ready status. This bit is the complement of the UART data set ready (**nUARTDSR**) modem status input. That is, the bit is 1 when the modem status input is 0.

CTS: Clear To Send status. This bit is the complement of the UART clear to send (**nUARTCTS**) modem status input. That is, the bit is 1 when the modem status input is 0.

UART3IntIDIntClr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												RTIS	TIS	RIS	MIS

Address:

0x808E_001C - Read/Write

Default:

0x0000_0000

Definition:

UART3 Interrupt Identification and Interrupt Clear Register. Interrupt status is read from UART3IntIDIntClr. A write to UART3IntIDIntClr clears the modem status interrupt. All the bits are cleared to "0" when reset.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RTIS: Receive Timeout Interrupt Status. This bit is set to 1 if the receive timeout interrupt is asserted. This bit is cleared when the receive FIFO is empty or the receive line goes active.

Copyright 2007 Cirrus Logic

TIS: Transmit Interrupt Status. This bit is set to 1 if the **UARTTXINTR** transmit interrupt is asserted, which occurs when the transmit FIFO is not full. It is set to 0 when the transmit FIFO is full.

RIS: Receive Interrupt Status. This bit is set to 1 if the **UARTRXINTR** receive interrupt is asserted, which occurs when the receive FIFO is not empty. It is set to 0 when the receive FIFIO is empty.

MIS: Modem Interrupt Status. This bit is set to 1 if the **UARTMSINTR** modem status interrupt is asserted. This bit is cleared by writing any value to this register.

UART3LowPwrCntr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															

Address:

0x808E_0020 - Read/Write

Default:

0x0000_0000

Definition:

UART3 IrDA Low Power Divisor Register. This register is present in UART3 but is not supported.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

UART3DMACtrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD													DMAERR	TXDMAE	RXDMAE

Address:

0x808E_0028 - Read/Write

Copyright 2007 Cirrus Logic

Default:

0x0000_0000

Definition:

UART3 DMA Control Register

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
DMAERR:	RX DMA error handing enable. If 0, the RX DMA interface ignores error conditions in the UART receive section. If 1, the DMA interface stops and notifies the DMA block when an error occurs. Errors include break errors, parity errors, and framing errors.
TXDMAE:	TX DMA interface enable. Setting to 1 enables the private DMA interface to the transmit FIFO.
RXDMAE:	RX DMA interface enable. Setting to 1 enables the private DMA interface to the receive FIFO.

UART3ModemCtrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								0			RSVD	OUT2	OUT1	RSVD	

Address:

0x808E_0100 - Read/Write

Default:

0x0000_0000

Definition:

Modem Control Register. Only the **OUT1** and **OUT2** bits have functionality in UART3. The RTS and DTR bits exist but have no function.

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
OUT2:	**OUT2** function. Controls the **TENn** output behavior: 1 = **TENn** is driven by the UART3Flag.BUSY status bit; that is, **TENn** is low whenever the UART has transmit data to send. 0 = **TENn** is controlled by the OUT1 bit.
OUT1:	**OUT1** function. When OUT2 = "0", then **TENn** = OUT1. Otherwise OUT1 is ignored.

Copyright 2007 Cirrus Logic

0: Must be written as "0".

UART3HDLCCtrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				CMAS	TXCM	RXCM	TXENC	RXENC	SYNC	TFCEN	TABEN	RFCEN	RILEN	RFLEN	RTOEN

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLAG				CRCN	CRCApd	IDLE	AME		RSVD		RXE	TXE	TUS	CRCE	CRCS

Address:

0x808E_020C - Read/Write

Default:

0x0000_0000

Definition:

HDLC Control Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

CMAS: Clock Master:
1 - Transmitter and/or receiver use 1x clock generated by the internal transmitter.
0 - Transmitter and/or receiver use 1x clock generated externally.

TXCM: Transmit Clock Mode.
1 - Generate 1x clock when in synchronous HDLC mode using NRZ encoding.
0 - Do not generate clock.
This bit has no effect unless TXENC is clear and synchronous HDLC is enabled.

RXCM: Receive Clock Mode.
1 - Use external 1x clock when in synchronous HDLC mode using NRZ encoding.
0 - Do not use external clock.
This bit has no effect unless RXENC is clear and synchronous HDLC is enabled.

TXENC: Transmit Encoding method.
1 - Use Manchester bit encoding.
0 - Use NRZ bit encoding.
This bit has no effect unless synchronous HDLC is enabled

Copyright 2007 Cirrus Logic

RXENC: Receive Encoding method.
1 - Use Manchester bit encoding.
0 - Use NRZ bit encoding.
This bit has no effect unless synchronous HDLC is enabled.

SYNC: Synchronous / Asynchronous HDLC Enable.
0 - Select asynchronous HDLC for TX and RX.
1 - Select synchronous HDLC for TX and RX.

TFCEN: Transmit Frame Complete Interrupt Enable.
0 - TFC interrupt will not occur.
1 - TFC interrupt will occur whenever TFC bit is set.

TABEN: Transmit Frame Abort Interrupt Enable.
0 - TAB interrupt will not occur.
1 - TAB interrupt will occur whenever TAB bit is set.

RFCEN: Receive Frame Complete Interrupt Enable.
0 - RFC interrupt will not occur.
1 - RFC interrupt will occur whenever RAB bit or EOF bit is set.

RILEN: Receive Information Lost Interrupt Enable.
0 - RIL interrupt will not occur.
1 - RIL interrupt will occur whenever RIL bit is set.

RFLEN: Receive Frame Lost Interrupt Enable.
0 - RFL interrupt will not occur.
1 - RFL interrupt will occur whenever RFL bit is set.

RTOEN: Receiver Time Out Interrupt Enable.
0 - RTO interrupt will not occur.
1 - RTO interrupt will occur whenever RTO bit is set.

FLAG: Minimum number of opening and closing flags for HDLC TX. The minimum number of flags between packets is this 4-bit value plus one. Hence, 0000b forces at least one opening flag and one closing flag for each packet, and 1111b forces at least 16 opening and closing flags. The closing flags of one packet may also be the opening flags of the next one if the transmit line does not go idle in between. Note that HDLC RX does not count flags; only one is necessary (or three in Manchester mode).

CRCN: CRC polarity control.
0 - CRC transmitted not-inverted.
1 - CRC transmitted inverted.

Copyright 2007 Cirrus Logic

CRCApd: CRC pass through.
0 - Do not pass received CRC to CPU.
1 - Pass received CRC to CPU.

IDLE: Idle mode.
0 - Idle-in Mark mode - When HDLC is idle (not transmitting start or stop flags or packets), hold the transmit data pin high.
1 - Idle-in Flag mode - When HDLC is idle, transmit continuous flags.

AME: Address Match Enable. Activates address matching on received frames.
00 - No address matching
01 - 4 x 1 byte matching
10 - 2 x 2 byte matching
11 - Undefined, no matching

RXE: HDLC Receive Enable.
0 - Disable HDLC RX. If UART is still enabled, UART may still receive normally.
1 - Enable HDLC RX.

TXE: HDLC Transmit Enable.
0 - Disable HDLC TX. If UART is still enabled, UART may still transmit normally.
1 - Enable HDLC TX.

TUS: Transmit FIFO Underrun Select
0 - TX FIFO underrun causes CRC (if enabled) and stop flag to be transmitted.
1 - TX FIFO underrun causes abort (escape-flag) to be transmitted.

CRCE: CRC enable.
0 - No CRC generated by HDLC TX or expected by HDLC RX.
1 - HDLC TX automatically generates and sends a CRC at the end of a packet, and HDLC RX expects a CRC at the end of a packet.

CRCS: CRC size.
0 - CRC-16: $x^{16} + x^{12} + x^5 + 1$
1 - CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

Copyright 2007 Cirrus Logic

UART3HDLCAddMtchVal

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AMV															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AMV															

16

Address:

0x808E_0210 - Read/Write

Default:

0x0000_0000

Definition:

HDLC Address Match Value.

Bit Descriptions:

AMV: Address match value. Supports 8-bit and 16-bit address matching. If UART3HDLCCtrl.AME is "00" or "11", this register is not used.

UART3HDLCAddMask

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AMSK															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AMSK															

Address:

0x808E_0214 - Read/Write

Default:

0x0000_0000

Definition:

HDLC Address Mask.

Bit Descriptions:

AMSK: Address mask value. Supports 8-bit and 16-bit address masking. If UART3HDLCCtrl.AME is "00" or "11", this register is not used.

Copyright 2007 Cirrus Logic

UART3HDLCRXInfoBuf

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	BC											BFRE	BROR	BCRE	BRAB

Address:

0x808E_0218 - Read/Write

Default:

0x0000_0000

Definition:

HDLC Receive Information Buffer Register. This register is loaded when the last data byte in a received frame is read from the receive FIFO. The CPU has until the end of the next frame to read this register, or the RIL bit in the HDLC Status Register will be set.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

BC: Received frame Byte Count.
The total number of valid bytes read from the RX FIFO during the last HDLC frame.

BFRE: Buffered Framing Error.
0 - No framing errors were encountered in the last frame.
1 - A framing error occurred during the last frame, causing the remainder of the frame to be discarded.

BROR: Buffered Receiver Over Run.
0 - The RX buffer did not overrun during the last frame.
1 - The receive FIFO did overrun during the last frame. The remainder of the frame was discarded.

BCRE: Buffered CRC Error.
0 - No CRC check errors occurred in the last frame.
1 - The CRC calculated on the incoming data did not match the CRC value contained in the last frame.

BRAB: Buffered Receiver Abort.
0 - No abort occurred in the last frame.
1 - The last frame was aborted.

Copyright 2007 Cirrus Logic

UART3HDLCSts

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	CRE	ROR	TBY	RIF	RSVD	RAB	RTO	EOF	RFL	RIL	RFC	RFS	TAB	TFC	TFS

Address:

0x808E_021C - Read/Write

Default:

0x0000_0000

Definition:

HDLC Status Register. The TFS and RFS bits in this register are replicas of bits in the UART3 status register.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

CRE: CRC Error. (Read Only)
0 - No CRC check errors encountered in incoming frame.
1 - CRC calculated on the incoming data does not match CRC value contained within the received frame. This bit is set with the last data in the incoming frame along with EOF.

Note: This bit reflects the status associated with the last character read from the RX FIFO. It changes with reads from the RX FIFO.

ROR: Receive FIFO Overrun. (Read Only)
0 - RX FIFO has not overrun.
1 - RX logic attempted to place data in the RX FIFO while it was full. The most recently read data is the last valid data before the overrun. The rest of the incoming frame is dropped. EOF is also set.

Note: This bit reflects the status associated with the last character read from the RX FIFO. It changes with reads from the RX FIFO.

TBY: Transmitter Busy. (Read Only)
0 - TX is idle, disabled, or transmitting an abort.
1 - TX is currently sending a frame (address, control, data, CRC or start/stop flag).

RIF: Receiver In Frame. (Read Only)
0 - RX is idle, disabled or receiving start flags
1 - RX is receiving a frame.

Copyright 2007 Cirrus Logic

RAB: Receiver Abort. (Read Only)
0 - No abort has been detected for the incoming frame.
1 - Abort detected during receipt of incoming frame. The most recently read data is the last valid data before the abort. EOF is also set.

Note: This bit reflects the status associated with the last character read from the RX FIFO. It changes with reads from the RX FIFO.

RTO: Receiver Time Out.
Set to "1" whenever the HDLC RX has received four consecutive flags, or four character times of idle or space. Cleared by writing a "1" to this bit.

EOF: End of Frame (read only).
0 - Current frame has not been received completely.
1 - The data most recently read from the RX FIFO is the last byte of data within the frame.

Note: This bit reflects the status associated with the last character read from the RX FIFO. It changes with reads from the RX FIFO.

RFL: Receive Frame Lost. (Read/Write)
Set to "1" when an ROR occurred at the start of a new frame, before any data for the frame could be put into the RX FIFO. Cleared by writing a "1" to this bit.

RIL: Receive Information buffer Lost. (Read/Write)
Set to "1" when the last data for a frame is read from the RX FIFO and the UART1HDLCRXInfoBuf has not been read since the last data of the previous frame was read. That is, the information loaded into the UART1HDLCRXInfoBuf about the previous frame was never read and has been overwritten. Cleared by writing a "1" to this bit.

RFC: Received Frame Complete. (Read/Write)
Set to "1" when the last data byte for the frame is read from the RX FIFO (this also triggers an update of the UART1HDLCRXInfoBuf). Cleared by writing to a "1" to this bit.

RFS: Receive FIFO Service request. (Read Only)
This bit is a copy of the RIS bit in the UART interrupt identification register.
0 - RX FIFO is empty or RX is disabled.
1 - RX FIFO not empty and RX enabled.
May generate an interrupt and signal a DMA service request.

Copyright 2007 Cirrus Logic

TAB: Transmitted Frame Aborted. (Read/Write)
Set "1" when a transmitted frame is terminated with an abort. Cleared by writing to a "1" to this bit.

TFC: Transmit Frame Complete. (Read/Write)
Set to "1" whenever a transmitted frame completes, whether terminated normally or aborted. Cleared by writing to a "1" to this bit.

TFS: Transmit FIFO Service request. (Read Only)
This bit is a copy of the TIS bit in the UART interrupt identification register.
0 - TX FIFO is full or TX disabled.
1 - TX FIFO not full and TX enabled. May generate an interrupt and signal a DMA service request.

Copyright 2007 Cirrus Logic

Chapter 17

IrDA

17.1 Introduction

This module implements the physical layer of an infrared serial port that is compliant with Version 1.1 of the Infrared Data Association (IrDA) standard. It supports communication speeds of up to 4 MBit/s. When combined with analog transducer components, it provides a complete interface between infrared media and an AMBA compliant peripheral bus (APB).

Three different encoder/decoder units implement the supported modulation schemes and data encoding systems defined by the IrDA standard:

- Slow Infrared (SIR) - This interface attaches to the output of UART2. The UART2 registers handle the data and control for this interface, though the IrDA interface enable register selects the SIR function.
- Medium Infrared (MIR) - Transmission/reception rates can be 0.576 or 1.152 Mb/s.
- Fast Infrared (FIR) - Transmission/reception rate is 4 Mb/s.

17.2 IrDA Interfaces

The Infrared Interface Module implements in hardware the physical layer of an infrared serial port, compliant with version 1.1 of the IrDA standard. Communication speeds of up to 4 Mbit/sec are supported. When combined with analog transducer components, it provides a complete interface between infrared media and an AMBA compliant peripheral bus (APB).

The Module comprises three separate encoder/decoder units for implementing three different combinations of modulation scheme and data encoding system defined by the IrDA standard. These are:

- Slow Infrared - SIR - This interface attaches to the output of a UART. All data and control for this interface is done through the UART registers. The SIR encoder function is selected using the IrDA interface enable register.
- Medium Infrared - MIR - This interface is independent of a UART. Transmission/reception rates can be 0.576 or 1.152 Mbit/sec.
- Fast Infrared - FIR - This interface is independent of a UART. Transmission/reception rates can be 4 Mbit/sec.

Copyright 2007 Cirrus Logic

17.3 Shared IrDA Interface Feature

This section describes features common to the MIR and FIR interfaces (the SIR interface has been designed to share the enable register and device pins but is otherwise a separate interface assumed to be controlled by UART2).

17.3.1 Overview

The Slow Infrared (SIR) Encoder/Decoder is used to modulate and demodulate serial data using the Hewlett-Packard® Serial Infrared standard (HP-SIR) for bit encoding. Serial transmit data from UART2 is modulated using return-to-zero (RTZ) encoding to produce an output to drive the Ir transmitter LED, while data received from the Ir detector is converted into a serial bit stream to drive a UART's serial input. The SIR supports data rates up to 115.2 kbit/s.

The Medium Speed Infrared (MIR) Encoder/Decoder encodes/decodes peripheral bus data according to a modified HDLC standard, using flag characters, bit stuffing and a 16 bit CRC checker. MIR uses the same RTZ modulation and demodulation scheme used by the SIR. Two signal bit rates are supported: 0.576 Mbit/s and 1.152 Mbit/s.

The Fast Infrared Encoder/Decoder (FIR) operates at a fixed bit rate of 4 Mbit/s. Modulation/demodulation is by a phase shift key scheme called pulse position modulation (4 PPM). One of four signalling symbols represent each possible pair of data bits. Data encoding uses a packet format that prefixes bit and symbol synchronization flags to data and appends a 32-bit CRC and stop flag to the end of each packet. The start and stop flags use signalling symbols that are not used to encode data, hence bit stuffing of data is not required in this mode.

Only one of the Encoder/Decoder modules can be enabled to transmit and receive data from the IrDA transducers at one time Selection of an Ir sub-module is by means of the IrEnable register. The MIR and FIR sub-modules can be regarded by programmers as independent entities which are operated using common control and data registers, but which report status data via separate read registers.

Detailed descriptions of the MIR and FIR are given in the following sections. The SIR, however, has no data or control registers. It interfaces directly to a UART's serial stream. With the exception of the IrEnable register, it has no presence on the memory map and has no interface to the APB via the Infrared interface.

17.3.2 Functional Description

This section gives a programmer's guide to operating the IrDA interface. It includes detail on the general configuration and the transmit and receive processes.

Copyright 2007 Cirrus Logic

17.3.2.1 General Configuration

17.3.2.1.1 Select Ir Mode

The IrEnable register selects which of the three Ir sub-modules is used to operate the IrDA interface. Only one of the three may be active at any one time. The reset value for this register is zero, which disables all three encoder/decoder modules. The bottom two bits of this register select the encoder/decoder module according to the tabulated values:

Table 17-1. Bit Values to Select Ir Module

IrEnable EN1	IrEnable EN0	Encoder Selected
0	0	None
0	1	SIR
1	0	MIR
1	1	FIR

SIR does not use the data transfer mechanism described in this section. After selecting SIR mode, all data transfer operations are made through a UART, as if connection is through a serial cable without handshake lines. The features described below are implemented for the MIR and FIR modes.

17.3.2.1.2 Select Data Rate

The data rates for MIR and FIR are as follows:

- MIR - Clear BRD bit in IrControl (IrCon) for 0.576 Mbit/sec, Set BRD bit in IrCon for 1.152 Mbit/sec.
- FIR - Fixed at 4 Mbit/sec.

17.3.2.2 Transmitting Data

17.3.2.2.1 Initialization

The principal method of data transfer from memory to the active IrDA encoder (MIR or FIR) is by DMA. Typically DMA can be used to transfer data of any length into the transmit FIFO when requested by the infrared peripheral. When polling or interrupts are used to perform the data transfer, a mechanism exists for transmitting data packets that are not a multiple of 4 bytes in length. This uses a register called IrDataTail and its use is described in the next section.

The DMA route is usually provided to overcome any large interrupt response times that may exist in the SoC where the Infrared module is going to be used. These large interrupt response times can make programmed I/O an impractical method for transferring large Ir data packets.

Copyright 2007 Cirrus Logic

17.3.2.2.2 The Transmit Process

This section describes the transmission process in detail.

1. *Is last transmission complete?* - Ensure that the Infrared peripheral is not currently receiving or transmitting data by reading the RSY (for half-duplex communications) and TBY bits in the IrFlag register. If either is set, postpone the start of transmission.
2. *Disable IrDA* - If you are changing Ir mode, first disable Ir. To disable IrDA, first clear IrCtrl.RXE and IrCtrl.TXE. Secondly, clear the IrEnable.EN field to be "00".
3. *Disabling UART2 for MIR and FIR* - For MIR and FIR, disable UART2 by writing "0" to UART2Ctrl and 0 to IrCtrl.
4. *Set up the DMA Engine* - If DMA is being used, set up the DMA engine by setting up the registers of the DMA block.
5. *Enabling Clocks* - For MIR, set up the MIR clock in MIRClkDiv. Select 0.576 or 1.152 Mbps mode by clearing or setting IrCtrl.BRD. For FIR, enable the FIR clock by setting PwrCnt.FIR_EN.
6. *Select Ir Mode* - Select SIR, MIR, or FIR mode by writing the IrEnable.EN bit field to be "01", "10", or "11".
7. *Clear Interrupt Sticky Bits* - For MIR, write the MISR register, setting the TFC, TAB, RFL, and RIL bits to clear them. Then read the IrRIB register to clear the RFC bit. For FIR, write the FISR register, setting the TFC, TAB, RFL, and RIL bits to clear them. Then read the IrRIB register to clear the RFC bit.
8. *Select Transmit Underrun Action* - When DMA is used, the TUS bit should be cleared.
9. *Enable Transmit* - Set the IrCtrl.TXE Transmit Enable bit. Also set IrCtrl.RXE if receive is to be enabled. If DMA is used, also set IrDMACR.TXDMAE (and IrDMACR.RXDMAE if receive is to be enabled).
10. *Preloading the Transmit FIFO* - Copy the first two full words of data into the transmit FIFO by writing them into the IrData register. The Ir encode block can hold up to 11 bytes of data (two words in the FIFO plus up to three bytes in the IrDataTail register). If this is sufficient to hold the complete transmission data packet, DMA will not be needed. The IrCon.TUS bit should be cleared. This will cause the Ir encoder to correctly send the CRC and end of frame flag. **Note:** Prefilling the FIFO must happen immediately after enabling MIR or FIR. Preloading the FIFO is unnecessary for SIR. Also note that preloading the FIFO is unnecessary for MIR and FIR if DMA is used.
11. *Loading the IrDataTail Register* - In the PIO and IRQ case, once the FIFO has been preloaded, the IrDataTail register can be loaded. The IrDataTail register contains the last bytes in the frame (1, 2 or 3 bytes left over from the last whole word provided by PIO or IRQ). **Note:** If DMA is used, loading the IrDataTail register is unnecessary, as the IrDataTail register is disabled in that case.
12. *Send out the data* - If DMA is being used, everything is now enabled for the transmission process to begin. If PIO or IRQ is being used, data should be written to the IrData register.

Copyright 2007 Cirrus Logic

17.3.2.2.3 Sending Packets Which are Not a Multiple of 4 Bytes In Length

The transmit FIFO is 32 bits wide. When using polling or interrupts to effect the transfer, loading the FIFO with less than 32 bits would cause extraneous zero bits to be transmitted. This issue is taken care of automatically by the DMA, so no special action is required. However in the case of polling or interrupt-driven transfers, the IrDataTail register is the mechanism used to preload the last 1, 2 or 3 bytes of a frame. When the transfer is complete and the FIFO is empty, any bytes stored in the IrDataTail register are transmitted before the Ir encoder sends the CRC and end-of-frame flags. There are three distinct addresses to which the end of frame data is written. This allows a single word write to specify the data to be transmitted and the number of trailing bytes to send If there is a single trailing byte to transmit, write to address offset 0x014, for two bytes write to 0x018, and if there are three trailing bytes write to 0x01C. (See Table 17-2.)

Table 17-2. Address Offsets for End-of-Frame Data

Bytes to transmit	Address offset to use
1	0x014
2	0x018
3	0x01C

17.3.2.2.4 End of Frame Interrupt

Once all the data sent to the FIFO has been taken by the Ir interface, the FIFO will underrun. When this occurs any data that has been preloaded into the IrDataTail register will be used and the Transmitted Frame Complete (TFC) interrupt will be generated.

17.3.2.2.5 Disable Transmit Circuitry

To save power, the Transmit Enable (TXE) bit can be cleared in the IrEnable register if there are no frames that need to be sent.

17.3.2.2.6 Error conditions

Transmitted frame abort is only signalled if IrCon register bit TUS is set to 1.

17.3.2.3 Receiving Data

The end of a reception frame will cause an interrupt, which may be masked using the mask register (MIMR/FIMR). The end of frame interrupt occurs after the last data value has been transferred, including any odd bytes in the frame tail.

17.3.2.3.1 Initialization

The following settings are required:

Address Matching — To use Address Match filtering, set the local 8 bit address in the Address Match Value Register and set the Address Match Enable bit in the IrCon register.

Copyright 2007 Cirrus Logic

Set up DMA	Set up a DMA buffer (the buffer should be greater than twice the maximum possible size of received frames). Enable DMA. Alternatively, two buffers may be used which are each the maximum possible frame size long. The DMA would then be programmed to switch between the two buffers.
Enable Ir Receive	Set the Receive Enable bit (RXE) in IrEnable.

17.3.2.3.2 End of Frame Interrupt

The Receive Frame Complete (RFC) interrupt is generated when the last data in a frame is read from the receive FIFO. To check whether the frame was received correctly (no errors) and for information on frame size, the Receive Information Buffer register (IrRIB) must be read by the interrupt service routine. This also clears the RFC interrupt condition.

Note: By the time the ARM Core responds to this interrupt, the interface may have already started reception of a new frame.

17.3.2.3.3 End of Frame: Using Programmed I/O

If interrupt driven programmed I/O is used instead of DMA, every time the Receive Buffer Service (RFS) interrupt is serviced the IrFlag register must be read before the IrData register, if the IrFlag values are needed. Their Flag register gives information about error conditions that correspond to the data value at the head of the receive FIFO.

Note: The IrRIB registers stores status flags for a complete frame.

17.3.2.3.4 Error Conditions

Receive error conditions do not generate interrupts. Reading the IrData word clears the IrFlag register bits listed below.

Receiver Abort Detected	When set, this indicates that the transmitter sent an abort signal during frame transmission.
Receiver Overrun	This indicates that data has not been read for the IrData register in time and has resulted in data loss from the frame. When this occurs the interface automatically discards the remainder of the incoming frame.
CRC Error	If the CRC for the received data does not match the CRC value contained in the incoming data stream this condition will occur.
Frame Error (FIR only)	This indicates that a framing error has been detected.

Copyright 2007 Cirrus Logic

The data word and flags are held in the 39-bit wide receiver FIFO. Reading an IrData word removes both the data and its associated flag bits from the FIFO causing the next word in the FIFO (if present) to be transferred into the IrFlag and data registers. However, all error conditions encountered during a frame are remembered. At the end of frame they can be read form the IrRIB register.

When a receive overrun (ROR) or FIR framing error (FRE) is detected the remainder of the frame will be discarded by the receive logic (not put into the receive FIFO). In the case of receive overruns, if the end of frame (EOF) bit in the last entry in the FIFO is clear then the Receive Buffer Overrun (ROR) and EOF bits will be set. If an overrun occurs and the last entry in the FIFO already has the EOF bit set then the RFL interrupt will be triggered. In the case of a framing error an extra entry will be put into the FIFO with FIR Framing Error (FRE) and EOF set, this entry will not contain any valid data.

If programmed IO is used to service the IrDA interface instead of DMA a similar process occurs. Interrupt requests to service the receive FIFO will not occur until the rest of the frame has been discarded.

At the end of a frame, a valid end of frame (EOF) or an abort (RAB), a DMA request corresponding to the last word (which may hold 1, 2, 3, or 4 bytes of valid data) of the received frame will be raised. DMA will take the word. At that point the receive FIFO should be empty and the DMA request may be deasserted. The DMA request will be reasserted when data for a following frame is loaded into the receive FIFO.

The above behavior means there is no need for ARM Core intervention to service the IrDA interface between successive receive frames.

17.3.2.4 Special Conditions

17.3.2.4.1 Early Termination of Transmission

Clearing IrCon.TXE (transmit enable bit) stops transmission immediately. All data within the FIFO, transmit buffer and serial output shifter is cleared.

17.3.2.4.2 Early Termination of Reception

Clearing IrCon.RXE receive enable bit stops reception immediately. All data within the receive buffer, serial input shifter and FIFO is cleared.

17.3.2.4.3 Changing IrDA Mode

Poll the Transmitter Disabled bits – FD or MD bits – in IrEnable register until end of transmission is indicated. The new mode can then be set as described in *4.2.1General Configuration*.

17.3.2.4.4 Loopback Mode

For test purposes, data will be looped back – internally – from the output of the transmit serial shifter into the input of the receive serial shifter when IrEnable.LBM is set.

Copyright 2007 Cirrus Logic

17.3.3 Control Information Buffering

The ARM Core needs several items of information about a received frame that are not held in data DMAed from the receive FIFO, or stored in the DMA controller itself (because the DMA unit may be receiving the next frame by the time the ARM Core starts to work on the frame just completed). The additional information is as follows:

- A receive overrun or framing error occurred during frame reception.
- The frame failed the CRC check at the end of reception.
- Transmission of the frame was aborted.
- The number of bytes of valid data received in the frame (i.e. up to the end of frame or the overrun/framing error condition).

A control information buffer register is loaded whenever an end of received frame condition occurs. This event also generates an interrupt, which must be serviced before the end of the next received frame (at which point the buffered control information would be overwritten). The interrupt may be cleared by reading from the control information buffer register or by writing a '1' to its status bit position.

17.4 Medium IrDA Specific Features

The MIR comprises a dedicated serial port and RZI modulator/demodulator supporting the Infrared Data Association (IrDA) standard for transmission/reception at 0.576 and 1.152 Mb/s.

Frames contain an 8 bit address, an optional control field, a data field of any size that is a multiple of 8 bits and a 16-bit CRC-CCITT. The start/stop flag and CRC generation/checking is performed in the hardware. Data can be selectively saved in the receive buffer by programming an address with which to compare against all incoming frames. Interrupts are signalled when CRC checks performed on received data indicate an error, when a receiver abort occurs, when the transmit buffer underruns during an active frame and is aborted, when the receive buffer overruns and data is lost.

17.4.1 Introduction

17.4.1.1 Bit Encoding

The MIR bit encoding uses an RZI modulation scheme where a "0" is represented by a light pulse. For both 0.576 and 1.152 Mbps data rates, the optical pulse duration is normally 1/4 of a bit duration. For example, if the data frame (in the order of transmission) is 11010010b, then Figure 17-1 represents the signal that is actually transmitted.

Copyright 2007 Cirrus Logic

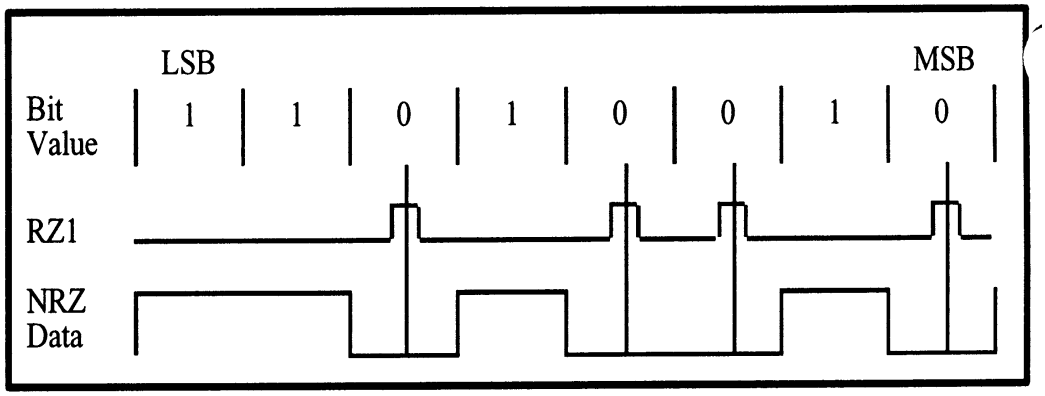


Figure 17-1. RZ1/NRZ Bit Encoding Example

17.4.1.2 Frame Format

MIR uses a flag (reserved bit pattern) to denote the beginning and end of a frame of information and to synchronize frame transmission. A double flag is used to indicate the start of a frame and a single flag the end. The flag contains eight bits, which start and end with a zero and contain six sequential ones in the middle (01111110b). This sequence of six ones is unique because all data between the start and stop flag is prohibited from having more than five consecutive ones. Data that violates this rule is altered before transmission by automatically inserting a zero after five consecutive ones are detected in the transmitted bit stream. This technique is commonly referred to as "bit stuffing" and is transparent to the user. The information field within a MIR frame is placed between the start and stop flags, consisting of an 8 bit address, an optional 8 bit control field, a data field containing any multiple of 8 bits and a 16 bit cyclic redundancy check (CRC-CCITT). Note that each byte within the address, control and data fields is transmitted and received LSB first, ending with the byte's MSB. However, the CRC is transmitted and received MSB first. The MIR frame format is outlined below in Table 17-3.

Table 17-3. MIR Frame Format

8 Bits	8 Bits	8 Bits	8 Bits (optional)	Any multiple of 8 Bits	16 Bits	8 Bits
Start Flag 0111 1110	Start Flag 0111 1110	Address	Control	Data	CRC-CCITT	Stop Flag 0111 1110

Copyright 2007 Cirrus Logic

17.4.1.2.1 Address Field

The 8 bit address field is used by a transmitter to target a select group of receivers when multiple stations are connected using the infrared link. The address allows up to 255 stations to be uniquely addressed (00000000b to 11111110b). The global address (11111111b) is used to broadcast messages to all stations. The serial port contains an 8 bit register that is used to program a unique address for broadcast recognition as well as a control bit to enable/disable the address match function. Note that the address of received frames is stored in the receive buffer along with normal data and that it is transmitted and received starting with its LSB and ending with its MSB.

17.4.1.2.2 Control Field

The MIR control field is typically 8 bits, but can be any length. The serial port does not provide any hardware decode support for the control byte, but instead treats all bytes between the address and the CRC as data. Thus any control bits appear as data to the programmer. Note that the control field is transmitted and received starting with its LSB and ending with its MSB.

17.4.1.2.3 Data Field

The data field can be any length that is a multiple of 8 bits, including zero. The user determines the data field length according to the application requirements and transmission characteristics of the target system. Usually a length is selected which maximizes the amount of data that can be transmitted per frame, while allowing the CRC checker to be able to consistently detect all errors during transmission. All data fields must be a multiple of 8 bits. If a data field that is not a multiple of 8 bits is received, an abort is signalled and the end of frame tag is set within the receive buffer. Also note that each byte within the data field is transmitted and received starting with its LSB and ending with its MSB.

17.4.1.2.4 CRC Field

MIR uses the established CCITT cyclical redundancy check (CRC) to detect bit errors that occur during transmission. A 16 bit CRC-CCITT is computed using the address, control and data fields and is included in each frame. A separate CRC generator is implemented in both the transmit and receive logic. The transmitter calculates a CRC while data is actively transmitted and places the 16 bit value at the end of each frame before the stop flag is transmitted. The receiver calculates a CRC for each received data frame and compares the calculated CRC to the expected CRC value contained within the end of each received frame. If the calculated value does not match the expected value, an interrupt is signalled. The CRC computation logic is preset to all ones before reception/transmission of each frame. Note that the CRC is transmitted and received starting with its MSB and ending with its LSB. The CRC uses the four term polynomial:

$$CRC(x) = (x^{16} + x^{12} + x^5 + 1)$$

Copyright 2007 Cirrus Logic

17.4.2 Functional Description

Following reset, the MIR is disabled. Reset also causes the transmit and receive buffers and tail register to be flushed (buffers marked as empty). To transmit data in MIR mode, use the following procedure:

1. Set the EN bits in the IrEnable register to 10b for MIR mode. Do not begin data transmission.
2. Before enabling the MIR, the user must first clear any writable or "sticky" status bits that are set by writing a one to each bit. (A sticky bit is a readable status bit that may be cleared by writing a one to its location.) Set the TAB and TFC bits in the MISR register, then read the MISR register to clear all interrupts.
3. Next, the desired mode of operation is programmed in the control register. Set the TXE and RXE bits in the IrCtrl register.
4. Write 1 to 3 bytes to the appropriate IrDataTail register.
5. Once the MIR is enabled, transmission/reception of data can begin on the transmit and receive pins.

17.4.2.1 Baud Rate Generation

The baud or bit rate is derived by dividing down an 18.423MHz clock. The clock is divided down by either 1 (BRD=1) or 2 (BRD=0) and then by a fixed value of four, generating the transmit clock for 1.152Mb/s and 0.576Mb/s data rates, respectively. The receive clock is generated by the receiver Digital Phase Locked Loop (DPLL). The DPLL uses a sample clock that is undivided. A sample rate counter (incremented at the sample clock rate) is used to generate a receive clock at the nominal data rate (sample clock divided by 41 and two-thirds). The sample rate counter is reset on the detection of each positive-going data transition (indicating the RZI encoding of a "0") to ensure that synchronization with the incoming data stream is maintained.

17.4.2.2 Receive Operation

Once the MIR receiver is enabled it enters hunt mode, searching the incoming data stream for the flag (01111110b). The flag serves to achieve bit synchronization, denotes the beginning of a frame and delineates the boundaries of individual bytes of data. The end of the second flag denotes the beginning of the address byte. Once the flag is found, the receiver is synchronized to incoming data and hunt mode is exited.

After each bit is decoded, a serial shifter is used to receive the incoming data a byte at a time. Once the flag is recognized, each subsequent byte of data is decoded and placed within a two byte temporary buffer. A temporary buffer is used to prevent the CRC from being placed within the receive buffer. When the temporary buffer is filled, data values are pushed out one by one to the receive buffer. The first byte of a frame is the address. If receiver address matching is enabled, the received address is compared to the address programmed in the address match value field in a control register. If the two values are equal or if the incoming address contains all ones, all subsequent data bytes including the address byte are stored in the receive buffer. If the values do not match, the receive logic does not store any data in the

Copyright 2007 Cirrus Logic

receive buffer, ignores the remainder of the frame and begins to search for the stop flag. The second byte of the frame can contain an optional control field that must be decoded in software (There is no hardware support within the MIR). Use of a control byte is determined by the user.

When the receive buffer contains a word of data, an interrupt or DMA request is signalled. If the data is not removed soon enough and the buffer is completely filled, an overrun error is generated when the receive logic attempts to place additional data into the full buffer. If this occurs all subsequent data in the frame is discarded by the interface and the last valid entry in the buffer is marked with the ROR and EOF bits. The interface will stall in this state until the receive buffer is emptied.

Frames can contain any amount of data in multiples of 8 bits. Although the MIR protocol does not limit frame size, in practice they tend to be implemented in numbers ranging from hundreds to a couple of thousand bytes. In general this interface expects received frame size to be limited to 2047 bytes. However, the interface can continue to operate past this limit provided that software drivers are written that carefully check the indicated frame length with the amount of data transferred (in the DMA case this is a little more difficult).

The receive logic continuously searches for the stop flag at the end of the frame. Once it is recognized, the last byte that was placed within the receive buffer is flagged as the last byte of the frame and the two bytes remaining within the temporary buffer are removed and used as the 16 bit CRC value for the frame. Instead of placing this in the receive buffer, the receive logic compares it to the CRC-CCITT value which is continuously calculated using the incoming data stream. If they do not match, the last byte that was placed within the receive buffer is also flagged with a CRC error. The CRC value is not placed in the receive buffer.

The MIR protocol permits back to back frames to be received. When this occurs, three flags separate back to back frames.

Most commercial IrDA transceivers can generate an abort (7 to 13 ones) when their transmit buffer underruns. The receive logic contains a counter that increments each time a one is decoded before entering the serial shifter and is reset any time a zero is decoded. When seven or more ones are detected, a receiver abort occurs. Note that data is moved from the serial shifter to the temporary buffer a byte at a time and seven consecutive ones may bridge two bytes. For this reason, after an abort is detected, the remaining data in the serial shifter is discarded along with the most recent byte of data placed in the temporary buffer. After this data is discarded, the oldest byte of data in the temporary buffer is placed in the receive buffer, the EOF tag is set within the top entry of the buffer (next to the byte transferred from the temporary buffer), the receiver abort interrupt is signalled and the receiver logic enters hunt mode until it recognizes the next flag.

This interface also generates an abort condition when a stop flag is received that is not byte aligned with the rest of the data in the frame. In this case the over flow data bits past the last byte boundary are discarded. It is not possible for the programmer to distinguish this condition for an normal abort condition.

If the user disables the receiver during operation, reception of the current data byte is stopped immediately, the serial shifter and receive buffer are cleared and all clocks used by the receive logic are automatically shut off to conserve power.

Copyright 2007 Cirrus Logic

17.4.2.3 Transmit Operation

Immediately after enabling the MIR for transmission, the user may either "prime" the transmit buffer by filling it with data (see section Section 17.4.2 on page 17--11 for details) or allow service requests to cause the CPU or DMA to fill the buffer once the MIR is enabled. Once enabled, the transmit logic issues a service request if its buffer is empty. A Serial Infrared Interaction Pulse (SIP) is transmitted in order to guarantee non-disruptive co-existence with slower (up to 115.2 Kbps) systems, for example another device attempting to use its SIR. This is followed by continuous transmission of flags until valid data resides within the buffer. Once a byte of data resides at the bottom of the transmit buffer, it is transferred to the serial shifter, is encoded and shifted out onto the transmit pin clocked by the programmed baud rate clock. Note that the flags and CRC value are automatically transmitted and need not be placed in the transmit buffer.

When the transmit buffer has space for another word, an interrupt and/or DMA service request is signalled. If new data is not supplied soon enough, the buffer is completely emptied and the transmit logic attempts to take additional data from the empty buffer, one of two actions can be taken as programmed by the user. An underrun can either signal the normal completion of a frame or an unexpected termination of a frame in progress.

When normal frame completion is selected and an underrun occurs, the transmit logic transmits the 16 bit CRC value calculated during the transmission of all data within the frame (including the address and control bytes), followed by a flag to denote the end of the frame. The transmitter then transmits an SIP, followed by a continuous transmission of flags until data is once again available within the buffer. Once data is available, the transmitter begins transmission of the next frame.

When unexpected frame termination is selected and an underrun occurs, the transmit logic outputs an abort and interrupts the CPU. An abort continues to be transmitted until data is once again available in the transmit buffer. The MIR then transmits an SIP, followed by a double flag and starts the new frame. The off-chip receiver may choose to ignore the abort and continue to receive data, or to signal the serial port to retry transmission of the aborted frame. If the user disables the transmitter during operation, transmission of the current data byte is stopped immediately, the serial shifter and transmit buffer are cleared and all clocks used by the transmit logic are automatically disabled to conserve power.

17.5 Fast IrDA Specific Features

The Fast Infrared port (FIR) operates at half-duplex and provides direct connection to commercially available Infrared Data Association (IrDA) compliant LED transceivers. The FIR supports the 4.0 Mbps IrDA standard, using four pulse position modulation (4 PPM) and a specialized serial packet protocol developed expressly for IrDA transmission.

Copyright 2007 Cirrus Logic

17.5.1 Introduction

17.5.1.1 4PPM Modulation

Four position pulse modulation (4PPM) is used for the high-speed transmission rate of 4.0 Mbps. Payload data is divided into data bit pairs (DBPs) for encoding with LSBs transmitted first. Each DBP is represented by one of four symbols (DDs) comprising a single 125 ms pulse within a 500 ms symbol period. The 125 ms quarters of a symbol are known as "chips". The resulting signal waveform for the four data DDs is shown in Figure 17-2 and Figure 17-3 and shows modulation of the byte, 10110001b which is constructed using four DBPs.

Note: 1. Bits within each DBP are not reordered, but the least significant DBP is transmitted first.

Note: 2. A "chip" in the context of the FIR is one time slice in the Position Modulation (PPM) symbol.

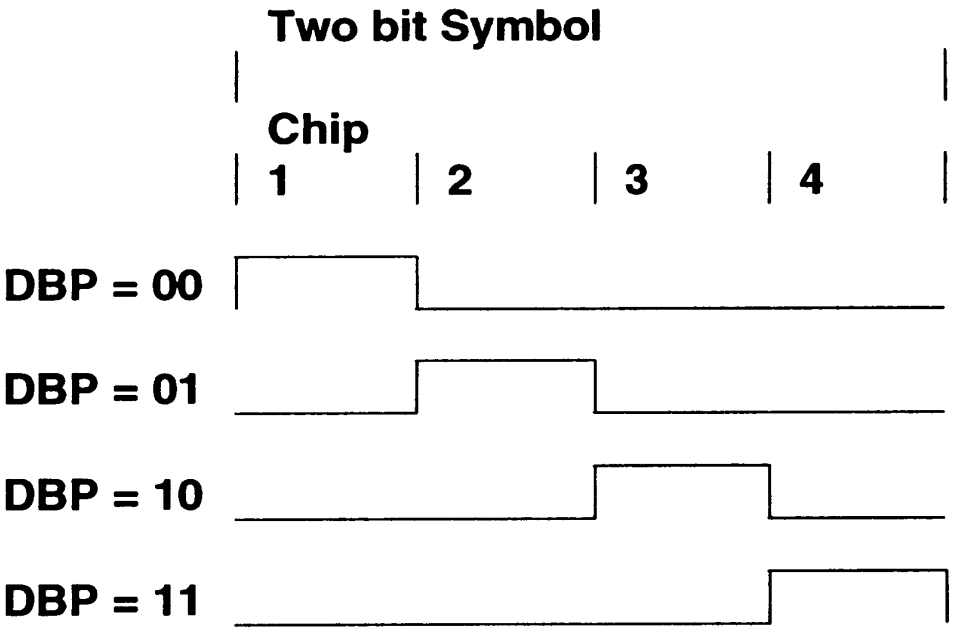


Figure 17-2. 4PPM Modulation Encoding

Copyright 2007 Cirrus Logic

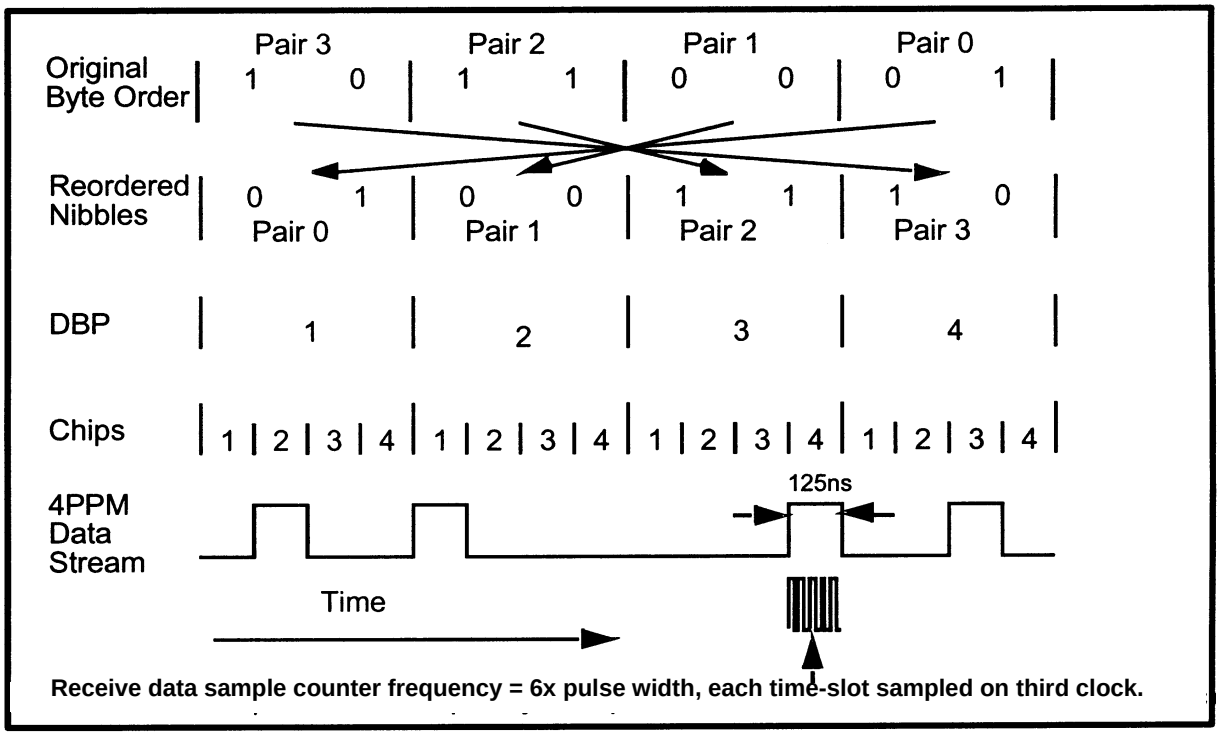


Figure 17-3. 4PPM Modulation Example

17.5.1.2 4.0 Mbps FIR Frame Format

When the 4.0 Mbps transmission rate is used, the high-speed serial/parallel (FIR) interface within the FIR is used along with the 4PPM bit encoding. The high-speed frame format shown in Figure 17-4, is similar to the SDLC format with several minor modifications: the start/stop flags and CRC are twice as long and instead of one start flag, a preamble and start flag of differing length are used.

64 symbols	8 symbols	4 DDs (8 bits)	4 DDs (8 bits)	8180 DDs max (2045 bytes)	16 DDs (32 bits)	8 symbols
Preamble	**Start Flag**	**Address**	**Control (optional)**	**Data**	**CRC-32**	**Stop Flag**
	Start Flag	|0000|1100|0000|1100|0110|0000|0110|0000|				
		|0000|1100|0000|1100|0000|0110|0000|0110|				**Stop Flag**
	Preamble	|1000|0000|1010|1000|... repeated 16 times				

Figure 17-4. IrDA (4.0 Mbps) Transmission Format

Copyright 2007 Cirrus Logic

The preamble, start and stop flags are a mixture of symbols which contain either 0, 1, or 2 pulses within the four time slots. Symbols with 0 and 2 pulses are used to construct flags since they represent invalid data bit pairings (one pulse required per symbol to represent one of four bit pairs). The preamble contains sixteen repeated transmissions of the four symbols: **1000 0000 1010 1000**, the start flag contains one transmission of eight symbols: **0000 1100 0000 1100 0110 0000 0110 0000** and the stop flag contains one transmission of eight symbols: **0000 1100 0000 1100 0000 0110 0000 0110**. The address, control, data and CRC-32 all use the standard 4PPM DDs described above.

17.5.1.2.1 Address Field

The 8 bit address field is used by a transmitter to target a select group of receivers when multiple stations are connected to the same set of serial lines. The address allows up to 255 stations to be uniquely addressed (00000000b to 11111110b). The global address (11111111b) is use to broadcast messages to all stations. Serial port 1 contains an 8 bit register which is used to program a unique address for broadcast recognition as well as a control bit to enable/disable the address match function. Note that the address of received frames is stored in the receive buffer along with normal data and that it is transmitted and received starting with its LSB and ending with its MSB.

17.5.1.2.2 Control Field

The IPC control field is 8 bits and is optional (as defined by the user). The FIR does not provide any hardware decode support for the control byte, but instead treats all bytes between the address and the CRC as data. Note that the control field is transmitted and received starting with its LSB and ending with its MSB.

17.5.1.2.3 Data Field

The data field can be any length which is a multiple of 8 bits, from 0 to 2045 bytes. The user determines the data field length according to the application requirements and transmission characteristics of the target system. Usually a length is selected which maximizes the amount of data which can be transmitted per frame, while allowing the CRC checker to be able to consistently detect all errors during transmission. Note that the serial port does not contain any hardware which restricts the maximum amount of data transmitted or received. It is up to the user to maintain these limits. If a data field which is not a multiple of 8 bits is received an abort is signalled. Also note that each byte within the data field is transmitted and received starting with its LSB and ending with its MSB.

17.5.1.2.4 CRC Field

The FIR uses the established 32 bit cyclical redundancy check (CRC-32) to detect bit errors which occur during transmission. A 32 bit CRC is computed using the address, control and data fields and is included in each frame. A separate CRC generator is implemented in both the transmit and receive logic. The transmitter calculates a CRC while data is actively transmitted byte shifting each byte transmitted through its serial shifter LSB first, then places the inverse of the resultant 32 bit value at the end of each frame before the flag is transmitted. In a similar manner, the receiver also calculates a CRC for each received data frame and compares the calculated CRC to the expected CRC value contained within the end of each received frame. If the calculated value does not match the expected value, an interrupt is

Copyright 2007 Cirrus Logic

signalled. The CRC computation logic is preset to all ones before reception/transmission of each frame and the result is inverted before it used for comparison or transmission. Note that unlike the address, control and data fields, the 32 bit inverted CRC value is transmitted and received from least significant byte to most significant and within each byte the least significant nibble is encoded/decoded first. The cyclical redundancy checker uses the 32 term polynomial:

$$CRC(x) = (x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1)$$

17.5.2 Functional Description

Following reset, the FIR is disabled. Reset also causes the transmit and receive buffers and tail register to be flushed (buffers marked as empty). To transmit data in FIR mode, use the following procedure:

1. Set the EN bits in the IrEnable register to 11b for FIR mode. Do not begin data transmission.
2. Before enabling the FIR, the user must first clear any writable or "sticky" status bits that are set by writing a one to each bit. (A sticky bit is a readable status bit that may be cleared by writing a one to its location.) Set the TAB and TFC bits in the FISR register, then read the FISR register to clear all interrupts.
3. Next, the desired mode of operation is programmed in the control register. Set the TXE and RXE bits in the IrCtrl register.
4. Write 1 to 3 bytes to the appropriate IrDataTail register.
5. Once the FIR is enabled, transmission/reception of data can begin on the transmit and receive pins.

17.5.2.1 Baud Rate Generation

The baud rate is derived by dividing down a fixed 48 MHz clock. The 8 MHz baud (time-slot) clock for the receiver is synchronized with the 4 PPM data stream each time a transition is detected on the receive data line using a digital PLL. To encode a 4.0 Mbps data stream, the required "symbol" frequency is 2.0 MHz, with four chips per symbol at a frequency of 8.0 MHz. Receive data is sampled half way through each time-slot period by counting three out of the six 48 MHz clock periods which make up each chip. Refer to Figure 17-3 on page 17-15. The symbols are synchronized during preamble reception. Recall that the preamble consists of four symbols repeated sixteen times. This repeating pattern is used to identify the first time-slot or beginning of a symbol and resets the two bit chip counter logic, such that the 4 PPM data is properly decoded.

Copyright 2007 Cirrus Logic

17.5.2.2 Receive Operation

The IrDA standard specifies that all transmission occurs at half-duplex. This restriction forces the user to enable one direction at a given time; either the transmit or receive logic, but not both. However, the FIR's hardware does not impose such a restriction. The user may enable both the transmitter and receiver at the same time. Although forbidden by the IrDA standard, this feature is particularly useful when using the FIR's loop back mode, which internally connects the output of the transmit serial shifter to the input of the receive serial shifter.

After the FIR is enabled for 4.0 Mbps transmission, the receiver logic begins by selecting an arbitrary symbol boundary, receives four incoming 4 PPM symbols from the input pin using a serial shifter and latches and decodes the symbols one at a time. If the symbols do not decode to the correct preamble, the chip counter's clock is forced to skip one 8MHz period, effectively delaying the chip count by one. This process is repeated until the preamble is recognized, signifying that the chip counter is synchronized. The preamble may be repeated as few as sixteen times, or may be continuously repeated to indicate an idle receive line.

At any time after the transmission of sixteen preambles, the start flag may be received. The start flag is eight symbols long. If any portion of the start flag does not match the standard encoding, the receiver signals a framing error and the receive logic once again begins to look for the frame preamble.

Once the correct start flag is recognized, each subsequent grouping of four DDs is decoded into a data byte, placed within a five byte temporary buffer which is used to prevent the CRC from being placed within the receive buffer. When the temporary buffer is filled, data values are pushed out one by one to the receive buffer. The first data byte of a frame is the address. If receiver address matching is enabled, the received address is compared to the address programmed in the address match value field in one of the control registers. If the two values are equal or if the incoming address contains all ones, all subsequent data bytes including the address byte are stored in the receive buffer. If the values do not match, the receiver logic does not store any data in the receive buffer, ignores the remainder of the frame and begins to search for the next preamble. The second data byte of the frame can contain an optional control field as defined by the user and must be decoded in software (there is no hardware support within the FIR).

Frames can contain any amount of data in multiples of 8 bits up to a maximum of 2047 bytes (including the address and control byte). In general this interface expects received frame size to be limited to 2047 bytes. However, the interface can continue to operate past this limit, thus it is the responsibility of the user to check that the size of each incoming frame does not exceed the IrDA protocol's maximum allowed frame size. The BC field in the IrRIB register can not be used for this since it will over flow (and wrap), the true frame length can be deduced from the DMA buffer position in combination with the BC field.

When the receive buffer contains a word of data, an interrupt or DMA request is signalled. If the data is not removed soon enough and the buffer is completely filled, an overrun error is generated when the receive logic attempts to place additional data into the full buffer. If this occurs all subsequent data in the frame is discarded by the interface and the last valid entry in the buffer is marked with the ROR and EOF bits. The interface will stall in this state until the receiver buffer is emptied.

Copyright 2007 Cirrus Logic

When a framing error is detected all subsequent data in the frame is discarded by the interface and an entry is put into the buffer with the FRE and EOF bits set The data in this buffer entry is invalid.

If any two sequential symbols within the data field do not contain pulses (are 0000b), the frame is aborted. The oldest byte in the temporary buffer is moved to the receive buffer (the remaining four buffer entries are discarded). The end of frame (EOF) tag is set within the same buffer entry where the last "good" byte of data resides and the receiver logic begins to search for the preamble. An abort occurs if any data symbol contains 0011b, 1010b, 0101b, or 1001b (invalid symbols which do not occur in the stop flag).

The receiver continuously searches for the 8 symbol stop flag. Once it is recognized, the last byte placed within the receive buffer is flagged as the last byte of the frame and the data in the temporary buffer is removed and used as the 32 bit CRC value for the frame. Instead of placing this in the receive buffer, the receiver compares it to the CRC-32 value which is continuously calculated using the incoming data stream. If they do not match, the last byte which was placed in the receiver buffer is also tagged with a CRC error. The CRC value is not placed in the receive buffer.

If the user disables the FIR's receiver during operation, reception of the current data byte is stopped immediately, the serial shifter and receive buffer are cleared and all clocks used by the receive logic are automatically shut off to conserve power.

17.5.2.3 Transmit Operation

Immediately after enabling the FIR for transmission, the user may either "prime" the transmit buffer by filling it with data (see section Section 17.5.2 on page 17--17 for details) or allow service requests to cause the CPU or DMA to fill the buffer once the FIR is enabled. Once enabled, the transmit logic issues a service request if its buffer is empty. For each frame output, a minimum of sixteen preambles are transmitted. If data is not available after the sixteenth preamble, additional preambles are output until a byte of valid data resides within the bottom of the transmit buffer. The preambles are then followed by the start flag and then the data from the transmit buffer. Four symbols (8 bits) are encoded at a time and then loaded into a serial shift register. The contents are shifted out onto the transmit pin clocked by the 8 MHz baud clock. Note that the preamble, start and stop flags and CRC value is automatically transmitted and need not be placed in the transmit buffer.

When the transmit buffer is emptied, an interrupt and/or DMA service request is signalled. If new data is not supplied quickly enough and the transmit logic attempts to take additional data from the empty buffer, one of two actions can be taken as programmed by the user. An underrun can either signal the normal completion of a frame or an unexpected termination of a frame in progress.

When normal frame completion is selected and an underrun occurs, the transmit logic transmits the 32 bit CRC value calculated during the transmission of all data within the frame (including the address and control bytes), followed by the stop flag to denote the end of the frame. The transmitter then continuously transmits preambles until data is once again available within the buffer. Once data is available, the transmitter begins transmission of the next frame.

Copyright 2007 Cirrus Logic

When unexpected frame termination is selected and an underrun occurs, the transmit logic outputs an abort and interrupts the CPU. An abort continues to be transmitted until data is once again available in the transmit buffer. The FIR then transmits 16 preambles, a start flag and starts the new frame. The remote receiver may choose to ignore the abort and continue to receive data, or to signal the FIR to retry transmission of the aborted frame.

At the end of each frame transmitted, the FIR outputs a pulse called the serial infrared interaction pulse (SIP). A SIP is required at least every 500 ms to keep slower speed devices (115.2 kbps and slower) from colliding with the higher speed transmission. The SIP simulates a start bit which causes all low speed devices to stay off the bus for at least another 500 ms. Transmission of the SIP pulse causes the transmit pin to be forced high for a duration of 1.625 µs and low for 7.375 µs (total SIP period = 9.0 µs). After the 9.0 µs elapses, the preamble is then transmitted continuously to indicate to the remote receiver that the FIR's transmitter is in the idle state. The preamble continues to be transmitted until new data is available within the transmit buffer, or the FIR's transmitter is disabled. Note that it is the responsibility of the user to ensure that a frame completes once every 500 ms such that a SIP pulse is produced keeping all low speed devices from interrupting transmission. Because most IrDA compatible devices produce a SIP after each frame transmitted, the user may only need to ensure that a frame is either transmitted or received by the FIR every 500 ms.

Note that frame length does not represent a significant portion of the 500 ms time frame in which a SIP must be produced. At 4.0 Mbps, the longest frame allowed is 16,568 bits, which takes just over 4 ms to transmit. Also note that the FIR issues a SIP when the transmitter is first enabled, to ensure all low speed devices are silenced before transmitting its first frame.

If the user disables the FIR's transmitter during operation, transmission of the current data byte is stopped immediately, the serial shifter and transmit buffer are cleared. All clocks used by the transmit logic are automatically shut off to conserve power.

17.5.3 IrDA Connectivity

The IrDA controller uses package pins **RXD1** and **TXD1**. The IrDA input signal is always **RXD1**. Syscon register DeviceCfg.IonU2 controls what drives bit **TXD1**. See Figure 17-4.

Table 17-4. DeviceCfg.IonU2 Pin Function

DeviceCfg.IonU2	Pin TXD1 Function
0	UART2 is the output signal
1	Logical OR of IrDA output signal and UART2 SIR output signal

Therefore, to use any IrDA mode, FIR, MIR or SIR, set IonU2. To use UART2 as a UART, clear IonU2.

Copyright 2007 Cirrus Logic

17.5.4 IrDA Integration Information

17.5.4.1 Enabling Infrared Modes

Table 17-5. UART2 / IrDA Modes

Mode	DeviceCfg Register		UART2Ctrl Register		IrEnable Register	
	U2EN	IonU2	SIREn	UARTE	EN[1]	EN[0]
Disabled	0	x	0	0	0	0
UART2	1	0	0	1	0	0
SIR	1	1	1	1	0	1
MIR	x	1	0	0	1	0
FIR	x	1	0	0	1	1

17.5.4.2 Clocking Requirements

There are four clocks, PCLK, MIRCLK, FIRCLK, and UARTCLK.

Version 1.1 of the Infrared Data Association standard indicates the following:

- FIRCLK must by 48.0 MHz with a tolerance of 0.01%.
- MIRCLK must be 18.432 MHz with a tolerance of 0.1%.

The worst case ratio that can be supported for PCLK:FIRCLK is a ratio of 1:5. The maximum that PCLK can be is 66 MHz, therefore:

$$\frac{1}{5}F_{FIRCLK} < F_{PCLK} < 66.0MHz$$

Any frequencies outside the above range are not supported and will result in incorrect behavior of the FIR mode of the infrared peripheral.

Since MIRCLK is 18.432 MHz, PCLK can be as low as 3.68 MHz and as high as 66 MHz. Any PCLK frequency in this range is allowable. Any PCLK frequencies outside the range are not supported and will result in incorrect behavior of the MIR mode of the infrared peripheral, therefore:

$$3.68MHz \le F_{PCLK} \le 66.0MHz$$

The tolerance of UARTCLK is defined by the UART to which it is connected.

UARTCLK frequency must accommodate the desired range of baud rates:

$$F_{UARTCLK_{MIN}} \ge 32 \times baudrate_{MAX}$$

$$F_{UARTCLK_{MAX}} \le 32 \times 65536 \times baudrate_{MIN}$$

The frequency of UARTCLK must also be within the required error limits for all baud rates to be used.

Copyright 2007 Cirrus Logic

To allow sufficient time to write the received data to the receive FIFO, UARTCLK must be less than or equal to four times the frequency of PCLK:

$$F_{UARTCLK} \leq 4 \times F_{PCLK}$$

If the IrDA SIR functionality is required, UARTCLK must have a frequency between 2.7 MHz and 542.7 MHz to ensure that the low-power mode transmit pulse duration complies with the IrDA SIR specification.

17.5.4.3 Bus Bandwidth Requirements

There are four different IrDA modes with different bandwidth requirements. Furthermore, there are two basic ways of moving data to or from the IrDA FIFOs:

- Direct DMA interface - this permits byte-wide access to the IrDA without using the APB. The DMA block will pack/unpack individual bytes so that it reads or writes full 32-bit words rather than individual bytes.
- Accessing the IrDA via the APB - this requires APB/AHB bus bandwidth. Then, both a read and write are required for each 32-bit data word.

Assuming most bytes in a packet are moved either via the DMA interface or via 32-bit word accesses to the IrDA controller on the APB, Table 17-6 indicates the maximum average number of memory accesses per second to service IrDA TX or RX:

Table 17-6. IrDA Service Memory Accesses / Second

Infrared Mode	Bit Rate (bits / second)	Bus accesses / second	
		DMA	APB
SIR	115,200	3,600	7,200
Slow MIR	576,000	18,000	36,000
Fast MIR	1,152,000	36,000	72,000
FIR	4,000,000	125,000	250,000

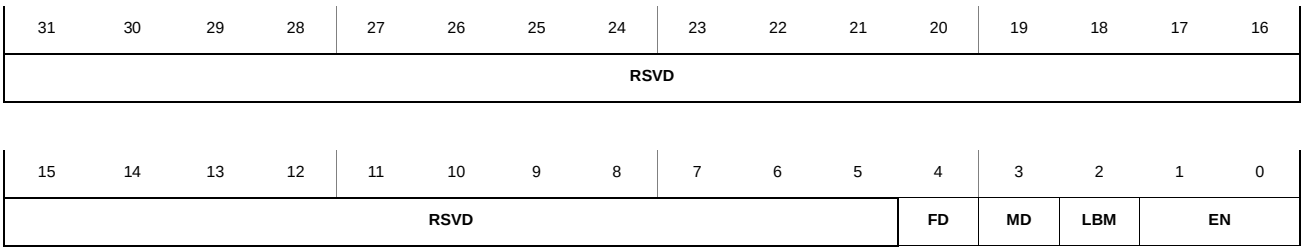
Note that the SIR mode bit rate is a worst case value.

Copyright 2007 Cirrus Logic

17.6 Registers

Register Descriptions

IrEnable



Address:

0x808B_0000 - Read/Write

Default:

0x0000_0018

Definition:

IrDA Enable Register. This register selects which Infrared interface module is active. The Medium and Fast modules share common control, flag, and data interfaces while maintaining separate status registers.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

FD: Fast done status. Read-only bit indicating that the FIR transmit module has completed transition of the current frame and that it is safe to disable the module using the EN control bits.

MD: Medium done status. Read-only bit indicating that the MIR transmit module has completed transmission of the current frame and that it is safe to disable the module using the EN control bits.

LBM: Loopback Mode, for MIR and FIR operation.
0 - Normal operation.
1 - Loopback active, the transmit serial shifter is directly connected to the receive serial shifter.

Copyright 2007 Cirrus Logic

EN: Enable value:
00 - No encoder selected
01 - SIR, 0 to 0.1152Mbit/s data rate, using the UART2 interface
10 - MIR, 0.576 or 1.152Mbit/s data rate, using IrDA interface
11 - FIR, 4.0Mbit/s data rate, using IrDA interface.

17

Note: While the FIR transmit section is enabled, the FD bit is low, and while the MIR transmit section is enabled, the MD bit is low. In FIR mode, the FD bit does not go high until the TXE bit in the IrCtrl register is cleared, and in MIR mode, the same bit must be cleared for MD to go high. Monitor the TBY bit in the IrFlag register to discover whether a packet is fully transmitted before clearing TXE.

IrCtrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								AME	RXP	TXP	RXE	TXE	TUS	BRD	0

Address:
0x808B_0004

Default:
0x0000_0000

Definition:
IrDA Control Register. This register selects various operating parameters. Note that the RXE and TXE bit must be cleared before selecting a different interface with the IrEnable register EN bits. The other bits in this register may be changed while the interface is active.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

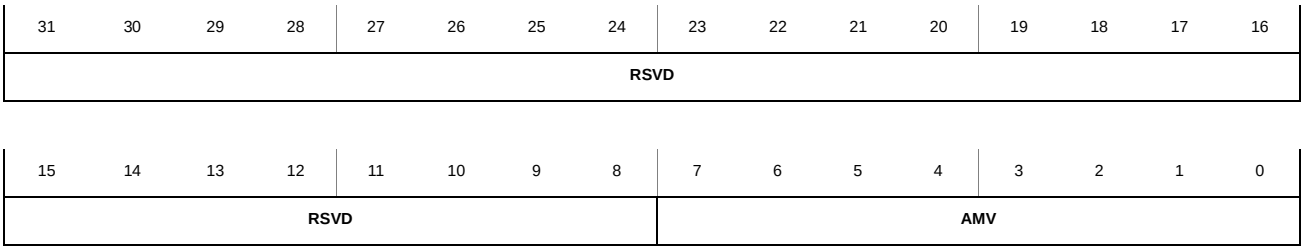
AME: Address Match Enable.
0 - Disable receiver address match function, store data from all incoming frames in the receive buffer.
1 - Enable receiver address match function, do not buffer data unless address is recognized or incoming address contains all ones.

RXP: Receive Polarity Control.
0 - Data input is not inverted before decoding.
1 - Data input is inverted before decoding.

Copyright 2007 Cirrus Logic

TXP:	Transmit Polarity Control. 0 - Encoded data is not inverted before being passed to the pins. 1 - Encoded data is inverted before being passed to the pins.
RXE:	Receive Enable. 0 - Ir receive logic is disabled and clocks are stopped. 1 - Ir receive logic is enabled.
TXE:	Transmit Enable. 0 - Transmit logic is disabled and clocks are stopped. 1 - Transmit logic is enabled.
TUS:	Transmit buffer Underrun Select. 0 - Transmit buffer underrun causes CRC, stop flag, and SIP to be transmitted. 1 - Transmit buffer underrun causes an abort to be transmitted.
BRD:	MIR Bit rate select. 0 - MIR data rate is 0.576 Mbit/s. 1 - MIR data rate is 1.152 Mbit/s.
0:	Must be written to "0".

IrAdrMatchVal



Address:

0x808B_0008 - Read/Write

Default:

0x0000_0000

Definition:

IrDA Address Match Value Register contains the 8 bit address match value field which is used by the receiver to selectively store only the data within the receive frames which have the same address. For incoming frames which have the same address value as the AMV field, the frame's address, control and data is stored in the receive buffer. For those that do not match, the remainder of the frame is ignored and the receive logic searches for the beginning of the next frame. This register is used for both MIR and FIR. The AME bit in IrCtrl must be set to enable this function. Frames containing an

Copyright 2007 Cirrus Logic

address of all ones are broadcast frames, and are always matched regardless of the value in the AMV. The AMV may be written at any time, allowing the address match value to be changed during active receive operation.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

AMV: Address Match Value.

17

IrFlag

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD						TBY	RIF	RSY	EOF	WST	WST	FRE	ROR	CRE	RAB

Address:

0x808B_008B - Read Only

Default:

0x0000_0000

Definition:

IrDA Flag Register. Contains the nine read only flags which indicate the current state of the IrDA Interface.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

TBY: Transmitter Busy Flag.
0 - Transmitter is idle, or disabled, or an abort is being transmitted.
1 - Transmit logic is currently transmitting a frame.

RIF: Receiver In Frame.
0 - Receiver is in preamble/start flag or is in hunt mode.
1 - Receiver is in a frame.

RSY: Receiver Synchronized Flag.
0 - Receiver is in hunt mode.
1 - Receiver logic is synchronized within the incoming data.

EOF: End of Frame.
0 - Current frame is not completed.
1 - The word in the receive buffer contains the last byte of data within the frame. When the last word in the current frame is read this bit is cleared.

Copyright 2007 Cirrus Logic

WST: Width Status.
00 - All four bytes in receive buffer are valid.
01 - Least significant byte is valid only.
10 - Least significant two bytes are valid only.
11 - Least significant three bytes are valid only.

FRE: FIR Framing Error.
0 - No framing errors encountered in the receipt of FIR data.
1 - Framing error occurred, FIR preamble followed by something other than another preamble or FIR start flag. The data in the buffer is invalid.

ROR: Receive buffer Overrun.
0 - Receive buffer has not experienced an overrun.
1 - Receive logic attempted to place data into receive buffer while it was full. The next data value in the buffer is the last piece of "good" data before the buffer was overrun.

CRE: CRC Error.
0 - No CRC check errors encountered in the data.
1 - CRC calculated on the incoming data does not match CRC value contained within the received frame.

RAB: Receiver Abort.
0 - No abort has been detected for the incoming frame.
1 - Abort detected during receipt of the incoming frame, EOF bit set in receive buffer next to the last piece of "good" data received before abort.

IrData

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															

Address:
0x808B_0010 - Read/Write

Default:
0x0000_0000

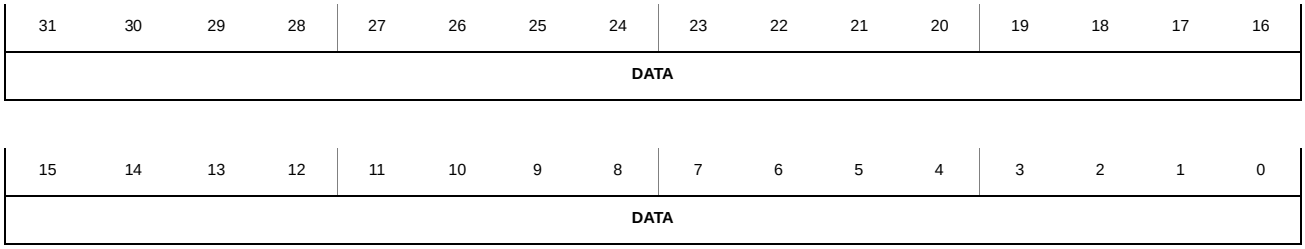
Definition:
IrDA Data Register. Provides access to the transmit and receive buffers used by the MIR and FIR interfaces.

Copyright 2007 Cirrus Logic

Bit Descriptions:

DATA: IrDA data word. Values written and sent to the transmit FIFO. Values read are from the receiver FIFO.

IrDataTail



Address:

0x808B_0014, 0x808B_0018, 0x808B_001C - Write Only

Default:

0x0000_0000

Definition:

IrDA Data Tail Register. This is a 24-bit write only register used for transmitting frames whose payload data is not an integer multiple of 4 bytes long. The bit locations are cleared when read by the transmit logic or when the TXE control bit is clear. The IrData Tail register may be written using one of three addresses. Bits two and three of the address determine how many bytes within the word are significant, that is, are intended for transmission. If none of the address is written, the register remains marked as empty and payload data will be read by the transmit logic from the 32-bit FIFO only. The status of this register does not affect the TFS flag, nor does it cause interrupts or DMA requests to be generated.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

DATA: IrDA transmit payload data. Write to address 0x014, least significant byte is transmitted. Write to address 0x018, least significant two bytes are transmitted. Write to address 0x01C, least significant three bytes are transmitted.

Copyright 2007 Cirrus Logic

IrRIB

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	BC											BFRE	BROR	BCRE	BRAB

Address:

0x808B_0020 - Read Only

Default:

0x0000_0000

Definition:

IrDA Receive Information Register. This register contains 15 read only bits that identify flag and byte count values from the last received frame. The bits are copied from the flag register when the last data in a frame is read from the receive FIFO.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

BC: Byte Count. The total number of valid bytes read from the interface during the last frame. If the total number of bytes is greater than 2047, only the lower eleven bits are presented.

BFRE: Buffered Framing Error.
0 - No framing errors were encountered during the last frame.
1 - A framing error occurred during the last frame causing the remainder of the frame to be discarded.

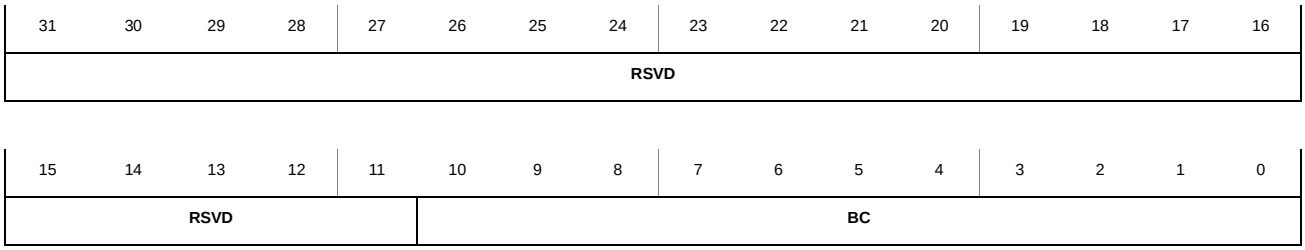
BROR: Buffered Receive buffer Overrun.
0 - The receive buffer did not overrun during the last frame.
1 - Receive logic attempted to place data into receive buffer while it was full during the last frame causing the remainder of the frame to be discarded.

BCRE: Buffered CRC Error.
0 - No CRC check errors encountered in the last frame.
1 - CRC calculated on the incoming data did not match CRC value contained within the received frame for the last frame.

Copyright 2007 Cirrus Logic

BRAB: Buffered Receiver Abort.
0 - No abort was detected in the last frame.
1 - The last frame was terminated with an abort condition.

IrTR0



Address:
0x808B_0024 - Read Only

Default:
0x0000_0000

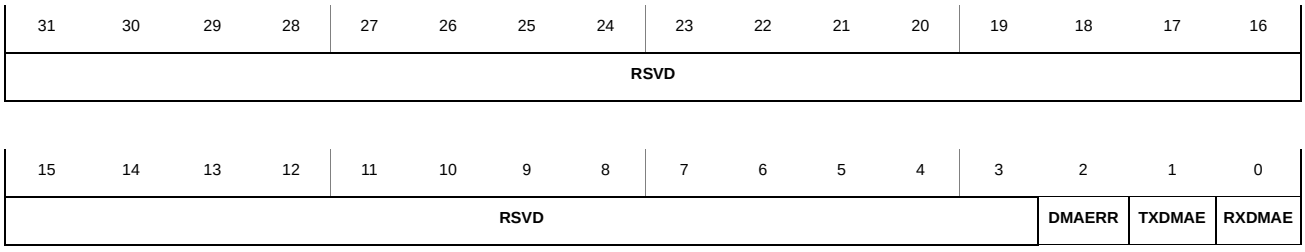
Definition:
IrDA Test Register 0. This register indicates the received byte count.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

BC: Byte Count. The total number of valid bytes read by the receiver.

IrDMACR



Address:
0x808B_0028 - Read/Write

Default:
0x0000_0000

Definition:
IrDA DMA Control Register.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

Copyright 2007 Cirrus Logic

DMAERR: RX DMA error handing enable. If 0, the RX DMA interface ignores error conditions in the IrDA receive section. If "1", the DMA interface stops and notifies the DMA block when an error occurs. Errors include framing errors, receive abort, and CRC mismatch.

TXDMAE: TX DMA interface enable. Setting to "1" enables the private DMA interface to the transmit FIFO.

RXDMAE: RX DMA interface enable. Setting to "1" enables the private DMA interface to the receive FIFO.

SIRTR0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								SIREN	SIROUT	TXD	RXD	SIRT	SIRIN	S16CLK	TSIRC

Address:

0x808B_0030 - Read/Write

Default:

0x0000_0000, except that bit 4 is unknown at reset

Definition:

IrDA Slow InfraRed Test Register 0.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

SIREN: The state of the **SIREN** after synchronization. Read only.

SIROUT: The state of **SIROUT** output from the InfraRed block. Read only.

TXD: The state of the **TXD** input to the InfraRed block from UART2. Read only.

RXD: The state of the **RXD** output from the InfraRed block to UART2. Read only.

Copyright 2007 Cirrus Logic

MISR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									RFL	RIL	RFC	RFS	TAB	TFC	TFS

Address:

0x808B_0080 - Read/Write

Default:

0x0000_0000

Definition:

MIR Status Register.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RFL: Receive Frame Lost. Set to a "1" when a ROR occurred at the start of a new frame, before any data for the frame could be put into the receive FIFO. This bit is cleared by writing a "1" to this bit. This occurs if the last entry in the FIFO already contains a valid EOF bit from a previous frame when a FIFO overrun occurs. The ROR bit cannot be placed into the FIFO and all data associated with the frame is lost.

RIL: Receive Information Buffer Lost. Set to a "1" when the last data for a frame is read from the receive FIFO and the RFC bit is still set from a previous end of frame. This bit is cleared by writing a "1" to this bit. This is triggered if the RFC bit is already set before the last data from a frame is read from the IrData register. It indicates that the data from the IrRIB register was lost. This can occur if the CPU does not respond to the RFC interrupt before another (short) frame completes and is read from the IrData register by the DMA controller.

RFC: Received Frame Complete. Set to "1" when the last data for a frame is read from the receive FIFO (via the IrData register). This event also triggers the IrRIB to load the IrFlag and byte count. This bit is cleared when the IrRIB register is read.

Copyright 2007 Cirrus Logic

RFS: Receive buffer Service Request (read only).
0 - Receive buffer is empty or the receiver is discarding data or the receiver is disabled.
1 - Receive buffer is not empty and the receiver is enabled, DMA service request signaled.

TAB: Transmit Frame Aborted. Set to "1" when a transmitted frame is terminated with an abort. This will only occur if the TUS bit is set in the IrCtrl register. Writing a "1" to this bit clears it.

TFC: Transmitted Frame Complete. Set to "1" whenever a transmitted frame completes, whether it is terminated with a CRC followed by a stop flag or terminated with an abort. Writing a "1" to this bit clears it.

TFS: Transmit buffer Service Request (read only).
0 - Transmit buffer is full or transmitter disabled.
1 - Transmit buffer is not full and the transmitter is enabled, DMA service is signaled.
The bit is automatically cleared after the buffer is filled.

MIMR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									RFL	RIL	RFC	RFS	TAB	TFC	TFS

Address:
0x808B_0084 - Read/Write

Default:
0x0000_0000

Definition:
MIR Interrupt Mask Register.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RFL: RFL mask bit. When high, the MIR RFL status can generate an interrupt.

RIL: RIL mask bit. When high, the MIR RIL status can generate an interrupt.

Copyright 2007 Cirrus Logic

17

RFC: RFC mask bit. When high, the MIR RFC status can generate an interrupt.

RFS: RFS mask bit. When high, the MIR RFS status can generate an interrupt.

TAB: TAB mask bit. When high, the MIR TAB status can generate an interrupt.

TFC: TFC mask bit. When high, the MIR TFC status can generate an interrupt.

TFS: TFS mask bit. When high, the MIR TFS status can generate an interrupt.

MIIR

Address:

0x808B_0088 - Read Only

Default:

0x0000_0000

Definition:

MIR Interrupt Register. The IrDA interrupt is asserted if any bit in the MIIR is high.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RFL: Logical AND of MIR RFL status bit and RFL mask bit.

RIL: Logical AND of MIR RIL status bit and RIL mask bit.

RFC: Logical AND of MIR RFC status bit and RFC mask bit.

RFS: Logical AND of MIR RFS status bit and RFS mask bit.

TAB: Logical AND of MIR TAB status bit and TAB mask bit.

TFC: Logical AND of MIR TFC status bit and TFC mask bit.

TFS: Logical AND of MIR TFS status bit and TFS mask bit.

Copyright 2007 Cirrus Logic

FISR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									RFL	RIL	RFC	RFS	TAB	TFC	TFS

Address:

0x808B_0180 - Read/Write

Default:

0x0000_0000

Definition:

FIR Status Register.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RFL: Receive Frame Lost. Set to a "1" when a ROR occurred at the start of a new frame, before any data for the frame could be put into the receive FIFO. This bit is cleared by writing a "1" to this bit. This occurs if the last entry in the FIFO already contains a valid EOF bit from a previous frame when a FIFO overrun occurs. The ROR bit cannot be placed into the FIFO and all data associated with the frame is lost.

RIL: Receive Information Buffer Lost. Set to a "1" when the last data for a frame is read from the receive FIFO (via the IrData register) and the RFC bit is still set from a previous end of frame. It indicates that data in the IrRIB register for the previous frame was lost. This can occur if the CPU does not respond to the RFC interrupt before another frame completes and is read from the IrData register by the DMA controller. This bit is cleared by writing a "1" to this bit.

RFC: Received Frame Complete. Set to "1" when the last data for a frame is read from the receive FIFO (via the IrData register). This event also triggers the IrRIB to load the IrFlag and byte count. This bit is cleared when the IrRIB register is read.

Copyright 2007 Cirrus Logic

17

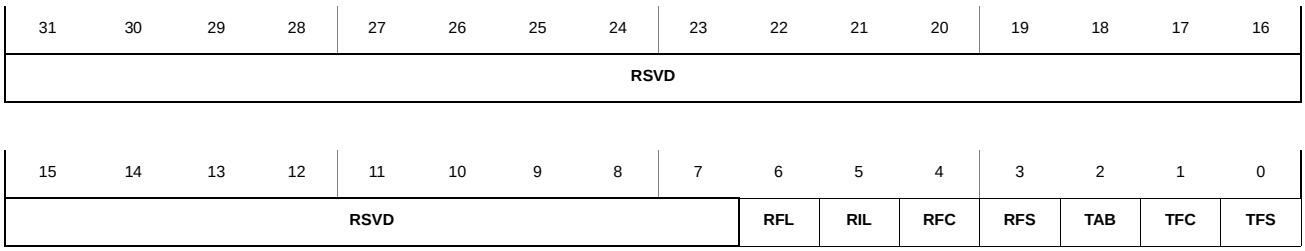
RFS: Receive buffer Service Request (read only).
0 - Receive buffer is empty or the receiver is discarding data or the receiver is disabled.
1 - Receive buffer is not empty and the receiver is enabled, DMA service request signaled.
The bit is automatically cleared when the receive buffer is emptied.

TAB: Transmit Frame Aborted. Set to "1" when a transmitted frame is terminated with an abort. This will only occur if the TUS bit is set in the IrCtrl register. The bit is cleared by writing a "1" to this bit.

TFC: Transmitted Frame Complete. Set to "1" whenever a transmitted frame completes (whether it is terminated with a CRC followed by a stop flag or terminated with an abort). This bit is cleared by writing a "1" to this bit.

TFS: Transmit buffer Service Request (read only).
0 - Transmit buffer is full or transmitter disabled.
1 - Transmit buffer is not full and the transmitter is enabled, DMA service is signaled.
The bit is automatically cleared after the buffer is filled.

FIMR



Address:
0x808B_0184 - Read/Write

Default:
0x0000_0000

Definition:
FIR Interrupt Mask Register.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RFL: RFL mask bit. When high, the FIR RFL status can generate an interrupt.

RIL: RIL mask bit. When high, the FIR RIL status can generate an interrupt.

Copyright 2007 Cirrus Logic

RFC: RFC mask bit. When high, the FIR RFC status can generate an interrupt.

RFS: RFS mask bit. When high, the FIR RFS status can generate an interrupt.

TAB: TAB mask bit. When high, the FIR TAB status can generate an interrupt.

TFC: TFC mask bit. When high, the FIR TFC status can generate an interrupt.

TFS: TFS mask bit. When high, the FIR TFS status can generate an interrupt.

FIIR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									RFL	RIL	RFC	RFS	TAB	TFC	TFS

Address:

0x808B_0188 - Read Only

Default:

0x0000_0000

Definition:

FIR Interrupt Register. An interrupt is signalled from this block if any bit is high in the FIIR.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RFL: Logical AND of FIR RFL status bit and RFL mask bit.

RIL: Logical AND of FIR RIL status bit and RIL mask bit.

RFC: Logical AND of FIR RFC status bit and RFC mask bit.

RFS: Logical AND of FIR RFS status bit and RFS mask bit.

TAB: Logical AND of FIR TAB status bit and TAB mask bit.

TFC: Logical AND of FIR TFC status bit and TFC mask bit.

TFS: Logical AND of FIR TFS status bit and TFS mask bit.

Copyright 2007 Cirrus Logic

17

Copyright 2007 Cirrus Logic

Chapter 18
Timers

18.1 Introduction

The timers are used to control timed events in the system. For example, a wait can be inserted by setting the timer value to an appropriate value and waiting for the timer interrupt.

The Timers block contains two 16-bit timers, one 32-bit timer and one 40-bit time stamp debug timer.

18.1.1 Features

The processor has these timer features:

- Two 16-bit timers
  - Free running
  - Load based
- One 32-bit timer
  - Free running
  - Load based
- One 40-bit timer
  - Free running

18.1.2 16 and 32-bit Timer Operation

The two 16-bit timers are referred to as TC1 and TC2. Each of these timers has an associated 16-bit read/write data register and a control register. Each counter is loaded with the value written to the data register immediately. This value will then be decremented on the next active clock edge to arrive after the write. When the timer counter decrements to “0”, it will assert the appropriate interrupt. The timer counters can be read at any time. The clock source and mode is selectable by writing to various bits in the system control register. Clock sources are 508 kHz and 2 kHz. Both of these clock sources are synchronized to the main system AHB bus clock (HCLK).

Timer 3 (TC3) has the exact same operation as TC1 and TC2, but it is a 32-bit counter. It has the same register arrangement as TC1 and TC2, providing a load, value, control and clear register. The 16- and 32-bit timer counters can operate in two modes, free running mode or pre-load mode.

Copyright 2007 Cirrus Logic

18.1.2.1 Free Running Mode

In free running mode, counters TC1 and TC2 will wrap to 0xFFFF when they reach zero (underflow), and continue counting down. Counter TC3 will wrap to 0xFFFFFFFF when it underflows, and continues counting down.

18.1.2.2 Pre-load Mode

In pre-load (periodic) mode, the value written to the TC1, TC2 or TC3 Load registers is automatically re-loaded when the counter underflows. This mode can be used to generate a programmable periodic interrupt.

18.1.3 40-bit Timer Operation

The time stamp debug timer is a 40-bit up-counter used only for long term debugging (TC4). Its clock source is the 14.7456 MHz clock, divided by 15 to give a 983.04 kHz reference. The timer value may be read at any time by reading the lower 32-bit word first and then the high byte. Dividing the result by 983 yields a timestamp in milliseconds. The debug timer does not cause an interrupt. The timer is controlled by a single enable bit. When the timer is enabled, it begins counting from zero and when it is disabled, it is cleared back to zero. When it reaches its maximum value (0xFF_FFFF_FFFF) it wraps around to zero and continues counting upwards.

18.2 Registers

Table 18-1. Timers Register Map

Address	Read Location	Write Location	Size	Reset Value
0x8081_0000	"Timer1Load,"	"Timer1Load,"	16 bits	0
0x8081_0004	"Timer1Value,"	-	16 bits	0
0x8081_0008	"Timer1Control,"	"Timer1Control,"	8 bits	0
0x8081_000C	Reserved	"Timer1Clear,"	1 bit	-
0x8081_0020	"Timer1Load,"	"Timer2Load"	16 bits	0
0x8081_0024	"Timer2Value"	-	16 bits	0
0x8081_0028	"Timer2Control,"	"Timer2Control,"	8 bits	0
0x8081_002C	Reserved	"Timer2Clear,"	1 bit	-
0x8081_0060	"Timer4ValueLow"	-	32	0
0x8081_0064	Timer4Enable[a] / "Timer4ValueHigh"	Timer4Enable	9	0
0x8081_0080	"Timer3Load"	"Timer3Load"	32 bits	0
0x8081_0084	"Timer3Value"	-	32 bits	0
0x8081_0088	"Timer3Control"	"Timer3Load"	32 bits	0
0x8081_008C	Reserved	"Timer3Clear"	1 bit	-
0x8081_0010	Reserved	Reserved	-	-
0x8081_0030	Reserved	Reserved	-	-
0x8081_0040	Reserved	Reserved	-	-
0x8081_0090	Reserved	Reserved	-	-

a. "Enable" is a field in the "Timer4ValueHigh" register.

Copyright 2007 Cirrus Logic

Register Descriptions

Timer1Load,

Timer2Load

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load															

Address:

Timer1 - 0x8081_0000 - Read/Write
Timer2 - 0x8081_0020 - Read/Write

Reset Value:

0x0000_0000

Definition:

The Load register contains the initial value of the timer and is also used as the reload value in periodic timer mode. The timer is loaded by writing to the Load register when the timer is disabled. The Timer Value register is updated with the Timer Load value as soon as the Timer Load register is written. The Load register should not be written after the Timer is enabled because this causes the Timer Value register to be updated with an undetermined value.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

Load: Initial load value of the timer.

Timer3Load

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Load															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load															

Address:

Timer3 - 0x8081_0080 - Read/Write

Reset Value:

0x0000_0000

Copyright 2007 Cirrus Logic

Definition:

The Load register contains the initial value of the timer and is also used as the reload value in periodic timer mode. The timer is loaded by writing to the Load register when the timer is disabled. The Timer Value register is updated with the Timer Load value as soon as the Timer Load register is written to. The Load register should not be written to after the Timer is enabled as this causes the Timer Value register to be updated with an undetermined value.

Bit Descriptions:

Load: Initial load value of the timer.

Timer1Value,

Timer2Value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value															

Address:

Timer1 - 0x8081_0004 - Read Only
Timer2 - 0x8081_0024 - Read Only

Reset Value:

0x0000_0000

Definition:

The Value location gives the current value of the timer. When the Timer Load register is written to, the Value register is also updated with this Load value.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

Value: Current value of the timer.

Timer3Value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Value															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value															

Copyright 2007 Cirrus Logic

Address:

Timer3 - 0x8081_0084 - Read Only

Reset Value:

0x0000_0000

Definition:

The Value location gives the current value of the timer. When the Timer Load register is written to, the Value register is also updated with this Load value.

Bit Descriptions:

Value: Current value of the timer.

Timer1Clear,

Timer2Clear,

Timer3Clear

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															

Address:

Timer1 - 0x8081_000C - Write Only
Timer2 - 0x8081_002C - Write Only
Timer3 - 0x8081_008C - Write Only

Reset Value:

Not defined.

Definition:

Writing any value to the Clear location clears an interrupt generated by the timer.

Bit Descriptions:

RSVD: This register has no readable bits. It is just a write trigger.

Copyright 2007 Cirrus Logic

18

Timer1Control,

Timer2Control,

Timer3Control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								ENABLE	MODE	RSVD		CLKSEL	RSVD		

Address:

Timer1 - 0x8081_0008 - Read/Write
Timer2 - 0x8081_0028 - Read/Write
Timer3 - 0x8081_0088 - Read/Write

Reset Value:

0x0000_0000

Definition:

The Control register provides enable/disable and mode configurations for the timer.

Bit Descriptions:

RSVD:	Reserved. Unknown during a Read operation.
ENABLE:	Timer enable bit. This bit must be set to "1" to enable the timer. When the timer is disabled, its clock sources are turned off. Before re-enabling the timer, its Load register must be written to again.
MODE:	This bit sets the mode of operation of the timer. When set to 1, the timer is in periodic timer mode and when set to "0", the timer is in free running mode.
CLKSEL:	When set to "1", the 508 kHz clock is selected and when set to "0", the 2 kHz clock is selected.

Copyright 2007 Cirrus Logic

Timer4ValueLow

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Value															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value															

Address:

Timer4 - 0x8081_0060 - Read Only

Reset Value:

0x0000_0000

Definition:

This read-only register contains the low word of the time stamp debug timer (Timer4). When this register is read, the high byte of the Timer4 counter is saved in the Timer4ValueHigh register.

Bit Descriptions:

Value: Read Only Low Word of the Timer4 counter.

Timer4ValueHigh

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD							Enable	Value							

Address:

Timer4 - 0x8081_0064 - Read/Write

Reset Value:

0x0000_0000

Definition:

This is a 9-bit read/write register.

Enable is the only bit that matters during a register write. When set to "1", the timer is enabled and begins to count upwards. When set to 0, the debug timer registers are cleared to all zeros and the timer stops counting

Timer4ValueHigh is a read-only value and contains the high byte of the Timer4 counter. Note that the Timer4ValueLow register must first be read to store the high byte of the TC4 in Timer4ValueHigh register.

Copyright 2007 Cirrus Logic

Bit Descriptions:

RSVD:	Reserved. Unknown during a Read operation.
Enable:	Read/Write. Enable for Timer4.
Value:	Read only. High Byte of the Timer4 counter.

Copyright 2007 Cirrus Logic

Chapter 19

Watchdog Timer

19.1 Introduction

The Watchdog Timer provides a mechanism for generating a system-wide reset should the system hang. This functionality allows the Watchdog to recover the system and report the recovery to software. To prevent system-wide reset, software must periodically reset the Watchdog via an APB write operation. It is possible to disable the Watchdog through either hardware or software.

The Watchdog timer circuitry consists of a 7-bit counter. The most significant bit of the counter is used to trigger the **WATCHDOG_RESETn** output signal to the system control module for generating **HRESETn**.

The amount of time before a **WATCHDOG_RESETn** is initiated as well as the duration of the reset pulse is as follows:

- Time-out or **WATCHDOG_RESETn** duration = 64 / WATCHDOG_CLK frequency (units are seconds).
- For a 256 Hz WATCHDOG_CLK, time-out and reset pulse duration are 64 / 256 = 250 msec.

To keep the reset pulse from occurring, SW must reset the Watchdog timer (sometimes known as “kick the dog”) to a predetermined count on a periodic basis. This resets the counter, which prevents the **WATCH_RESETn** from activating. The counter is reset by writing 0x5555 to the Watchdog register. The Watchdog should be reset at least 2 WATCHDOG_CLK periods earlier than the time-out calculation would indicate, due to clock synchronization and handshaking circuitry.

Once a Watchdog reset occurs, the timer also provides a 250 ms duration reset pulse. The Watchdog also defaults to providing the pulse duration when the reset is from other sources such as user reset (external reset on **RSTOn**), AMBA bus reset (**HRESETn**), or power on reset (internal chip voltage detect power on signal **PWR_RESETn**). The reset pulse duration can be disabled by pulling the **CSn[2]** (**HW_RSTPULSE_DISABLEn**) signal low during the bus reset (**HRESETn** low). This immediately frees the Watchdog reset output line when reset becomes inactive. In either case, if the reset pulse duration is provided or not, the Watchdog counter will start over after the **WATCHDOG_RESETn** output becomes inactive. This begins a new 250 ms cycle after reset becomes inactive before software must reset the counter.

Copyright 2007 Cirrus Logic

19.1.1 Watchdog Activation

The Watchdog circuitry may be disabled via software for test purposes on products that do not wish to use a Watchdog timer by writing 0xAA55 to the Watchdog register. The Watchdog may also be re-enabled via software by writing 0xAAAA to the Watchdog register.

The Watchdog circuitry may be disabled via hardware on products that do not need to use a Watchdog timer, by applying an external pull down on the **CSn[1]** (**HW_WATCHDOG_DISABLEn**) signal during reset. This will allow the block to detect the presence of the resistor during the bus reset (**HRESETn** low) and disable the counter. During reset, the chip will disable the output driver and provide a weak pull-up resistor on this pad.

19.1.2 Clocking Requirements

The WATCHDOG_CLK for stepping the counter in the Watchdog has a frequency that is nominally 256 Hz, for generating a 250 ms time-out and a 250 ms reset pulse duration.

19.1.3 Reset Requirements

The Watchdog block has the following four reset inputs:

- **HRESETn**: This is the AHB bus reset signal from the Syscon block, which includes a software reset.
- **USR_RESETn**: This is the external user reset input, and its status is kept in register Watchdog[2].
- **PWR_RESETn**: This is the power-on-reset input for resetting everything including reset status bits. The power-on-reset is generated by a combination of the external **PRSTn** pin and the on chip voltage monitor/power up detector.
- **RESET_KEYS_DETECTED**: The Watchdog will time out if the three-key reset signal from the key scanning controller is activated. This input disables the ability to reset the Watchdog. If the Watchdog is hardware or software disabled, detection of the three-key reset will over-ride the Watchdog counter disable and cause the circuit to time out and generate the **WATCHDOG_RESETn** output anyway. It behaves as a **USR_RESETn** signal.

19.1.4 Watchdog Status

The Watchdog timer register can be read to determine the cause of a reset. The register contains user reset status (external reset on **RSTOn**), three-key reset status from the key scan controller, and Watchdog reset status bits (reset caused by **WATCHDOG_RESETn**). The state of these bits determines if the reset condition was the result of a user reset, a three-key reset, a power on reset, or a watch dog time-out. The status of these bits can only be cleared by a power on reset (internal chip voltage detect power on signal **PWR_RESETn**). An additional 7-bit status register is provided in the Watchdog module as WDSTAT. This status value is held through all resets but power on reset. The system can be reset by a three-key reset, a user reset, or a Watchdog reset without losing the contents of this register.

Copyright 2007 Cirrus Logic

Note: A software reset can reset the system without this register losing its contents.

19.1 Registers

Table 19-1. Watchdog Timer Register Memory Map

Address	Name	SW locked	Type	Size	Description
0x8094_0000	"Watchdog"	No	Read/Write	16/3 bits	Watchdog Control Register
0x8094_0004	"WDStatus"	No	Read/Write	7 bits	Watchdog Status Storage Register

Note: Watchdog registers are intended to be word-accessed only. Since the least significant bytes of the address bus are not decoded, byte and half word accesses are not allowed and may have unpredictable results.

Register Descriptions

Watchdog

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTL								CTL/PLSDSN	CTL/OVRID	CTL/SWDIS	CTL/HWDIS	CTL/URST	CTL/3KRST	CTL/WD	

Address:
0x8094_0000 - Read/Write

Default:
0x0000_0000

Definition:
Watchdog control register.

Bit Descriptions:

RSVD: Reserved. Unknown during read.

WRITE ONLY BIT FIELDS

CTL: Watchdog control bits. The ARM Core writes 0x5555 to this half-word to periodically restart the watchdog timer. Writing 0xAA55 to this hword will disable the watchdog timer. Writing 0xAAAA to this hword will re-enable the watchdog timer.

Copyright 2007 Cirrus Logic

READ ONLY BIT FIELDS

PLSDSN: Pulse Disable Not. The Watchdog internal PLSDIS bit monitors the **HW_PULSE_DISABLEn** latch status in the watchdog module. This provides status of the hardware pulse duration disable function. Active low means that the reset pulse is disabled.

OVRID: Software Override of the hardware watchdog disable. The OVRID bit monitors the SW_OVERIDE_HW_DISABLE register status in the watchdog module. This provides status of the watchdog software disable overriding the hardware disable function. This bit is active high when the software disable is overriding the hardware disable.

SWDIS: Software Watchdog Disable. The SWDIS bit monitors the SW_WATCHDOG_DISABLE register status in the watchdog module. This provides status of the watchdog software disable function. This bit is active high when the watchdog is software disabled.

HWDIS: Hardware Watchdog Disable. The HWDIS bit monitors the **HW_WATCHDOG_DISABLEN** latch status in the watchdog module. This provides status of the watchdog hardware disable function. This bit is active high when the watchdog is hardware disabled.

URST: User Reset Status flip flop. Read only. When "1", this bit indicates that the last reset was generated by the user reset signal (externally on **RSTOn**). This bit is not cleared by any resets other than power on reset, **PWR_RESETn**.

3KRST: Three-Key Reset Status flip flop. Read only. When "1", this bit indicates that the last reset signal was generated by a three-key reset from the key scan controller. This bit is not cleared by any resets other than power on reset, **PWR_RESETn**.

WD: Watchdog Reset Status flip flop. Read only. When "1", this bit indicates that the last reset was generated because of a watch dog time out. This bit is not cleared by any resets other than power on reset, **PWR_RESETn**.

Copyright 2007 Cirrus Logic

WDStatus

Address:

0x8094_0004 - Read/Write

Default:

0x0000_0000

Definition:

Watchdog status storage register. It can be used for storing your own status, and it can only be cleared by power-on-reset.

Bit Descriptions:

RSVD: Reserved. Unknown during read.

STAT: Watchdog Status bits. This is a watchdog status storage register that is not cleared by any resets other than power-on-reset and **PWR_RESETn**. The system can be reset by a three-key reset, a user reset, or a watchdog reset without losing the contents of this register.

Copyright 2007 Cirrus Logic

19

Copyright 2007 Cirrus Logic

Chapter 20

Real Time Clock With Software Trim

20.1 Introduction

The Real Time Clock (RTC) is a circuit that keeps track of the system date and time. The RTC operates from the normal device power supply and the 32 kHz input clock. The RTC circuit operates whenever power is applied to the device and the 32 kHz input clock is running.

The Real Time Clock section is composed of two blocks - Real Time Clock and the RTC TRIM.

The RTC module provides second level precision for internal time keeping. In addition, the block provides an interrupt based on a comparison register. The Real Time Clock block operates whenever power is applied and the 32 kHz input clock is running. However, the RTC cannot be used to wake the system via its interrupt when the system is in a STANDBY or HALT state, where PCLK is inactive.

The RTC TRIM block takes the 32,768 Hz clock from the RTC oscillator and creates a digitally compensated 1 Hz reference clock for use by the time keeping functions.

Note: The 32,768 Hz clock is referred to as the 32 kHz clock throughout the text.

20.1.1 Software Trim

The Real Time Clock oscillator software compensation circuitry allows software controlled digital compensation of a 32.768 kHz crystal oscillator. Typically, crystal oscillators must be externally compensated using discrete components. They are mechanically calibrated during manufacture. Software controlled digital compensation allows the oscillator to be electronically calibrated by automatic test equipment during manufacture. The circuit also enables readjustment in the field under software control.

The RTCSWComp register value is determined by manufacturing during board initialization to adjust the frequency of the 1 Hz clock. Refer to Section 20.1.1.1 for details on how to calculate the value in this register. This value is then stored in FLASH memory for retrieval when the product is first enabled in the field. The compensation values need to be restored once the RTC is permanently enabled for field use.

The compensation value consists of two parts: a counter preload value to act as an integer divider, (RTCSWComp.INT[15:0]), and the number of 32.768 kHz clocks to delete on a periodic interval (RTCSWComp.DEL[4:0]).

Copyright 2007 Cirrus Logic

20.1.1.1 Software Compensation

The 1 Hz clock is generated by running a programmable counter clocked by the 32.768 KHZ crystal oscillator reference. If the crystal reference and oscillator were perfect, a counter that counted 32768 clocks would provide a 1 Hz reference. However, the counter pre-load value is programmable to allow inaccuracies in the crystal and oscillator circuit. Simply allowing a different counter pre-load value only gives an accuracy of:

(½LSB / 32768 bits) x (3600 sec. / 1 hr) x (24hrs/day) x (30 days/month)
~= +/- 40 sec. per month

To further increase the accuracy, a fractional compensation is needed. This compensation mechanism provides a much better nominal RTC accuracy. The 1 Hz clock feeding the RTC is obtained by dividing the output of the 32.768 KHZ oscillator by an integer value. However, factors such as inaccuracy of the crystal, varying capacitance of the board traces, leads, and connections, etc., will cause the reference frequency to be inaccurate. This is corrected in software by adjusting the 1 Hz clock period through an integer compensation (by adjusting the counter preload) and with a fractional compensation (via deleting clocks at a fixed interval). By measuring the frequency of the reference crystal, and setting the RTCSWComp register value, the clock can be adjusted to a nominal accuracy of better than +/- 5 seconds per month.

20.1.1.2 Oscillator Frequency Calibration

Manufacturing can use a high precision frequency counter to measure the RTC 32.768 kHz reference clock via the **EGPIO[1]** pin when the RSTCR.RonG bit is set. This mode isolates the measurement of the oscillator circuit during manufacturing test to avoid disturbing the crystal reference frequency through added probe capacitance, etc. The compensation is accomplished by dividing the output of the oscillator by a integer value (with a pre-loadable counter) and then doing a fractional adjustment by periodically deleting clocks to the counter.

20.1.1.3 RTCSWComp Value Determination

After the true frequency of the oscillator is known, it is separated into integer and fractional portions. The integer portion of the frequency (less one) is set as the counter pre-load value. When the counter reaches zero, a carry pulse is generated and the counter is pre-loaded again. The carry pulse is used as the RTC 1 Hz signal reference.

The fractional part of the adjustment is done by deleting clocks from the clock stream feeding the integer counter. The period interval between deleting clocks is 32 seconds. The number of clocks deleted is set by RTCSWComp.DEL[4:0].

Copyright 2007 Cirrus Logic

20.1.1.4 Example - Measured Value Split Into Integer and Fractional Component

The manufacturing tester measures the oscillator output to be 33,455.870 Hz. For the integer portion, 33,455 - 32,768 is 687 cycles over the nominal frequency of the crystal. The integer pre-load value for the counter should always be chosen so that the actual clock frequency is faster than the value needed to generate a 1 Hz reference. Therefore, the RTCSWComp.INT[15:0] value is loaded with the binary equivalent of 33,455-1 or 0x82AE.

The fractional component of the oscillator output was measured to be 0.870 Hz. Software must adjust the clock so that the average number of cycles that are counted before generating one 1 Hz clock is 33 455-1.

Because the clock frequency is 0.870 Hz faster than the integer value, the 1 Hz clock generated by just the integer compensation is slightly faster than needed and may be slowed down by deleting clocks.

The fractional compensation value must be programmed to delete 0.870 Hz on average to bring the 1 Hz output frequency down to the proper value. Since the compensation procedure is performed only every 32 seconds, the value must be set to delete (0.870*32) = 27.84 which, when rounded, is 28 clocks every 32 seconds. The rounded 0.16 cycles per 32 seconds (or 0.005 Hz) represents the error in compensation. The RTCSWComp.DEL[4:0] fractional compensation value should be loaded with the hexadecimal equivalent of 28 - 1 or 0x1B.

20.1.1.5 Maximum Error Calculation vs. Real Time Clock Accuracy

The maximum error is 0.5½ clocks per 32 seconds. Therefore at 32.768 kHz, the maximum error is:

(0.5½ clock / 32 sec) x (1 sec / 32,768 clocks nominal) x
(2592000 sec/1month) = 1.24 seconds/month maximum error

To maintain an accuracy of +/- 5 seconds per month the required interval is calculated to be:

(5 seconds/1 month) x (1 month/2,592,000 seconds) = 1.93E-6

= (1 second/32,768 clocks) x (½clock / X-interval seconds)

X-interval = 7.9 seconds

Therefore to maintain a 5-second-per-month accuracy the compensation circuit only has to adjust within ½ of a 32.768 KHZ clock every 7.9 seconds. This could be done with a 3 bit clock delete value and an 8 second (3 bits clocked by 1 Hz) counter. However, the 1.24 second per month number is better and has been implemented in this device.

20.1.1.6 Real-Time Interrupt

To allow a Real Time Interrupt to be generated, VIC2 **INT[10]** has been connected to the 1 Hz clock. This interrupt should be configured as edge-triggered.

Copyright 2007 Cirrus Logic

20.1.2 Reset Control

The RTC block level reset operation is a bit complicated. The reset strategy is for the time-keeping part of the RTC to survive a system reset, and only be initialized by a power-on reset. The RTC interrupt enable is cleared by a user reset, so that a time count match (alarm interrupt) would disable with system reset.

The following register is initialized only by **PRSTn**: RTCSWComp

The following registers are initialized by **PRSTn**: RTCData, RTCMatch, RTCLoad, and RTCCtrl.

20

20.1 Registers

Table 20-1. Real Time Clock Register Memory Map

Address	Name	Description
0x8092_0000	"RTCData"	RTC Data Register
0x8092_0004	"RTCMatch"	RTC Match Register
0x8092_0008	"RTCSts"	RTC Status/EOI Register
0x8092_000C	"RTCLoad"	RTC Load Register
0x8092_0010	"RTCCtrl"	RTC Control Register
0x8092_0098	"RTCSWComp"	RTC Software Compensation

Register Descriptions

RTCData

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RTCDR															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCDR															

Address:
0x8092_0000 - Read Only

Default:
0x0000_0000

Definition:
RTC Data Register. Contains the 32 bit RTC counter value. This counter is incremented by the 1 Hz clock output from the RTC Trim module.

Bit Descriptions:

RTCDR: Counter value.

Copyright 2007 Cirrus Logic

RTCMatch

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RTCMR															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCMR															

Address:

0x8092_0004 - Read/Write

Default:

0x0000_0000

Definition:

RTC Match Register. Contain the 32 bit match value. When the RTCData value equals the RTCMatch value the RTC will generate an interrupt if the RTCCtrl.MIE bit is set to "1".

Bit Descriptions:

RTCMR: Match value.

RTCSts

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															INTR

Address:

0x8092_0008 - Read/Write

Default:

0x0000_0000

Definition:

RTC Interrupt Status and End Of Interrupt Register. Writing to this register clears the asserted interrupt.

Bit Descriptions:

RSVD: Reserved, unknown during read.

INTR: Interrupt status,
1 - RTC interrupt is asserted
0 - no interrupt.

Copyright 2007 Cirrus Logic

RTCLoad

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RTCLR															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCLR															

Address:

0x8092_000C - Read/Write

Default:

0x0000_0000

Definition:

RTC Load Register. Contains the 32 bit load value. Data written to this register is transferred to the RTCData on the next 1 Hz tick.

Bit Descriptions:

RTCLR: Load value.

RTCCtrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															MIE

Address:

0x8092_0010 - Read/Write

Default:

0x0000_0000

Definition:

RTC Interrupt Control Register. Contains the interrupt enable control bit.

Bit Descriptions:

RSVD: Reserved, unknown during read.

MIE: Match Interrupt Enable,
1 - RTC match interrupt is enabled
0 - interrupt disabled.

Copyright 2007 Cirrus Logic

RTCSWComp

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD										0	DEL				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT															

Address:

0x8092_0108 - Read/Write

Default:

0x0000_7FFF

Mask:

003F_FFFF

Definition:

RTC Software Compensation Register.

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
0:	Must be written as "0".
DEL:	Number of clocks to delete. This value determines the number of 32.768 KHZ clocks to delete every 32 seconds for compensating the oscillator. The value defaults to 0x0 which deletes no clock pulses.
INT:	Counter pre-load Integer value. This value is pre-loaded into the counter as the integer divide portion of the oscillator compensation. If set to 0x0000, no integer divide occurs and no clock pulses are deleted. The value defaults to 0x7FFF which causes the divider to divide by exactly 32,768 to generate a 1 Hz clock.

Copyright 2007 Cirrus Logic

20

Copyright 2007 Cirrus Logic

Chapter 21

I²S Controller

21

21.1 Introduction

The I²S controller is used to stream serial audio data between the external I²S CODECs', ADCs/DACs, and the ARM Core. It consists of 3 transmitter channels and 3 receiver channels. Each channel handles a single stereo stream. The transmitter and receiver are completely independent of each other and are programmed separately. Each channel (RX and TX) has its own set of addressable registers which allows access through the ARM APB or DMA accesses.

Figure 21-1 gives an architectural overview of the I²S controller. Table 21-1 lists the I²S controller.input and output signals.

The i2s_audioclk_mux section performs gating on the incoming audio clocks based on the settings within the TX and RX clock configuration registers and delivers a known clock definition to the rest of the I²S controller.

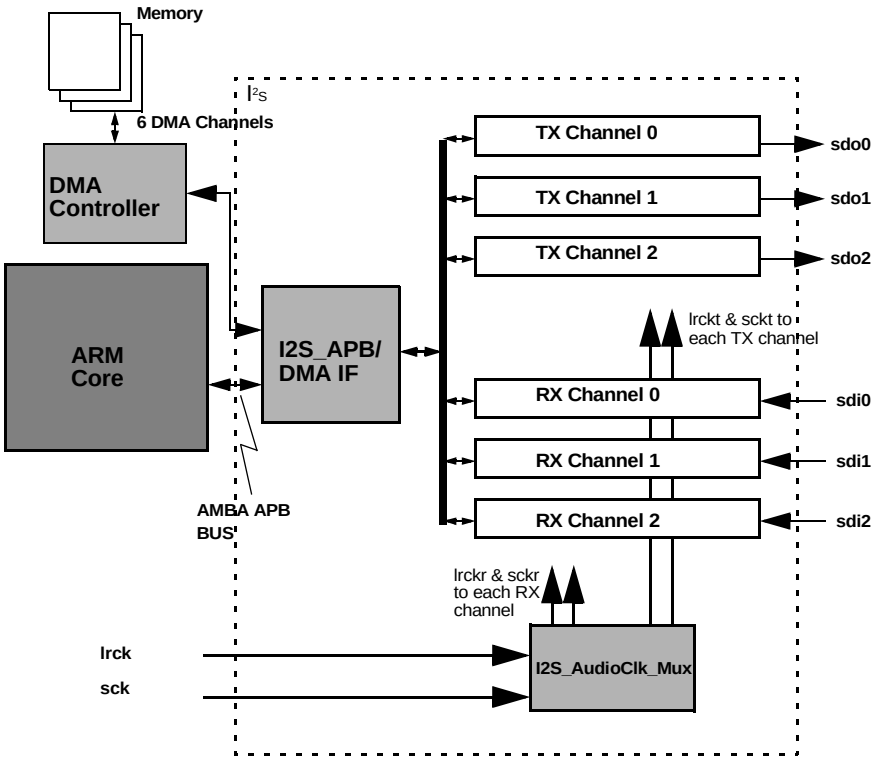


Figure 21-1. Architectural Overview of the I²S Controller

Copyright 2007 Cirrus Logic

21

Table 21-1. I²S Controller Input and Output Signals

Signal Name	Type	Description
lrck	IN	Left/right Word Audio slave clock.
sck	IN	Audio bit slave clock.
sdi0	IN	Serial data for channel 0
sdi1	IN	Serial data for channel 1
sdi2	IN	Serial data for channel 2
sdo0	OUT	Serial data output for TX channel 0
sdo1	OUT	Serial data output for TX channel 1
sdo2	OUT	Serial data output for TX channel 2

The primary I²S port and the I²S clocks are multiplexed and can be assigned to either the SSP pins or the Ac97 pins. The second and third I²S ports use the same clock pins as the primary I²S port, but their serial output and input pins are multiplexed with EGPIO pins. The second I²S port's serial output and serial input pins are multiplexed with EGPIO[4] and EGPIO[5] respectively and are enabled by setting DeviceCfg.A1onG. The third I²S port's serial output and serial input pins are multiplexed with EGPIO[6] and EGPIO[13] respectively and are enabled by setting DeviceCfg.A2onG.

Table 21-2. Audio Interfaces Pin Assignment

Pin Name	Normal Mode	I²S on SSP Mode	I²S on AC'97 Mode
	Pin Description	Pin Description	Pin Description
SCLK1	SPI Bit Clock	I²S Serial Clock	SPI Bit Clock
SFRM1	SPI Frame Clock	I²S Frame Clock	SPI Frame Clock
SSPRX1	SPI Serial Input	I²S Serial Input	SPI Serial Input
SSPTX1	SPI Serial Output	I²S Serial Output	SPI Serial Output
		(No I²S Master Clock)	
ARSTn	AC'97 Reset	AC'97 Reset	I²S Master Clock
ABITCLK	AC'97 Bit Clock	AC'97 Bit Clock	I²S Serial Clock
ASYNC	AC'97 Frame Clock	AC'97 Frame Clock	I²S Frame Clock
ASDI	AC'97 Serial Input	AC'97 Serial Input	I²S Serial Input
ASDO	AC'97 Serial Output	AC'97 Serial Output	I²S Serial Output

21.2 I²S Transmitter Channel Overview

Each I²S TX channel provides a single stereo I²S compliant output. The Transmit channel can operate in master or slave mode. Data is transferred between the ARM Core and the I²S controller via an interrupt based mechanism or DMA access. The ARM Core or host processor must write words in multiples of 2 (that is, a left and right stereo pair). These words are serially shifted out, timed with respect to the audio bit clock and word clock (SCLK and LRCK) that are generated (see Chapter 5, "Clock Control" on page 5-4 for additional details). The key features of the I²S transmitter are:

- Three transmit data channels, master or slave mode.

Copyright 2007 Cirrus Logic

21

- Supports 16/24/32 bit word lengths.
- Programmable left/right word clock polarity on the serial frame.
- Programmable Bit Clock polarity.
- Programmable data validity, that is, data valid on the rising/negative edge of the bit clock.
- Programmable first data bit position (I^2S or non-I^2S format).
- Programmable Left or Right data word justification
- Programmable data shift direction, that is, MSB or LSB transmitted first.
- Data underflow detection, that is, re-transmission of old data.
- Clock domain synchronization.
- DMA access.

Table 21-3. Transmitter FIFO's

	Right Sample 7	Byte 7	Byte 6	Byte 5	Byte 4
7	Left Sample 7	Byte 3	Byte 2	Byte 1	Byte 0
	Right Sample 6	Byte 7	Byte 6	Byte 5	Byte 4
:	:	:	:	:	:
:	:	:	:	:	:
	Right Sample 0	Byte 7	Byte 6	Byte 5	Byte 4
1	Left Sample 1	Byte 3	Byte 2	Byte 1	Byte 0
	Right Sample 0	Byte 7	Byte 6	Byte 5	Byte 4
0	Left Sample 0	Byte 3	Byte 2	Byte 1	Byte 0

Each channel has a 16 deep by 32bit wide FIFO where the ARM or DMA controller can write up to 8 sets of left/right data pairs before enabling the channel for transmission. In order to fill the FIFO the following sequence of events must be performed by the programmer. (**NOTE**: The following discussion is with respect to 1 channel only but applies to all.)

1. Enable I^2S controller: The I^2S global control register bit, I2SGlCtrl[0] must be written to in order to turn on the PCLK to the I^2S controller. The I^2S controller will not function correctly if this is not done.

2. Write to the FIFO: Once the I^2S controller is enabled, the TX FIFO may be written to by either the DMA or the ARM.

   Each FIFO is split up into 8 locations. Each location consists of 2 X 32bit register and can hold one left and one right stereo sample (16, 24 or 32 bits per sample). For APB accesses, the left and right samples must be written to different addresses: I2STX0Lft register address for left samples and I2STX0Rt register address for right samples (see register definitions).

   In order to fill a FIFO location, the programmer must write two data words, corresponding to left and right stereo data, to the FIFO. Only when both words are written by the programmer will the FIFO be loaded. Assuming this is the first FIFO write,

Copyright 2007 Cirrus Logic

these two words will occupy positions 0 and 1 in the FIFO. The FIFO now contains one complete left / right stereo sample. The words written by the programmer must always be right justified when writing 16-bit and 24-bit values.

If the programmer writes another left and right stereo sample to the I2STX0Lft and I2STX0Rt registers respectively, these words are loaded into the FIFO and will occupy positions 2 and 3. Subsequent writes will fill positions 4 and 5 and so on. The FIFO full flag is set when all 8 FIFO locations are filled by left / right sample pairs.

If an attempt is made to write another left / right stereo pair to the FIFO while it is full, the new samples are ignored and the FIFO overflow flag is set. (See "Register Descriptions" on page 448 on clearing this flag.) None of the existing FIFO locations are overwritten.

3. Enable the I²S transmit channel

Once the FIFO has been loaded, the channel enable I2STX0En one-bit register (see "I²S TX Register Descriptions" on page 21-13) is set. At this point, both the left and right stereo data from the 1st FIFO location are read by the I²S controller and copied into separate left and right holding registers. The left holding register is parallel loaded into a shift register and is serially shifted out the I²S sdo0 data line. This shifting out process is timed on the I²S audio word and bit clock. Once the left sample is shifted out, the right holding register is parallel loaded into the shift register and is serially shifted out the I²S sdo0 line.

If the I²S controller is programmed to transmit 16 or 24 bit words, the lower 16 or 24 bits are taken from the holding registers and loaded into the shift register. The upper bits are ignored by the I²S controller.

When the right sample is loaded into the shift register, the I²S controller reads the next left and right stereo samples from the 2nd FIFO location. After these have been loaded into the shift register in the same manner as above, FIFO location 3 is read and so on. After samples 15 and 16 (FIFO location 7) are taken from the FIFO, the FIFO read pointer will wrap around to location 0 and continue as before as long as the channel is enabled. If the I²S controller is disabled at any point, all FIFO locations are zeroed and the FIFO write and read pointers are reset.

If the transmit channel corresponding to the FIFO is disabled, the I²S controller will stop transmitting the current sample that is in the shift register. The data in the FIFO is not touched and the FIFO read and write pointers stay as they are. Upon re-enabling the channel, the I²S controller will advance the FIFO pointer, read the left and right stereo samples, and transmit them. The effect of this is that the data currently residing in the holding registers at the time the channel is disabled is lost.

To end transmission of data completely while there is data in the FIFO, first disable the corresponding channel. This action will ensure that the channels state machines are reset. The next step should be to disable the I²S controller, which will result in the FIFO's being reset. Any samples currently in the FIFO will be lost as a result.

Copyright 2007 Cirrus Logic

The I²S transmit and receive channels should be disabled before changes are made to the control registers. Once the new configuration has been set, the channels can be re-enabled following the specified start order.

If a channel is enabled while the FIFO is empty, no samples are read from the FIFO. The I²S controller will parallel load whatever is currently in the left holding register into the shift register. Once these contents have been shifted out, the right holding register is then parallel loaded into the shift register and then shifted out. If this occurs after the I²S controller has been reset, these holding registers will contain zero. If the I²S controller has been re-enabled after an earlier transmission, the holding registers will contain the last samples that were copied into them. As before, the I²S controller will attempt to read the FIFO after the right holding register has been loaded into the shift register. At this point, if the FIFO is still empty, the I²S controller will assert the FIFO underflow flag. No attempt is made to read the FIFO by the I²S controller and the read pointer stays pointing to location 0. The underflow will update a status bit in the Global Control Status register, I2SGlSts. (See "Register Descriptions" on page 448.) To clear the underflow the programmer must write at least one left and right stereo sample to the FIFO. Disabling the I²S controller will also clear the underflow.

The status of each FIFO is reflected in the Global Control Status register. There are 5 bits for each FIFO in this register that reflect the state of the FIFO. They are as follows:

- Tx0_underflow - Gets set when the I²S controller reads the FIFO when it is empty.
- Tx0_overflow - Gets set when the programmer attempts to write to the FIFO when it is full.
- Tx0_fifo_empty - Gets set when there no left and right stereo samples in the FIFO.
- Tx0_fifo_half_empty - Gets set when there are 4 left and right stereo samples or less in the FIFO.
- Tx0_fifo_full - Gets set when there are 8 left and right stereo samples in the FIFO.

21.3 I²S Receiver Channel Overview

The I²S Receiver channel enables audio compression algorithms executing on the ARM Core to receive stereo information from external CODECS.

Each I²S RX channel provides a single stereo I²S compliant input channel. The Receive channel can operate in master and slave mode. Data is received from the channel input and transferred into two registers, the left and right stereo pair. The ARM can then read the data from the channel. The key features are shown below.

- Three Receive data channels, master or slave mode.
- Supports 16/24/32 bit word lengths.
- Programmable left/right word clock polarity on the serial frame.
- Programmable bit clock polarity.
- Programmable data validity, that is, data valid on the rising/negative edge of the bit clock.

Copyright 2007 Cirrus Logic

- Programmable first data bit position. that is, I2S or non-I2S format.
- Programmable left or right data word justification.
- Programmable data shift direction, that is, MSB or LSB received first.
- Data overflow detection.
- Clock domain synchronization.
- DMA accesses.

The basic operation of the I2S receiver is that data is serially shifted in to form a pair of left / right words. This pair of words is written to a FIFO, which the ARM will read.

21.3.1 Receiver FIFO's

Each channel has a 16 deep by 32 bit wide FIFO where the ARM or DMA controller can read up to 8 sets of left / right data pairs. In order to receive left and right stereo data into the FIFO and read this data out from the FIFO, the following sequence of events must be performed by the programmer:

1. Enable the I2S controller.

   The I2S global control register bit, I2SGlCtrl[0], must be written to in order to turn on the PCLK to the I2S controller. The I2S controller will not function correctly if this is not done.

2. Enable the receive channel.

   The channel corresponding to the FIFO must be enabled in order for it to start sampling the data line. After being enabled, the I2S controller will wait until the start of the next incoming left stereo word as indicated by the audio word clock. When the start of the left word occurs, the I2S controller will sample the data line and load each bit into a dedicated left shift register. At the end of the left word and start of the right word as indicated by the audio word clock, the contents of the left shift register are loaded into a left data register. The I2S controller will continue to sample the data line loading each bit into a dedicated right shift register. At the end of the right word and start of the next left word, the contents of the right shift register are loaded into a right data register. One complete left and right stereo sample has now been received.

   At this point, the I2S controller signals to the FIFO that there is valid data ready to be written to the FIFO. The I2S controller will then write this stereo sample to FIFO location 0, which consists of 2 x 32 bit registers (assuming that this is the first sample to be received). The FIFO-empty bit in the I2S Global Control Status register (I2SGlSts) is now de-asserted. As more stereo sample pairs are received, they will be written to locations 1, 2, 3 and so on.

   The programmer can determine from the Global Control Status register if the FIFO has any valid left / right stereo samples. These samples are obtained from the FIFO via the APB by reading from the I2SRX0Lft and I2SRX0Rt registers. (See "Register

Copyright 2007 Cirrus Logic

Descriptions" on page 448.) Note that both left and right sample registers must be read for the I²S controller to consider the location to be free and modify the internal counter.

If the programmer attempts to read from the FIFO while it is empty, the contents that were last read from the FIFO will be put onto the APB bus. The FIFO read pointer is not updated and stays pointing to the same location. The FIFO underflow flag in the Global Control Status register is asserted. (See "Register Descriptions" on page 448.) If this happens to be the first attempted read by the programmer on the FIFO while the FIFO is still empty, the contents at FIFO location 0 are put onto the APB bus. These contents are zero if the I²S controller has been reset previously.

If the I²S controller signals to the FIFO that new stereo sample pairs have been received and the FIFO is full, the new samples are ignored. The existing contents in FIFO locations 0 to 7 are not touched. An internal Overflow bit is set, marking the FIFO pointer location at which the last good data was received (that is, at [current FIFO pointer location - 1]). When the FIFO pointer eventually points at this location again, after reading all 7 other FIFO locations, the FIFO overflow flag in the Global Control Status Register is asserted (and an interrupt is asserted, if enabled). The Status Register bit (and interrupt) is cleared by reading a left / right stereo sample pair from this FIFO location.

The data in the FIFO's is always right justified for word lengths of 16 and 24 bits. The upper bits will be set to zero by the I²S controller in this case.

The I²S transmit and receive channels should be disabled before modifying the control registers. Once the new configuration has been set, the channels can be re-enabled following the specified start order.

The status of each FIFO is reflected in the Global Control Status register. This register has 5 bits per FIFO that reflect the state of the FIFO. They are:

- Rx0_underflow - Gets set when the programmer reads the FIFO when it is empty.
- Rx0_overflow - Gets set when an Rx overflow has occurred, and the FIFO pointer is pointing at the last FIFO location where data was received before the overflow occurred
- Rx0_fifo_empty - Gets set when there are no left and right stereo samples in the FIFO.
- Rx0_fifo_half_full - Gets set when there are 4 left and right stereo samples or less in the FIFO.
- Rx0_fifo_full - Gets set when there are 8 left and right stereo samples in the FIFO.

21.4 I²S Master Clock Generation

The following information is required to generate a set of clocks for the I²S controller. The I²S port i2s_mstr_clk_cfg is used to supply the Syscon block the necessary control information in

Copyright 2007 Cirrus Logic

order to generate a set of audio clocks, LRCK (word clock) and SCLK (bit clock). The control bits required are:

- Master Mode Enable. (i2s_mstr_clk_cfg[0])
- Word Length Control (i2s_mstr_clk_cfg[2:1])
- Bit Clock Polarity (i2s_mstr_clk_cfg[3])
- Not Bit Clock Gating (i2s_mstr_clk_cfg[4]).
- Bit Clock Rate (i2s_mstr_clk_cfg[6:5])

These control bits come from the TX and the RX clock configuration registers and the word length registers. This control is sent out through the i2s_mstr_clk_cfg port of the I2S controller to the audio clock generator. The audio clock generator responds with the correct clock definition based on the settings received.

If both the TX and RX are required to be in master mode at the same time, both the RX and TX share the same master audio clocks. The following shows how i2s_mstr_clk_cfg is generated.

- If the Transmitter is enabled, the clock configuration information will always come from I2STXClkCfg register. Therefore, the I2SRXClkCfg (receiver clock configuration) register must be configured to be the same as the I2STXClkCfg (transmitter clock configuration) register in order to ensure correct operation of the receiver. The word lengths for both the TX and RX must be the same.
- If the Transmitter is disabled and the Receiver is required to be in master mode, then the i2s_mstr_clk_cfg output is generated from the I2SRXClkCfg register and the RX word length register.

Please note, the I2SClkDiv (Addr=0x8093_008C) register in the SYSCON block has an effect on I2S clock generation as well. The details are listed in Table 21-4. The controlling bit field for each function is determined by the ORIDE bit in the I2SClkDiv register (I2SClkDiv[29]). This table does not show the details of how to control this function. Please refer to each individual block for a detailed description.

Table 21-4. I2SClkDiv SYSCON Register Effect on I2S Clock Generation

Function	ORIDE=1	ORIDE=0
SCLK polarity	SPOL (I2SClkDiv[19])	i2s_mstr_clk_cfg[3]
SCLK Speed and Gating	DROP(I2SClkDiv[20]), SDIV(I2SClkDiv[16])	SCLK always is MCLK/2. SCLK is gated when i2s_mstr_clk_cfg[4]=0, i2s_mstr_clk_cfg[6:5]=0 and i2s_mstr_clk_cfg[2:1]=1, otherwise, SCLK is not gated.
LRCK Speed	LRDIV(I2SClkDiv[18:17])	i2s_mstr_clk_cfg[6:5]
Audio Slave Mode	SLAVE(I2SClkDiv[30])	i2s_mstr_clk_cfg[0]
Audio Clock (SCLK, LRCLK) Generation Enable	SENA(I2SClkDiv[31])	I2SonAC97 (DeviceCfg[6]) or I2SonSSP (DeviceCfg[7]). If either one is set, it enables the clock generation.

Copyright 2007 Cirrus Logic

Table 21-4. I2SClkDiv SYSCON Register Effect on I²S Clock Generation (Continued)

Function	ORIDE=1	ORIDE=0
Output Data Bit Align to SCLK Edge	When SPOL=1 and i2s_mstr_clk_cfg[3]=0, transition of output data bit and LRCK align to falling edge of SCLK When SPOL=0 and i2s_mstr_clk_cfg[3]=1, transition of output data bit and LRCK align to rising edge of SCLK;	The output data bit is always a half-cycle later to the SCLK edge which aligns to LRCK transition. If the SCLK rising edge is configured to align to the LRCK transition, then output data is aligned to falling edge of SCLK. If the SCLK falling edge aligns to the LRCK transition, then output data aligns to the SCLK rising edge.

21.5 I²S Bit Clock Rate Generation

Table 21-5. Bit Clock Rate Generation

Word Length	Bit Clock Rate (BCR[1:0])	not Bit Clock Gating (nBCG)	Actual bit clock rate with respect to LRCK
16	00	0 or 1	32x
24	00	0	64x with last 8 cycles gated off in each word.
24	00	1	64x Note the last 8 cycles are not gated off.
32	00	0 or 1	64x
Ignored	01	Ignored	Fixed at 32x
Ignored	10	Ignored	Fixed at 64x
Ignored	11	Ignored	Fixed at 128x

21.5.1 Example of the Bit Clock Generation.

For nBCG = 0 and BCR[1:0] = "10" the bit clock frequency is fixed at 64 times LRCK for word lengths of 32 and 24 and at 32x LRCK for word lengths of 16. In the case of 24 and 32 bit words, this 64x clock is then gating depending on the I²S controller word size. If the I²S controller word size is 32, then all of the 64x clock pulses are passed. If the I²S controller word size is 24, then the last 8 64x clock pulses are gated off in a LRCK cycle. For an I²S controller word size of 16 than all of the 32x clock pulses are passed. This is shown in Figure 21-2.

For other values of nBCG and BCR, the register bit descriptions define the bit clock operation.

Copyright 2007 Cirrus Logic

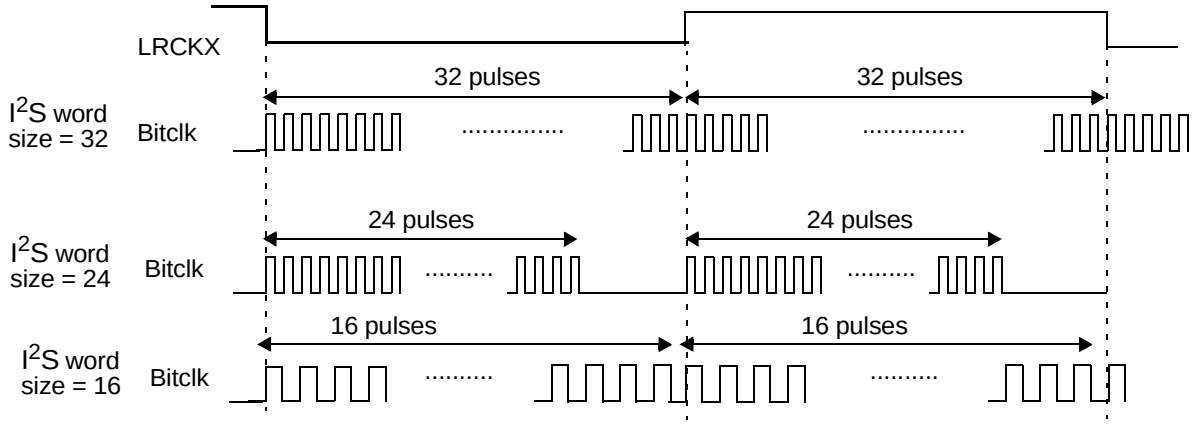


Figure 21-2. Bit Clock Generation Example

21.5.2 Example of Right Justified LRCK format

Figure 21-3 shows the frame format for Right Justified data. The word length is 16 in this case and the MSB is transmitted first. The bit clock rate is 64x so the for the first 16 clock cycles in each word there is no data as it is right justified in each word frame.

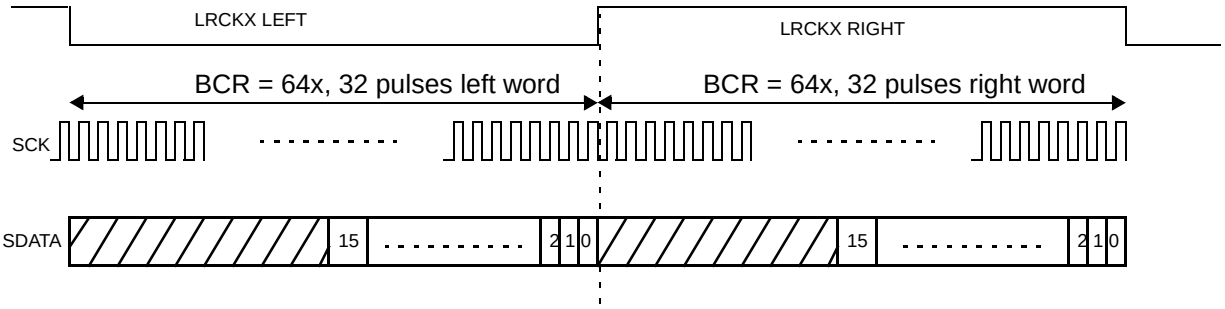


Figure 21-3. Frame Format for Right Justified Data

21.6 Interrupts

The I2S controller generates a single interrupt, I2SINTR to the ARM Core. This interrupt is a combination (logical OR) of all TX and RX internal interrupts.

The transmitter generates 4 internal interrupts within the I2S controller. Each of these reflect the status of the 3 individual TX FIFOs. These internal interrupts are as follows:

- TX0 FIFO empty.
- TX1 FIFO empty.

Copyright 2007 Cirrus Logic

- TX2 FIFO empty.
- TX underflow.

The first three can have their interrupt level determined by I2STXCtrl[0]. If this bit = 1, then the FIFO empty interrupt will occur when the FIFO is empty. If this bit = 0, then the FIFO empty interrupt will occur when the FIFO is half empty.

All four are combined and are maskable with the TX interrupt register enable bit, I2STXCtrl[1].

The FIFO empty internal interrupts are cleared if the FIFO's are filled with data or the corresponding channel is disabled.

The TX underflow internal interrupt is cleared by writing to both the left and right data registers of all enabled TX channels. This interrupt will also be cleared if the corresponding channel is disabled.

The I²S receiver generates 4 internal interrupts within the I²S controller. Each of these reflect the status of the 3 individual RX FIFOs. These internal interrupts are as follows:

- RX0 FIFO full.
- RX1 FIFO full.
- RX2 FIFO full.
- RX overflow.

The first three can have their interrupt level determined by I2SRXCtrl[0]. If this bit = 1, then the FIFO full interrupt will occur when the FIFO is full. If this bit = 0, then the FIFO full interrupt will occur when the FIFO is half full.

All four are combined and are maskable with the RX interrupt register enable bit, I2SRXCtrl[1].

The FIFO full internal interrupts are cleared if the FIFO's become less than full or the corresponding channel is disabled.

The RX overflow internal interrupt is cleared by reading both the left and right data registers of all enabled RX channels. This interrupt will also be cleared if the corresponding channel is disabled.

The RX and TX global interrupts are combined to form the I²S controller Interrupt, I2SINTR.

Table 21-6 summarizes which FIFO flags will generate interrupts when set. For example a transmitter FIFO empty flag will result in an interrupt but for a receiver FIFO empty flag a status bit only is set.

The sticky bits refer to bits I2SGISts[11:6]. A write of zero is required to clear the setting of these bits.

Copyright 2007 Cirrus Logic

Table 21-6. FIFO Flags

FIFO Flag	Transmitter	Receiver
FIFO empty	Interrupt and status bit	Status bit.
FIFO full	Status	Interrupt and status bit
FIFO overflow	Sticky bit	Interrupt and status bit
FIFO underflow	Interrupt and status bit	Sticky bit

21.7 Registers

21.7.1 I²S TX Registers

Table 21-7 summarizes the register set in the Transmitter. Each of the registers listed are addressable. The left and right data registers for channels 0, 1 and 2 can be accessed by both APB and DMA accesses. The remaining registers are concerned with control / status information and can be only accessed through the APB bus.

Table 21-7. I²S TX Registers

Address	Type	Width	Reset Value	Name	Description
0x8082_0010	R/W	32	0x0	I2STX0Lft	Left Transmit data register for channel 0
0x8082_0014	R/W	32	0x0	I2STX0Rt	Right Transmit data register for channel 0
0x8082_0018	R/W	32	0x0	I2STX1Lft	Left Transmit data register for channel 1
0x8082_001C	R/W	32	0x0	I2STX1Rt	Right Transmit data register for channel 1
0x8082_0020	R/W	32	0x0	I2STX2Lft	Left Transmit data register for channel 2
0x8082_0024	R/W	32	0x0	I2STX2Rt	Right Transmit data register for channel 2
0x8082_0028	R/W	3	0x0	I2STXLinCtrlData	Line Control data register
0x8082_002C	R/W	2	0x0	I2STXCtrl	Control register
0x8082_0030	R/W	2	0x0	I2STXWrdLen	Word Length
0x8082_0034	R/W	1	0x0	I2STX0En	TX0 Channel Enable
0x8082_0038	R/W	1	0x0	I2STX1En	TX1 Channel Enable
0x8082_003C	R/W	1	0x0	I2STX2En	TX2 Channel Enable

Copyright 2007 Cirrus Logic

I2S TX Register Descriptions

I2STX0Lft

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
i2s_tx0_left															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
i2s_tx0_left															

Address:

0x8082_0010 - Read/Write

Default:

0x0000_0000

Definition:

Transmit left data word for channel 0.

Bit Descriptions:

i2s_tx0_left: Transmit left data word for channel 0.

I2STX0Rt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
i2s_tx0_right															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
i2s_tx0_right															

Address:

0x8082_0014 - Read/Write

Default:

0x0000_0000

Definition:

Transmit right data word for channel 0.

Bit Descriptions:

i2s_tx0_right: Transmit right data word for channel 0.

Copyright 2007 Cirrus Logic

I2STX1Lft

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
i2s_tx1_left															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
i2s_tx1_left															

Address:

0x8082_0018 - Read/Write

Default:

0x0000_0000

Definition:

Transmit left data word for channel 1.

Bit Descriptions:

i2s_tx1_left: Transmit left data word for channel 1.

I2STX1Rt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
i2s_tx1_right															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
i2s_tx1_right															

Address:

0x8082_001C - Read/Write

Default:

0x0000_0000

Definition:

Transmit right data word for channel 1.

Bit Descriptions:

i2s_tx1_right: Transmit right data word for channel 1.

Copyright 2007 Cirrus Logic

I2STX2Lft

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
i2s_tx2_left															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
i2s_tx2_left															

Address:

0x8082_0020 - Read/Write

Default:

0x0000_0000

Definition:

Transmit left data word for channel 2.

Bit Descriptions:

i2s_tx2_left: Transmit left data word for channel 2.

I2STX2Rt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
i2s_tx2_right															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
i2s_tx2_right															

Address:

0x8082_0024 - Read/Write

Default:

0x0000_0000

Definition:

Transmit right data word for channel 2.

Bit Descriptions:

i2s_tx2_right: Transmit right data word for channel 2.

Copyright 2007 Cirrus Logic

I2STXLinCtrlData

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												Left_Right_Justify	TXUF_REPEAT_SAMPLE		TXDIR

Address:

0x8082_0028 - Read/Write

Default:

0x0000_0000

Definition:

Line Control Data Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

Left_Right_Justify: Determines how the data word is justified when being transmitted on the sdo line output.
0 - left justified.
1 - right justified

TXUF_REPEAT_SAMPLE:On TX underflow, the I2S controller transmits all zeros if this bit is "1".
If this bit is "0" the I2S controller repeats the last sample on underflow.

TXDIR: Transmit data shift direction.
0 - MSB first
1 - LSB first

I2STXCtrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD														TXUFIE	TXEMPTY_int_level

Address:

0x8082_002C - Read/Write

Default:

0x0000_0000

Copyright 2007 Cirrus Logic

Definition:

Transmit Control Register

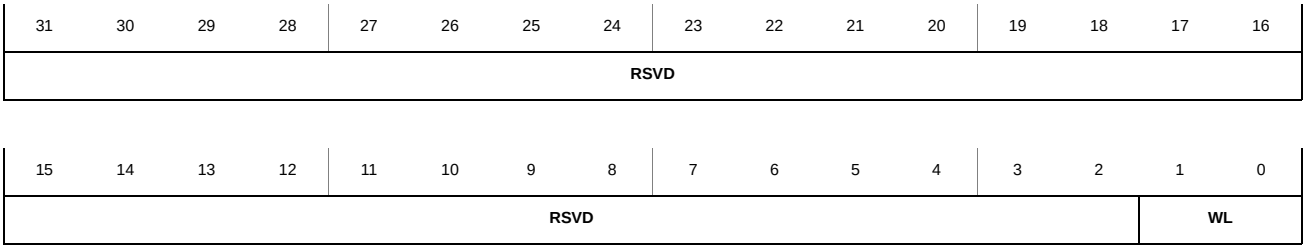
Bit Descriptions:

RSVD: Reserved. Unknown During Read.

TXUFIE: Transmit interrupt enable. Active high

TXEMPTY_int_level: Transmit empty interrupt level select.
0 - Generate interrupt when FIFO is half empty.
1 - Generate interrupt when FIFO is empty.

I2STXWrdLen



Address:

0x8082_0030 - Read/Write

Default:

0x0000_0000

Definition:

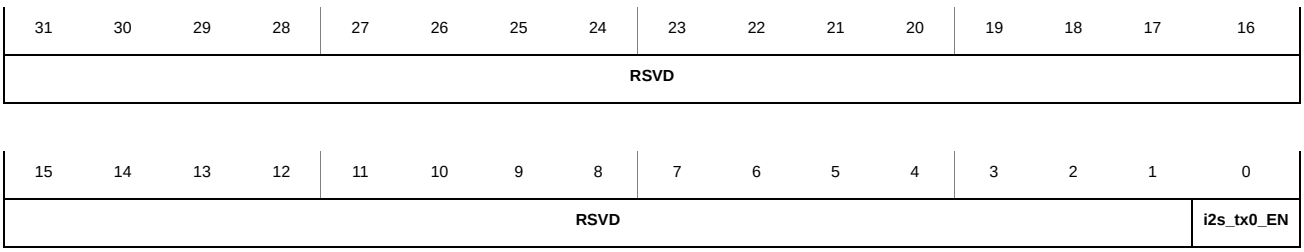
Transmit Word Length

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

WL: Transmit Word Length.
00 - 16 bit mode
01 - 24 bit mode
10 - 32 bit mode

I2STX0En



Address:

0x8082_0034 - Read/Write

Copyright 2007 Cirrus Logic

Default:

0x0000_0000

Definition:

TX0 Channel Enable

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

i2s_tx0_EN: TX0 Channel Enable

I2STX1En

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															i2s_tx1_EN

Address:

0x8082_0038 - Read/Write

Default:

0x0000_0000

Definition:

TX1 Channel Enable

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

i2s_tx1_EN: TX1 Channel Enable

I2STX2En

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															i2s_tx2_EN

Address:

0x8082_003C - Read/Write

Default:

0x0000_0000

Copyright 2007 Cirrus Logic

Definition:

TX2 Channel Enable

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

i2s_tx2_EN: TX2 Channel Enable

21.7.2 I²S RX Registers

The following table summarizes the register set in the I²S Receiver block. Each of the registers listed are addressable. The left and right data registers for channels 0, 1 and 2 can be accessed by both APB and DMA accesses. The remaining registers are concerned with control/status information and can be only accessed through the APB bus.

Table 21-8. I²S RX Registers

Address	Type	Width	Reset Value	Name	Description
0x8082_0040	R	32	0x0	I2SRX0Lft	Left Receive data register for channel 0
0x8082_0044	R	32	0x0	I2SRX0Rt	Right Receive data register for channel 0
0x8082_0048	R	32	0x0	I2SRX1Lft	Left Receive data register for channel 1
0x8082_004C	R	32	0x0	I2SRX1Rt	Right Receive data register for channel 1
0x8082_0050	R	32	0x0	I2SRX2Lft	Left Receive data register for channel 2
0x8082_0054	R	32	0x0	I2SRX2Rt	Right Receive data register for channel 2
0x8082_0058	R/W	2	0x0	I2SRXLinCtrlData	Line Control data register
0x8082_005C	R/W	2	0x0	I2SRXCtrl	Control register
0x8082_0060	R/W	2	0x0	I2SRXWrdLen	Word Length
0x8082_0064	R/W	1	0x0	I2SRX0En	RX0 Channel Enable
0x8082_0068	R/W	1	0x0	I2SRX1En	RX1 Channel Enable
0x8082_006C	R/W	1	0x0	I2SRX2En	RX2 Channel Enable

I²S RX Register Descriptions

I2SRX0Lft

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
i2s_rx0_left															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
i2s_rx0_left															

Address:

0x8082_0040 - Read Only

Default:

Copyright 2007 Cirrus Logic

0x0000_0000

Definition:

Receive left data word for channel 0.

Bit Descriptions:

i2s_rx0_left: Receive left data word for channel 0.

I2SRX0Rt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
i2s_rx0_right															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
i2s_rx0_right															

Address:

0x8082_0044 - Read Only

Default:

0x0000_0000

Definition:

Receive right data word for channel 0.

Bit Descriptions:

i2s_rx0_right: Receive right data word for channel 0.

I2SRX1Lft

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
i2s_rx1_left															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
i2s_rx1_left															

Address:

0x8082_0048 - Read Only

Default:

0x0000_0000

Definition:

Receive left data word for channel 1.

Bit Descriptions:

i2s_rx1_left: Receive left data word for channel 1.

Copyright 2007 Cirrus Logic

I2SRX1Rt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
i2s_rx1_right															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
i2s_rx1_right															

Address:

0x8082_004C - Read Only

Default:

0x0000_0000

Definition:

Receive right data word for channel 1.

Bit Descriptions:

i2s_rx1_right: Receive right data word for channel 1.

I2SRX2Lft

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
i2s_rx2_left															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
i2s_rx2_left															

Address:

0x8082_0050 - Read Only

Default:

0x0000_0000

Definition:

Receive left data word for channel 2.

Bit Descriptions:

i2s_rx2_left: Receive left data word for channel 2.

Copyright 2007 Cirrus Logic

I2SRX2Rt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
i2s_rx2_right															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
i2s_rx2_right															

Address:

0x8082_0054 - Read Only

Default:

0x0000_0000

Definition:

Receive right data word for channel 2.

Bit Descriptions:

i2s_rx2_right: Receive right data word for channel 2.

I2SRXLinCtrlData

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD														Left_Right_Justify	RXDIR

Address:

0x8082_0058 - Read/Write

Default:

0x0000_0000

Definition:

Receive Line Control Data Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read. Must be written as "0".

Left_Right_Justify: Receiver Data word Justification when being received on the SDI line input.
0 - Left justification.
1 - Right justification.

Copyright 2007 Cirrus Logic

RXDIR: Receive data shift direction.
0 - MSB first
1 - LSB first

I2SRXCtrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD														ROFLIE	RXFull_int_level

Address:
0x8082_005C - Read/Write

Default:
0x0000_0000

Definition:
Control Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read. Must be written as "0".

ROFLIE: Receive interrupt enable.
Active high

RXFull_int_level: Rx full interrupt level select.
0 - Generate interrupt when FIFO is half full.
1 - Generate interrupt when FIFO is full.

I2SRXWrdLen

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															WL

Address:
0x8082_0060 - Read/Write

Default:
0x0000_0000

Definition:
Word Length

Copyright 2007 Cirrus Logic

Bit Descriptions:

RSVD: Reserved. Unknown During Read. Must be written as "0".

WL: Receive Word Length.
00 - 16 bit mode
01 - 24 bit mode
10 - 32 bit mode

21

I2SRX0En

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															i2s_rx0_EN

Address:

0x8082_0064 - Read/Write

Default:

0x0000_0000

Definition:

RX0 Channel Enable

Bit Descriptions:

RSVD: Reserved. Unknown During Read. Must be written as "0".

i2s_rx0_EN: RX0 Channel Enable

I2SRX1En

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															i2s_rx1_EN

Address:

0x8082_0068 - Read/Write

Default:

0x0000_0000

Definition:

RX1 Channel Enable

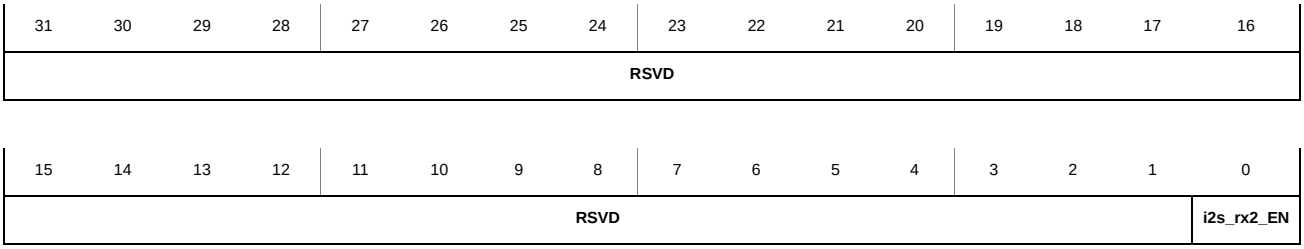
Copyright 2007 Cirrus Logic

Bit Descriptions:

RSVD: Reserved. Unknown During Read. Must be written as "0".

i2s_rx1_EN: RX1 Channel Enable

I2SRX2En



Address:

0x8082_006C - Read/Write

Default:

0x0000_0000

Definition:

RX2 Channel Enable

Bit Descriptions:

RSVD: Reserved. Unknown During Read. Must be written as "0".

i2s_rx2_EN: RX2 Channel Enable

21.7.3 I²S Configuration and Status Registers

Table 21-9. I²S Configuration and Status Registers

Address	Type	Width	Reset Value	Name	Description
0x8082_0000	R/W	7	0x0	I2STXClkCfg	Transmitter clock configuration register.
0x8082_0004	R/W	7	0x0	I2SRXClkCfg	Receiver clock configuration register
0x8082_0008	R/W	20	0x12492	I2SGISts	I²S Global Status register. This reflects the status of the 3 RX FIFOs and the 3 TX FIFOs.
0x8082_000C	R/W	2	0x0	I2SGICtrl	I²S Global Control register.

Copyright 2007 Cirrus Logic

I2S Configuration and Status Register Descriptions

I2STXClkCfg

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									i2s_tx_bcr		i2s_tx_nbcg	i2s_mstr	i2s_trel	i2s_tckp	i2s_tlrs

Address:

0x8082_0000 - Read/Write

Default

0x0000_0000

Definition:

Transmitter clock configuration register.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

i2s_tx_bcr: Defines the TX bit clock rate.
00 - I2STXClkCfg[4] defines the bit clock generation.
01 - Bit clock rate is fixed at 32x. Word length is ignored.
10 - Bit clock rate is fixed at 64x. Word length is ignored.

11 - Bit clock rate is fixed at 128x. Word length is ignored.

i2s_tx_nbcg: Defines TX not bit clock gating mode.

If I2STXClkCfg[5:6] = 00, this bit defines the bit clock rate, otherwise ignored.

Bit clock rate = 32x if word length is 16.
Bit clock rate = 64x if word length is 32.
Bit clock rate = 64x if word length is 24.

There is a special case when the word length is 24.
If this bit = 0 and the word length is 24, the last 8 cycles are gated off in each word.
If this bit = 1 and the word length is 24, the last 8 cycles are not gated off in each word.

Copyright 2007 Cirrus Logic

i2s_mstr: Defines if the TX Audio clocks are slave or master.
0 - slave mode.
1 - master mode.

i2s_trel: Determines the timing of the lrckt with respect to the sdox data outputs.

0 - Transition of lrckt occurs together with the first data bit.
1 - Transition of lrckt occurs one bitclk cycle before the first sdox data bit. This is I2S format.

i2s_tckp: Defines polarity of the TX bitclk.

1 - Positive clock polarity. The lrckt and sdox lines change synchronously with the positive edge of the bitclk and are considered valid during negative transitions.
0 - Negative clock polarity. The lrckt and sdox lines change synchronously with the negative edge of the bitclk and are considered valid during positive transitions.

i2s_tlrs: Defines the polarity of lrckt.

0 - if lrckt is low, then it is the left word, if lrckt is high, then it is the right word.
1 - if lrckt is low, then it is the right word, if lrckt is high, then it is the left word.

I2SRXClkCfg

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									i2s_rx_bcr		i2s_rx_nbcg	i2s_mstr	i2s_rrel	i2s_rckp	i2s_rlrs

Address:
0x8082_0004 - Read/Write

Default:
0x0000_0000

Definition:
Receiver clock configuration register.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

Copyright 2007 Cirrus Logic

i2s_rx_bcr: RX bit clock rate.
00 - I2SRXClkCfg[4] defines the bit clock generation.
01 - Bit clock rate is fixed at 32x. Word length is ignored.
10 - Bit clock rate is fixed at 64x. Word length is ignored.
11 - Bit clock rate is fixed at 128x. Word length is ignored.

i2s_rx_nbcg: Defines RX not bit clock gating mode.

If I2SRXClkCfg[5:6] = 00, this bit defines the bit clock rate, otherwise ignored.

Bit clock rate = 32x if word length is 16.
Bit clock rate = 64x if word length is 32.
Bit clock rate = 64x if word length is 24.

There is a special case when the word length is 24.
If this bit = 0 and the word length is 24, the last 8 cycles are gated off in each word.
If this bit = 1 and the word length is 24, the last 8 cycles are not gated off in each word.

i2s_mstr: Defines if the RX Audio clocks are slave or master.
0 - slave mode.
1 - master mode.

i2s_rrel: Determines the timing of the lrckr with respect to the sdix data inputs.
0 - Transition of lrckr occurs together with the first data bit.
1 - Transition of lrckr occurs one bitclk cycle before the first sdix data bit.

i2s_rckp: Defines polarity of the RX bitclk.
1 - Positive clock polarity. The lrckr and sdix lines change synchronously with the positive edge of the bitclk and are considered valid during negative transitions.
0 - Negative clock polarity. The lrckr and sdix lines change synchronously with the negative edge of the bitclk and are considered valid during positive transitions.

i2s_rlrs: Defines the polarity or lrckr.
0 - if lrckr is low then it is the left word, if lrckr is high then it is the right word.
1 - if lrckr is low then it is the right word, if lrckr is high then it is the left word.

Copyright 2007 Cirrus Logic

21.7.4 I²S Global Status Registers

I²S Global Status Registers

I2SGISts

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD		rx2_fifo_half_full	rx2_fifo_empty	rx2_fifo_full	tx2_fifo_half_empty	tx2_fifo_empty	tx2_fifo_full	rx1_fifo_half_full	rx1_fifo_empty	rx1_fifo_full	tx1_fifo_half_empty	tx1_fifo_empty	tx1_fifo_full	rx0_fifo_half_full	rx0_fifo_empty

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rx0_fifo_full	tx0_fifo_half_empty	tx0_fifo_empty	tx0_fifo_full	Rx2_underflow	Rx1_underflow	Rx0_underflow	Tx2_overflow	Tx1_overflow	Tx0_overflow	Rx2_overflow	Rx1_overflow	Rx0_overflow	Tx2_underflow	Tx1_underflow	Tx0_underflow

Address:

0x8082_0008 - Read/Write

Default:

0x0001_2492

Definition:

UART Data Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

Tx0_underflow: when = 1, TX0 FIFO has underflowed.

Tx1_underflow: when = 1, TX0 FIFO has underflowed.

Tx2_underflow: when = 1, TX0 FIFO has underflowed.

Rx0_overflow: when = 1, RX0 FIFO has overflowed and the FIFO pointer is currently pointing at the last data received before the overflow occurred.

Rx1_overflow: when = 1, RX1 FIFO has overflowed and the FIFO pointer is currently pointing at the last data received before the overflow occurred.

Rx2_overflow: when = 1, RX2 FIFO has overflowed and the FIFO pointer is currently pointing at the last data received before the overflow occurred.

Tx0_overflow: when = 1, the tx0 FIFO is full and an attempt has been made to write data to it by the APB or DMA. This bit is cleared by writing a 0 to it.

Copyright 2007 Cirrus Logic

Tx1_overflow: when = 1, the tx1 FIFO is full and an attempt has been made to write data to it by the APB or DMA. This bit is cleared by writing a 0 to it.

Tx2_overflow: when = 1, the tx2 FIFO is full and an attempt has been made to write data to it by the APB or DMA. This bit is cleared by writing a 0 to it.

Rx0_underflow: when = 1, the rx0 FIFO is empty and an attempt has been made to read data from it by the APB or DMA. This bit is cleared by writing a 0 to it.

Rx1_underflow: when = 1, the rx1 FIFO is empty and an attempt has been made to read data from it by the APB or DMA. This bit is cleared by writing a 0 to it.

Rx2_underflow: when = 1, the rx2 FIFO is empty and an attempt has been made to read data from it by the APB or DMA. This bit is cleared by writing a 0 to it.

tx0_fifo_full: when = 1, FIFO is full, otherwise not full

tx0_fifo_empty: when = 1, FIFO is empty, otherwise not empty

tx0_fifo_half_empty: when = 1, FIFO is half empty, otherwise less than half empty

rx0_fifo_full: when = 1, FIFO is full, otherwise not full

rx0_fifo_empty: when = 1, FIFO is empty, otherwise not empty

rx0_fifo_half_full: when = 1, FIFO is half full, otherwise less than half full

tx1_fifo_full: when = 1, FIFO is full, otherwise not full

tx1_fifo_empty: when = 1, FIFO is empty, otherwise not empty

tx1_fifo_half_empty: when = 1, FIFO is half empty, otherwise less than half empty

rx1_fifo_full: when = 1, FIFO is full, otherwise not full

rx1_fifo_empty: when = 1, FIFO is empty, otherwise not empty

rx1_fifo_half_full: when = 1, FIFO is half full, otherwise less than half full

tx2_fifo_full: when = 1, FIFO is full, otherwise not full

tx2_fifo_empty: when = 1, FIFO is empty, otherwise not empty

tx2_fifo_half_empty: when = 1, FIFO is half empty, otherwise less than half empty

rx2_fifo_full: when = 1, FIFO is full, otherwise not full

Copyright 2007 Cirrus Logic

rx2_fifo_empty: when = 1, FIFO is empty, otherwise not empty

rx2_fifo_half_full: when = 1, FIFO is half full, otherwise less than half full

I2SGlCtrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD														i2s_loopback	i2s_ife

Address:

0x8082_000C - Read/Write

Default:

0x0000_0000

Definition:

I2S Global Control Register

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

i2s_ife: Defines if I2S controller is enabled and PCLK is turned on for the I2S controller.
0 - PCLK is off.
1 - PCLK is on.

i2s_loopback: Defines loopback operation.
0 - not in loopback mode
1 - Loopback mode selected.

The I2S global register deals with enabling the block and whether loopback mode is used. The I2S enable bit determines whether the PCLK is turned on for the I2S. All registers except for the data registers can be written without the I2S PCLK enabled. The ARM provides its own clock cycles when writing to any of the control status registers.

When the I2S controller is required to transmit or receive data, PCLK must be turned on via this register.

The I2S controller loopback mode bit determines if TX channel 0 is connected to RX channel 0. This will allow data be sent in a loop fashion from the transmitter back through the receiver. This applies to all channels, with TX1 looped to RX1, and TX2 looped to RX2. When loopback is active, data at the receiver input is ignored and transmit data is sent out normally. The transmit section will control the clock configuration during loopback the same as if full-duplex operation was used.

Copyright 2007 Cirrus Logic

21

Copyright 2007 Cirrus Logic

Chapter 22

AC’97 Controller

22

22.1 Introduction

The AC’97 Controller includes a 5-pin serial interface to an external audio codec. The AC-Link is a bi-directional, fixed rate, serial PCM (Pulse Code Modulation) digital stream, dividing each audio frame into 12 outgoing and 12 incoming data streams (slots), each with 20-bit sample resolution.

The AC’97 Controller contains logic that controls the AC-Link to the audio codec and an interface to the AMBA APB.

The main features of the AC’97 are:

- Serial-to-parallel conversion on data received from the external codec.
- Parallel-to-serial conversion on data transmitted to the external codec.
- Reception / Transmission of control and status information.
- Supports up to 4 different sampling rates at a time with 4 transmit and 4 receive channels. The transmit and receive paths are buffered with internal FIFO memories allowing data to be stored independently in both transmit and receive modes. The data for the FIFOs can be written via either the APB interface or the DMA channels (1-3).

Table 22-1 lists the input and output signals for the AC’97 controller.

Table 22-1. AC’97 Input and Output Signals

Signal Name	Input/Output	Description
SDATAIN	Input	Serial input data stream from the audio codec. It contains status information and digital audio input streams.
BITCLK	Input	Clock from serial codec. Fixed at 12.288 MHz.
SDATAOUT	Output	This serial output transmits the control information and digital audio output streams to the audio codec.
SYNC	Output	Synchronization signal to the external codec. Fixed at 48 kHz. It is also output asynchronously when the audio codec is in warm reset state.
RESET	Output	Asynchronous cold reset (active low, resets codec registers).

The AC’97 pins are multiplexed and may be used for the I^2S controller instead of AC'97 by setting DeviceCfg.I2SonAC97.

The AC’97 Controller can support up to four different sampling rates at a time. To allow the controller to support all slots per frame, it has been assumed that the sampling rate for each different type of data are the same. For example, all audio data are at the same sampling rate

Copyright 2007 Cirrus Logic

22

and all modem data are at the same sampling rate. If the external codec supported the following channels: PCM LEFT, PCM RIGHT, MODEM1, PCM CENTRE, PCM L SURROUND, PCM R SURROUND, PCM LFE, MODEM2 and HSET, then the user would have to program the transmit side of the controller so that all the audio data was in channel 1, modem data in channel 2, and the HSET data in channel 3. The controller could also receive MIC data at a different rate. This data would have to be stored in channel 4. The controller is designed to allow any slot data to be stored into any channel the user wishes. If the external codec supports more than 4 sample rates, the user will have to determine which sample rates to allow.

The controller has four channels, which consist of a transmit FIFO, receive FIFO and their associated control logic. The control logic can be configured to allow the FIFOs to accept any data to or from any slot in a frame.

The receive part of each channel is controlled via its AC97RXCR register. This register controls the following:

- Which slot data from the received frame is to be stored in the FIFO. The controller will not store any other slots than those specified in these registers. The user must ensure that all slot data stored in the FIFO is at the same sampling rate.
- The length of time before a timeout interrupt is generated.
- Whether the FIFO is enabled or not.
- The number of bits in the slot that is captured.
- Whether the channel is enabled to receive data or not.

The transmit part of each channel is controlled via its AC97TXCR register. This register controls the following:

- Which slot the data in the FIFO is to be transmitted in, the user must ensure that all the data in the FIFO is intended for slots with the same sampling rate. The data must be supplied lowest slot number first.
- Whether the FIFO is enabled or not.
- The number of bits that need to be appended to the data from the CPU to make the word 20 bits.
- Whether the channel is enabled to transmit data or not.
- The transmit channel also supports variable sample rates via the Data Request Disable Slots from the external codec in slot 1. The data request bits for all audio and modem data are expected to occur at the same time.

Slot 0 for transmission is determined by the controller depending on the values in the AC97TXCR register, the data request bits, and the FIFO having valid data to send. If a slot does not have any data for transmission, the controller will fill the slot with zeros and set the Tag bits as invalid.

If the external codec does not support the Data Request Disable bits/Variable Rate Extension the bits will always be "0" meaning a sample rate of 48 kHz. As slots 1 and 2 are always

Copyright 2007 Cirrus Logic

transmitted at 48 kHz, the external codec does not have Data Request Disable bits for these slots. Data for transmission on slots 1, 2, and 12 can be obtained from either the channels or the registers SLOT1RXTX, AC97S2Data and AC97S12Data. However, consistent usage of one of these two methods should be maintained.

If the slot enable bits are set when receiving the data for slots 1, 2, and 12, the data is stored in the channel and not the SLOT1/2/12RX registers. If the slot enable bits are not set, then the data will always go to the registers.

The user should only use the channels to transmit slot 1 and 2 data when they are setting the external codec up for operation. Once the set up of the external codec is complete, the data for slot 1 and 2 should come via the SLOT1/2TX registers. This action frees up the channel.

22.2 Interrupts

The AC'97 Controller generates individual maskable active HIGH interrupts. Each interrupt may be enabled or disabled using the appropriate enable bit. Setting the bit HIGH enables the corresponding interrupt. This allows for a system interrupt controller to provide the mask registers for each interrupt.

The status of the individual interrupt sources can be read from appropriate register. The interrupts are ORed to create one interrupt (**AC97INTR**) for the AC'97 controller block

22.2.1 Channel Interrupts

The individual interrupts that are generated by each transmit/receive channel are described below. The status of the interrupts can be read from the AC97RISRx or AC97ISRx registers, and is masked in the AC97IEx register.

22.2.1.1 RIS

If the receive FIFO is enabled and the mask bit RIE is set, the FIFO receive interrupt is asserted when the AC'97 Controller receive FIFO is greater than or equal to half full. The receive interrupt is cleared when the FIFO becomes less than half full.

If the receive FIFO is disabled, it has a depth of one location. Any data received will fill that one location, causing the receive interrupt to be asserted high. The receive interrupt is cleared by performing a single read of the receive FIFO.

22.2.1.2 TIS

If the transmit FIFO is enabled and the mask bit TIE is set, the FIFO transmit interrupt is asserted when the AC'97 Controller transmit FIFO is at least half-empty. The FIFO transmit interrupt is cleared by filling the transmit FIFO more than half full.

If the transmit FIFO is disabled (has a depth of one location) and there is no data present in the transmitters single location, the transmit interrupt is asserted high. The transmit interrupt is cleared by performing a single write to the transmit FIFO.

Copyright 2007 Cirrus Logic

22.2.1.3 RTIS

The receive timeout interrupt is asserted when the receive FIFO is not empty and no further data is received over a number of frames. This number is set by the TOC value in the AC97RXCR register. The receive timeout interrupt is cleared when the FIFO becomes empty through reading all the data.

22.2.1.4 TCIS

The transmit complete interrupt is asserted when the transmit FIFO is empty and the parallel to serial shifter is empty. This indicates that there is no data left in the FIFOs to be sent.

22.2.2 Global Interrupts

The individual interrupts that are global for the AC97 controller are described below. The status of these interrupts can be read from the AC97GIS or AC97RGIS registers, and are masked in the AC97IM register.

22.2.2.1 CODECREADY

The Codec Ready Interrupt is asserted when the codec has indicated that it is ready by setting bit15 of Slot0.

This interrupt is cleared by writing a "1" to the appropriate bit of the AC97EOI register.

22.2.2.2 WINT

The Wake-up interrupt is asserted when a wake-up event will trigger the assertion of **SDATAIN** while the AC-Link is powered down. The wake-up is caused by the external codec's GPIO pins, which have been configured to generate a wake-up event via the codec's GPIO pin Wake-up Control register (0x52). An AC-Link wake-up interrupt is defined as a 0-to-1 transition on SDATAIN when the AC-Link is powered down. The controller knows when the external codec has been powered down as the SLOT1/2TX registers are monitored to check for this condition. When the wake up event has been detected on the SDATAIN line, an interrupt is generated to allow the ARM Core to reactivate the link with either a warm or cold reset.

This interrupt is cleared by writing a "1" to the appropriate bit of the AC97EOI register.

22.2.2.3 GPIOINT

The receive GPIOINT interrupt is asserted when bit 0 in slot 12 of the incoming **SDATAIN** is "1". This bit indicates that one or more of the bits in slot 12 have changed since the last frame. It is up to the interrupt service routine to read the AC97S12Data register in order to clear this interrupt. The external codec's register (0x54) GPIO pin Status reflects the state of all of the GPIO pins.

Copyright 2007 Cirrus Logic

22.2.2.4 GPIOTXCOMPLETE

The transmit GPIOTXCOMPLETE interrupt is asserted when all values written to the AC97S12Data have been transmitted. It is cleared when any data is written to the AC97S12Data.

22.2.2.5 SLOT2INT

The receive SLOT2INT interrupt is asserted when the AC97S2Data register has new data that has not been read. By reading the data in the AC97S2Data register the SLOT2INT interrupt is cleared.

22.2.2.6 SLOT1TXCOMPLETE

The transmit SLOT1TXCOMPLETE interrupt is asserted when all values written to the AC97S1Data have been transmitted. It is cleared when any data is written to the AC97S1Data.

22.2.2.7 SLOT2TXCOMPLETE

The transmit SLOT2TXCOMPLETE interrupt is asserted when all values written to the AC97S2Data have been transmitted. It is cleared when any data is written to the AC97S2Data.

22.3 System Loopback Testing

A loopback test mode is available for system testing so that data transmitted on **SDATAOUT** can also be received on **SDATAIN**. Loopback mode is entered when a "1" is written to the LOOP bit in AC97GCR register. For normal operation the LOOP bit must always be "0", which is also the default state at reset.

Note: For this test mode to work, an external bit clock will need to be supplied.

22.4 Registers

Table 22-2. AC'97 Register Memory Map

Address	Type	Name	Description
0x8088_0000	Read/Write	AC97DR1	Data read or written from/to FIFO1
0x8088_0004	Read/Write	AC97RXCR1	Control register for receive
0x8088_0008	Read/Write	AC97TXCR1	Control register for transmit
0x8088_000C	Read	AC97SR1	Status register
0x8088_0010	Read	AC97RISR1	Raw interrupt status register
0x8088_0014	Read	AC97ISR1	Interrupt Status
0x8088_0018	Read/Write	AC97IE1	Interrupt Enable
0x8088_001C	-	-	Reserved
0x8088_0020	Read/Write	AC97DR2	Data read or written from/to FIFO2
0x8088_0024	Read/Write	AC97RXCR2	Control register for receive
0x8088_0028	Read/Write	AC97TXCR2	Control register for transmit
0x8088_002C	Read	AC97SR2	Status register

Copyright 2007 Cirrus Logic

22

Table 22-2. AC'97 Register Memory Map (Continued)

Address	Type	Name	Description
0x8088_0030	Read	AC97RISR2	Raw interrupt status register
0x8088_0034	Read	AC97ISR2	Interrupt Status
0x8088_0038	Read/Write	AC97IE2	Interrupt Enable
0x8088_003C	-	-	Reserved
0x8088_0040	Read/Write	AC97DR3	Data read or written from/to FIFO3.
0x8088_0044	Read/Write	AC97RXCR3	Control register for receive
0x8088_0048	Read/Write	AC97TXCR3	Control register for transmit
0x8088_004C	Read	AC97SR3	Status register
0x8088_0050	Read	AC97RISR3	Raw interrupt status register
0x8088_0054	Read	AC97ISR3	Interrupt Status
0x8088_0058	Read/Write	AC97IE3	Interrupt Enable
0x8088_005C	-	-	Reserved
0x8088_0060	Read/Write	AC97DR4	Data read or written from/to FIFO4.
0x8088_0064	Read/Write	AC97RXCR4	Control register for receive
0x8088_0068	Read/Write	AC97TXCR4	Control register for transmit
0x8088_006C	Read	AC97SR4	Status register
0x8088_0070	Read	AC97RISR4	Raw interrupt status register
0x8088_0074	Read	AC97ISR4	Interrupt Status
0x8088_0078	Read/Write	AC97IE4	Interrupt Enable
0x8088_007C	-	-	Reserved
0x8088_0080	Read/Write	AC97S1Data	Data received/transmitted on SLOT1
0x8088_0084	Read/Write	AC97S2Data	Data received/transmitted on SLOT2
0x8088_0088	Read/Write	AC97S12Data	Data received/transmitted on SLOT12
0x8088_008C	Read/Write	AC97RGIS	Raw Global interrupt status register
0x8088_0090	Read	AC97GIS	Global interrupt status register
0x8088_0094	Read/Write	AC97IM	Interrupt mask register
0x8088_0098	Write	AC97EOI	End Of Interrupt register
0x8088_009C	Read/Write	AC97GCR	Main Control register
0x8088_00A0	Read/Write	AC97Reset	RESET control register.
0x8088_00A4	Read/Write	AC97SYNC	SYNC control register.
0x8088_00A8	Read	AC97GCIS	Global channel FIFO interrupt status register.

Register Descriptions

AC97DRx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD / DATA (See Definition, below.)												DATA			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															

Copyright 2007 Cirrus Logic

Address:

AC97DR1 - 0x8088_0000 - Read/Write
AC97DR2 - 0x8088_0020 - Read/Write
AC97DR3 - 0x8088_0040 - Read/Write
AC97DR4 - 0x8088_0060 - Read/Write

Definition:

The AC97DR registers are read / write data registers that are normally 20 bits wide. In 16-bit compact mode, all 32 available bits are used. This register is zero at reset.

22

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

DATA: Write - Transmit FIFO: The AC97TXCR register qualifies the data within the TX FIFO.
Read - Receive FIFO: The AC97RXCR register qualifies the data within the FIFO.

For words to be transmitted:

- If the FIFOs are enabled, data written to this location is pushed onto the transmit FIFO.
- If the FIFOs are not enabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO).

For received words:

- If the FIFOs are enabled, the data received is pushed onto the receive FIFO.
- If the FIFOs are not enabled, the data received is stored in the receiving holding register (the bottom word of the receive FIFO).

The receive FIFO is 21 bits wide. The 21st bit, the receive overrun error status, can only be read via the AC97ISR registers. The receive overrun error status bit is transferred down to the FIFO buffer along with the overrun data value.

AC97RXCRx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				TOC											FDIS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CM	RSIZE		RX12	RX11	RX10	RX9	RX8	RX7	RX6	RX5	RX4	RX3	RX2	RX1	REN

Address:

AC97RXCR1 - 0x8088_0004 - Read/Write
AC97RXCR2 - 0x8088_0024 - Read/Write
AC97RXCR3 - 0x8088_0044 - Read/Write
AC97RXCR4 - 0x8088_0064 - Read/Write

Copyright 2007 Cirrus Logic

22

Definition:

Receive Control Registers. The AC97RXCR registers are read/write registers that are 32 bits. The data contained within the register controls the data slots that are contained within the receive FIFO. The data contained within the RSIZE bits controls the number of zeros that are to be appended to data to make it 20 bits.

Should two channels be enabled for the same data slot, then data is taken from, or given to, the lower channel number.

The data from the receive channel is stored in the lowest slot first. If for example the receive FIFO is setup to store slots 3 and 4 then the first data word out of the FIFO will be slot 3 followed by slot 4.

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
TOC:	Time out count value. The FIFOs have the capability of generating a timeout interrupt when the receive FIFO is not empty and no further data is received for a period of time. This time period is specified by the value written here. The value is the number of frames that must occur without any data being received (a count of the SYNC signal). A write of "0" to this value disables the counter, and no timeout interrupt is generated. On reset the value is "0". The maximum count of 4096 will allow the timeout period to be set to 85 msec.
FDIS:	FIFO Disable 0 - The FIFO buffers are Enabled (FIFO mode). 1 - The FIFO is disabled (character mode). That is, the FIFO becomes 1-byte-deep holding registers.
CM:	Compact mode enable. If the RSIZE value is either "00" or "11" (setting the data word size to 12- or 16-bits) then the CM bit determines whether the two data words are compacted into one 32-bit word, or each is sent in a separate word. If the RSIZE value is either "01" or "10" (setting the data word size to 18- or 20-bits) then the CM bit has no effect. See Table 22-3. 0 - The data is justified into separate 32 bit words 1 - The two data words are compacted into one 32-bit word for reading by the CPU.

Copyright 2007 Cirrus Logic

RSIZE: Determines how many bits to a data word. See Table 22-3 for details of the interaction between RSIZE and CM.
00 data is 16 bits
01 data is 18 bits
10 data is 20 bits
11 data is 12 bits

Table 22-3. Interaction Between RSIZE and CM

CM	RSIZE		Data to CPU
0	0	0	Justified, one 16 bits
0	1	1	Justified, one 12 bits
1	0	0	Compacted, two 16 bits
1	1	1	Compacted, two 12 bits
X	1	0	Justified, 20 bit
X	0	1	Justified, 18 bit

RX12: FIFO stores SLOT12 data (takes precedence over AC97S12Data)

RX11: FIFO stores SLOT11 data

RX10: FIFO stores SLOT10 data

RX9: FIFO stores SLOT9 data

RX8: FIFO stores SLOT8 data

RX7: FIFO stores SLOT7 data

RX6: FIFO stores SLOT6 data

RX5: FIFO stores SLOT5 data

RX4: FIFO stores SLOT4 data

RX3: FIFO stores SLOT3 data

RX2: FIFO stores SLOT2 data

RX1: FIFO stores SLOT1 data

REN: A "1" written to this bit enables the receive for this FIFO and enables the PCLK for the respective channel.

Copyright 2007 Cirrus Logic

AC97TXCRx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															FDIS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CM	TSIZE		TX12	TX11	TX10	TX9	TX8	TX7	TX6	TX5	TX4	TX3	TX2	TX1	TEN

Address:

AC97TXCR1 - 0x8088_0008 - Read/Write
AC97TXCR2 - 0x8088_0028 - Read/Write
AC97TXCR3 - 0x8088_0048 - Read/Write
AC97TXCR4 - 0x8088_0068 - Read/Write

Definition:

Transmit Control Registers. The AC97TXCR registers are read/write. The data contained within the register controls the data slots that are contained within the FIFO's transmit register. The data within this FIFO must be of the same sampling frequency, such as all audio slot data at 44.1 kHz. This register is used to create slot 0 for transmitting. If this register specifies that the data within is for Slot1 and 2, this will take precedence over the data in the SLOT1TX and SLOT2TX register. If Slot 1 and 2 data is to be sent via this FIFO, it will always be transmitted at 48kHz. Therefore, it is advisable not to enable any other slots unless they too are sampled at 48kHz.

The data contained within the TSIZE bits controls the number of zeros that are to be appended to data to make it 20 bits.

Should two channels be enabled for the same data slot, then data is taken from/given to the lower channel number.

The data into the FIFO is stored in the lowest slot first. For example if the FIFO is set up to store in slots 3 and 4, then slot 3 is the first data into the FIFO and slot 4 the second.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

FDIS: FIFO Disable
0 - The FIFO buffers are Enabled (FIFO mode).
1 - The FIFO is disabled (character mode). That is, the FIFO becomes 1-byte-deep holding registers.

Copyright 2007 Cirrus Logic

CM: Compact mode enable. If the RSIZE value is either "00" or "11" (setting the data word size to 12- or 16-bits) then the CM bit determines whether the two data words are compacted into one 32-bit word, or each is sent in a separate word. If the RSIZE value is either "01" or "10" (setting the data word size to 18- or 20-bits) then the CM bit has no effect. See Table 22-4.
0 - The data is justified into one 32 bit word
1 - The two data words are compacted into one 32-bit word for reading by the CPU.

RSIZE: Determines how many bits to a data word. See Table 22-4 for details of the interaction between RSIZE and CM.
00 data is 16 bits
01 data is 18 bits
10 data is 20 bits
11 data is 12 bits

Table 22-4. Interaction Between RSIZE and CM Bits

CM	RSIZE		Data to CPU
0	0	0	Justified, one 16 bits
0	1	1	Justified, one 12 bits
1	0	0	Compacted, two 16 bits
1	1	1	Compacted, two 12 bits
X	1	0	Justified, 20 bit
X	0	1	Justified, 18 bit

TX12: FIFO stores SLOT12 data (takes precedence over AC97S12Data)

TX11: FIFO stores SLOT11 data

TX10: FIFO stores SLOT10 data

TX9: FIFO stores SLOT9 data

TX8: FIFO stores SLOT8 data

TX7: FIFO stores SLOT7 data

TX6: FIFO stores SLOT6 data

TX5: FIFO stores SLOT5 data

TX4: FIFO stores SLOT4 data

TX3: FIFO stores SLOT3 data

TX2: FIFO contains SLOT2 data (only use if sampling rate is 48 kHz). Takes precedence over AC97S2Data.

Copyright 2007 Cirrus Logic

TX1: FIFO contains SLOT1 data (only use if sampling rate is 48 kHz). Takes precedence over AC97S1DATA.

TEN: A "1" written to this bit enables the transmit for this FIFO and enables the PCLK for the respective Channel.

AC97SRx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									TXUE	RXOE	TXBUSY	TXFF	RXFF	TXFE	RXFE

Address:

AC97SR1 - 0x8088_000C - Read Only
AC97SR2 - 0x8088_002C - Read Only
AC97SR3 - 0x8088_004C - Read Only
AC97SR4 - 0x8088_006C - Read Only

Definition:

Status Registers. The AC'97 Controller status registers are read only registers that give information about the transmit/receive status of the block. After reset, the TXFF, RXFF and TXBUSY are "0", and TXFE and RXFE are "1".

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

TXUE: TX Underrun Error - This bit is set to "1" if an underrun error has been detected (if data is to be transmitted and the FIFO is empty).

This bit is cleared to "0" by writing to the AC97DR register.

Note: Bit will only be set if FIFO had been written to at least once in current data transfer.

RXOE: RX Overrun Error - This bit is set to "1" if an overrun error has been detected. This bit is set to "1" if data is received and the FIFO is already full.

This bit is cleared to "0" by reading the AC97DR register.

Copyright 2007 Cirrus Logic

TXBUSY: TXBUSY is set when TEN = "1" AND there is data in the FIFO, OR when data from this FIFO is being sent in the current frame.

TXBUSY is cleared at the start of the next frame following the assertion of the corresponding channel's TXFE flag (the value of TEN is irrelevant).

TXFF: Transmit FIFO full flag, active HIGH.
This bit is asserted HIGH if the transmit FIFO is full.

RXFF: Receive FIFO full flag, active HIGH.
This bit is asserted HIGH if the receive FIFO is full.

TXFE: Transmit FIFO empty flag, active HIGH.
This bit is asserted HIGH if the transmit FIFO is empty.

RXFE: Receive FIFO empty flag, active HIGH.
This bit is asserted HIGH if the receive FIFO is empty.

AC97RISRx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												RIS	TIS	RTIS	TCIS

Address:

AC97RISR1 - 0x8088_0010 - Read Only
AC97RISR2 - 0x8088_0030 - Read Only
AC97RISR3 - 0x8088_0050 - Read Only
AC97RISR4 - 0x8088_0070 - Read Only

Definition:

Raw Interrupt Status. The AC97ISR registers are the raw Interrupt status registers for the controller FIFOs. All bits are cleared to zero on reset except for the TCIS as the FIFO and shift register should both be empty. Any write to this register clears the overrun error.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RIS: RX Interrupt Status - This bit is set to "1" if the receive FIFO becomes half full.

TIS: TX Interrupt Status - This bit is set to "1" if the transmit FIFO becomes half empty.

Copyright 2007 Cirrus Logic

RTIS: RX Timeout Interrupt Status - If this bit is set to "1", the timeout FIFO interrupt is asserted.

TCIS: TX complete Interrupt Status - If this bit is set to "1", the transmit FIFO complete interrupt is asserted.

AC97ISRx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												RIS	TIS	RTIS	TCIS

Address:

AC97ISR1 - 0x8088_0014 - Read Only
AC97ISR2 - 0x8088_0034 - Read Only
AC97ISR3 - 0x8088_0054 - Read Only
AC97ISR4 - 0x8088_0074 - Read Only

Definition:

Interrupt Status Register. The AC97ISR registers are the Interrupt status registers for the controller FIFOs. All bits are cleared to zero on reset except for the TCIS as the FIFO and shift register should both be empty.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RIS: RX Interrupt Status - If this bit is set to "1", the receive FIFO interrupt is asserted.

TIS: TX Interrupt Status - If this bit is set to "1", the transmit FIFO interrupt is asserted.

RTIS: RX Timeout Interrupt Status - If this bit is set to "1", the timeout FIFO interrupt is asserted.

TCIS: TX complete Interrupt Status - If this bit is set to "1", the transmit FIFO complete interrupt is asserted.

Copyright 2007 Cirrus Logic

AC97IEx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												RIE	TIE	RTIE	TCIE

Address:

AC97IE1 - 0x8088_0018 - Read/Write
AC97IE2 - 0x8088_0038 - Read/Write
AC97IE3 - 0x8088_0058 - Read/Write
AC97IE4 - 0x8088_0078 - Read/Write

Definition:

Interrupt Enable Register. The AC97IE registers control the Interrupt Enables for the FIFOs within the controller. All bits are cleared on reset.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

RIE: Receive Interrupt Enable - If this bit is set to "1", the FIFO receive interrupt is enabled.

TIE: Transmit Interrupt Enable - If this bit is set to "1", the FIFO transmit interrupt is enabled.

RTIE: Receive Timeout Interrupt Enable - If this bit is set to "1", the FIFO receive timeout interrupt is enabled.

TCIE: Transmit Complete Interrupt Enable - If this bit is set to "1", the FIFO transmit complete interrupt is enabled.

AC97S1Data

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								DATA							

Address:

0x8088_0080 - Read/Write

Definition:

Slot 1 Data Register. The AC97S1Data register is a read / write register. When a write has occurred to this register, the data contained within it is sent on the

Copyright 2007 Cirrus Logic

next available frame in SLOT1. As both the AC97S1Data and AC97S2Data data are required for writes to the external codec, the AC97S2Data data will only become valid for transmission when the AC97S1Data has been written also. In order to perform a write to the external codec, the AC97S1Data register must be written to after the AC97S2Data register is written.

Bit[19] of Slot1 on SDATAOUT from the AC'97 indicates whether a read or a write is being performed to or from the external codec. This bit is generated automatically in the AC'97 as follows:

- When data is written to the AC97S2Data register, the read / write bit is set to "0"
- The read / write bit is set to "1" when the slot 2 data has been transmitted
- The read / write bit is set to "1" when a read occurs from either the AC97S1Data or AC97S2Data register

If a power down is required, then the software must write the address 0x26 to this location (for the external device), which will be recorded by the controller. If the AC97S2Data bit 12 is set, then the controller will go into power down mode.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

DATA: Read operation: Read data value of the last value written to this register via the AC-Link interface. The data contained within it is the last valid received slot 1, for example, the slot 0 received tagged slot1 as valid.

Write operation: Write data value to transmit on slot1 on the next available frame. Once the data has been transmitted, it will be marked as invalid.

AC97S2Data

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															

Address:

0x8088_0084 - Read/Write

Definition:

Slot 2 Data Register. The AC97S2Data register is a read / write register. When a write has occurred to this register, the data contained within it will be sent on the next available frame in SLOT2. In order to perform a write to the external

Copyright 2007 Cirrus Logic

codec, the AC97S2Data register must be written to before the AC97S1Data register is written.

If a power down is required, then the software must write to SLOT1TX location address 0x26, which is recorded by the controller. If the AC97S2Data bit 12 is set, then the controller will go into power down mode.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

DATA: Read operation: Read data value of the last value written to this register via the AC-Link interface.

Write operation: Write data value to transmit on slot 2 on next available frame. Once the data has been transmitted, it will marked as invalid.

AC97S12Data

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD												DATA			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															

Address:

0x8088_0088 - Read/Write

Definition:

Slot 12 Data Register. The AC97S12Data register is a read / write register. Data written to it will be sent on the next available frame in SLOT 12. When this register is read, the data contained within it is the data that was last received for SLOT 12.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

DATA: Read operation: Read data value of the last value received in SLOT 12. Bit 0 is monitored to see if a GPIOINT has occurred.

Write operation: Write data value to transmit on slot 12 on next available frame. Once the data has been transmitted it will marked as invalid.

Copyright 2007 Cirrus Logic

AC97RGIS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									SLOT2TX COMPLETE	CODEC READY	WINT	GPIO INT	GPIOTX COMPLETE	SLOT2RX VALID	SLOT1TX COMPLETE

Address:

0x8088_008C - Read Only

Definition:

Raw Global Interrupt Status Register. The AC'97 raw global interrupt status register is a read/write register that gives the status of various functions outside of the FIFO functionality within the controller.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

SLOT2TXCOMPLETE: Set when the AC97S2Data register has completed transmission. This bit is cleared when data is in the register to be transmitted.

CODECREADY: This bit is set to "1" during a wakeup when the codec indicates that it is ready by setting bit 15 of Slot0. It is cleared by writing to Bit1 of the AC97EOI Register.

WINT: RAW Wake-up Interrupt Status. If this bit is set to "1". The RAW Wake-up interrupt is asserted. This bit is cleared with a write to the AC97EOI register.

GPIOINT: The GPIOINT shows the raw status of the GPIOINT bit (slot 12 bit 0) in the receive frame, which is stored in the AC97S12Data register. This bit is cleared when the AC97S12Data register is read.

GPIOTXCOMPLETE: GPIO Transmission Complete. Set when a new value to the AC97S12Data register has completed transmission. Cleared when data is placed in the register to be transmitted.

SLOT2RXVALID: The AC97S2Data register has new data that has not been read. Reading the data in the AC97S2Data register clears this bit.

SLOT1TXCOMPLETE: Set when the AC97S1Data register has completed transmission. This bit is cleared when data is written to the AC97S1Data register to be transmitted.

Copyright 2007 Cirrus Logic

AC97GIS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									SLOT2TX COMPLETE	CODEC READY	WINT	GPIO INT	GPIOTX COMPLETE	SLOT2R XVALID	SLOT1TX COMPLETE

22

Address:

0x8088_0090 - Read Only

Definition:

Global Interrupt Status. The AC97GIS register is the global interrupt status register. All bits are cleared to zero on reset. Each bit is the logical AND of the corresponding bits in the AC97RGIS register and the AC97IM register.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

SLOT2TXCOMPLETE: If this bit is set to "1", the SLOT2TXCOMPLETE interrupt is asserted.

CODECREADY: This bit is set to "1" during a wakeup when the codec indicates that it is ready by setting bit 15 of Slot0

WINT: Wake-up Interrupt Status: If this bit is set to "1", the Wake-up Interrupt is asserted.

GPIOINT: GPIO Interrupt Status: If this bit is set to "1", the GPIOINT interrupt is asserted.

GPIOTXCOMPLETE: If this bit is set to "1", the GPIOTXCOMPLETE interrupt is asserted.

SLOT2RXVALID: If this bit is set to "1", SLOT2RXVALID interrupt is asserted.

SLOT1TXCOMPLETE: If this bit is set to "1", SLOT1TXCOMPLETE interrupt is asserted.

Copyright 2007 Cirrus Logic

AC97IM

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									SLOT2TX COMPLETE	CODEC READY	WINT	GPIO INT	GPIOTX COMPLETE	SLOT2RX VALID	SLOT1TX COMPLETE

Address:

0x8088_0094 - Read/Write

Definition:

Controller Interrupt Enable Register. The AC'97 Controller interrupt enable register is a read/write register that controls the interrupt enables for the interrupts outside the FIFO channels.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

SLOT2TXCOMPLETE: If this bit is set to "1", the SLOT2TXCOMPLETE interrupt is enabled.

CODECREADY: If this bit is set to "1", the Codec Ready Interrupt is enabled.

WINT: If this bit is set to "1", the Wake-up Interrupt is enabled.

GPIOINT: If this bit is set to "1", the GPIO interrupt is enabled.

GPIOTXCOMPLETE: If this bit is set to "1", the GPIOTXCOMPLETE interrupt is enabled.

SLOT2RXVALID: If this bit is set to "1", SLOT2RXVALID interrupt is enabled.

SLOT1TXCOMPLETE: If this bit is set to "1", SLOT1TXCOMPLETE interrupt is enabled.

AC97EOI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD														CODECREADY	WINT

Address:

0x8088_0098 - Write Only

Copyright 2007 Cirrus Logic

Definition:

End Of Interrupt Register. The AC'97 End Of Interrupt Register is a write-only register that allows the CODECREADY and WIS interrupts to be cleared. A write to this location clears the interrupt.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

CODECREADY: CODECREADY Interrupt Status Clear. A write of "1" to this location will clear the CODECREADY interrupt bit.

WINT: Wake-up Interrupt Status Clear. A write of "1" to this location will clear the WIS interrupt bit.

AC97GCR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD													OCODECReady	LOOP	AC97IFE

Address:

0x8088_009C - Read/Write

Definition:

Global Control Register. The AC97GCR register is the main control register for the AC'97 Controller. All bits are cleared on reset.

The AC97IFE creates the clock enable signal for the clock controller block. It is used to enable/disable both PCLK and AC97LK.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

OCODECReady: If set to "1", this bit will override normal CODEC-ready definition.

LOOP: Loopback mode: If this is set to "1", loopback test mode is enabled. Defaults to "0" when reset. Ensure this bit is always "0" for normal operation.

AC97IFE: AC97IF Enable: If this bit is set the AC'97 is enabled. Defaults to "0" on reset. When set to "0", all FIFOs are reset to "0".

Copyright 2007 Cirrus Logic

AC97Reset

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD													EFORCER	FORCED RESET	TIMED RESET

Address:

0x8088_00A0 - Read/Write

Definition:

Controller Reset Register. The AC'97 Controller RESET register is a read/write register that controls various functions within the AC'97 Controller of the RESET port. All the register bits are cleared to "0" when reset.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

EFORCER: Enable for the Forced RESET bit.
1 - FORCEDRESET become active
0 - FORCEDRESET has no effect.

FORCEDRESET: If the EFORCER bit is set to "1", the RESET port will follow whatever value is written to this bit. If this mechanism is used to control the RESET port, it is up to software to ensure that the signal is high long enough to meet the specification of the external device.

This bit has priority over the TIMEDRESET bit.

TIMEDRESET: If this bit is set to "1", the RESET port is forced to "0" for five pulses of the 2.9491 MHz clock
(0.339 µs x 5 = 1.695 µs maximum reset pulse and 1.356 µs minimum reset pulse using this 2.9491 MHz clock). After which this bit is zeroed.

Copyright 2007 Cirrus Logic

AC97SYNC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD													EFORCES	FORCED SYNC	TIMED SYNC

Address:

0x8088_00A4 - Read/Write

Definition:

Sync Control Register. The AC'97 Sync Controller register is a read / write register that controls various functions within the AC'97 Controller of the SYNC port. All the register bits are cleared to "0" when reset.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

EFORCES: Enable for Forced SYNC bit
1 - FORCEDSYNC become active
0 - FORCEDSYNC has no effect.

FORCEDSYNC: If EFORCES bit is set to "1", the SYNC port will follow whatever value is written to this bit. If this mechanism is used to control the SYNC port it is up to software to ensure that the signal is high long enough to meet the specification of the external device.

This bit has priority over the TIMEDSYNC bit.

TIMEDSYNC: If this bit is set to "1", the SYNC port is forced to "1" for five pulses of the 2.9491 MHz (0.339 µs x 5 = 1.695 µs maximum SYNC pulse and 1.356 µs minimum SYNC pulse using this clock). After which this bit is zeroed, allowing the SYNC to be controlled via the BITCLK counter.

Copyright 2007 Cirrus Logic

AC97GCIS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD								AC97GIS							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AC97ISR4				AC97ISR3				AC97ISR2				AC97ISR1			

Address:

0x8088_00A8 - Read Only

Definition:

Global Channel Interrupt Status. The AC97GCIS register (AC'97 Global Channel Interrupt Status) is read only, and echoes all the interrupt status registers in the controller. This register allows the software to read all the interrupt sources with one read.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

AC97GIS: Copy of the AC97GIS register

AC97ISR4: Copy of the AC97ISR 4 register

AC97ISR3: Copy of the AC97ISR 3 register

AC97ISR2: Copy of the AC97ISR 2 register

AC97ISR1: Copy of the AC97ISR 1 register

Copyright 2007 Cirrus Logic

Chapter 23

Synchronous Serial Port

23.1 Introduction

The Synchronous Serial Port (SSP) is a master or slave interface for synchronous serial communication with slave peripheral devices that have either Motorola® SPI, National Semiconductor® Microwire™, or Texas Instruments® synchronous serial interfaces.

The SSP performs serial-to-parallel conversion on data received from a peripheral device. The CPU or DMA reads and writes data and control and status information. The transmit and receive paths are buffered with internal FIFO memories allowing up to eight 16-bit values to be stored independently in both transmit and receive modes. Serial data is transmitted on **SSPTXD** and received on **SSPRXD**.

23.2 Features

Following is a list of features of the Synchronous Serial Port.

- Master or Slave operation
- Programmable clock bit rate and prescaler
- Separate transmit and receive FIFO memory buffers, 16-bits wide, 8 locations deep
- Programmable data frame size from 4 to 16 bits
- Independent masking of transmit FIFO, receive FIFO and receive overrun interrupts

The SSP has a programmable choice of interfaces: SPI, Microwire, or TI synchronous serial. The features of each of these are listed below.

- SPI features:
  - Full duplex, four-wire synchronous transfers
  - Programmable clock polarity and phase
- The feature of the National Semiconductor Microwire interface is:
  - Half duplex transfer using 8-bit control message
- Texas Instrument synchronous serial interface features:
  - Full duplex four-wire synchronous transfer
  - Transmit data pin can be in high impedance state when not transmitting

Copyright 2007 Cirrus Logic

23.3 SSP Functionality

The SSP includes a programmable bit rate clock divider and prescaler to generate the serial output clock SCLKOUT from the input clock SSPCLK. Bit rates are supported to 2MHz and beyond, subject to choice of frequency for SSPCLK. The maximum bit rate will usually be determined by peripheral devices.

The SSP operating mode, frame format and size are programmed though the control registers SSPCR0, SSPCR1.

Three individually maskable interrupt outputs, **SSPTXINTR**, **SSPRXINTR** and **SSPRORINTR** are generated:

- **SSPTXINTR** requests servicing of the transmit buffer
- **SSPRXINTR** requests servicing of the receive buffer
- **SSPRORINTR** indicates an overrun condition in the receive FIFO.

23.4 SSP Pin Multiplex

The SSP pins are multiplexed and may be used for the I^2S controller instead of SSP by setting DeviceCfg.I2SonSSP.

23.5 Configuring the SSP

Following reset, the SSP logic is disabled and must be configured when in this state. Control registers SSPCR0 and SSPCR1 need to be programmed to configure the peripheral as a master or slave operating under one of the following protocols:

- Motorola SPI
- Texas Instruments SSI
- National Semiconductor.

The bit rate, derived from the external SSPCLK, requires the programming of the clock prescale register SSPCPSR. The following procedure must be used to initialize the SSP function:

1. Set the enable bit (SSE) in register SSPCR1.
2. Write the other SSP configuration registers: SSPCR0 and SSPCPSR.
3. Clear the enable bit (SSE) in register SSPCR1.
4. Set the enable bit (SSE) in register SSPCR1.

23.5.1 Enabling SSP Operation

You can either prime the transmit FIFO, by writing up to eight 16-bit values when the SSP is disabled, or allow the transmit FIFO service request to interrupt the CPU. Once enabled, transmission or reception of data begins on the transmit (**SSPTXD**) and receive (**SSPRXD**) pins.

Copyright 2007 Cirrus Logic

23.5.2 Master/Slave Mode

To configure the SSP as a master, clear the SSPCR1 register master or slave selection bit (MS) to 0, which is the default value on reset. Setting the SSPCR1 register MS bit to 1 configures the SSP as a slave. When configured as a slave, enabling or disabling of the SSP **SSPTXD** signal is provided through the SSPCR1 slave mode **SSPTXD** output disable bit (SOD).

23.5.3 Serial Bit Rate Generation

The serial bit rate is derived by dividing down the 7.4 MHz SSPCLK. The clock is first divided by an even prescale value CPSDVSR from 2 to 254, which is programmed in SSPCPSR. The clock is further divided by a value from 1 to 256, which is 1 + SCR, where SCR is the value programmed in SSPCR0. The frequency of the output signal bit clock, **SCLKOUT,** is defined below:

$$F_{sspclkout} = F_{sspclk} / (cpsdvr \times (1 + scr))$$

23.5.4 Frame Format

Each data frame is between 4 and 16 bits long depending on the size of data programmed, and is transmitted starting with the MSB. There are three basic frame types that can be selected:

- Texas Instruments synchronous serial
- Motorola SPI
- National Semiconductor Microwire.

For all three formats, the serial clock (**SCLKOUT**) is held inactive while the SSP is idle, and transitions at the programmed frequency only during active transmission or reception of data. The idle state of **SCLKOUT** is utilized to provide a receive timeout indication that occurs when the receive FIFO still contains data after a timeout period.

For Motorola SPI and National Semiconductor Microwire frame formats, the serial frame (**SFRMOUT**) pin is active LOW, and is asserted (pulled down) during the entire transmission of the frame.

For Texas Instruments synchronous serial frame format, the **SFRMOUT** pin is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For this frame format, both the SSP and the off-chip slave device drive their output data on the rising edge of **SCLKOUT**, and latch data from the other device on the falling edge.

Unlike the full-duplex transmission of the other two frame formats, the National Semiconductor Microwire format uses a special master-slave messaging technique, which operates at half-duplex. In this mode, when a frame begins, an 8-bit control message is transmitted to the off-chip slave. During this transmit, no incoming data is received by the SSP. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the requested data. The returned data can be 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

Copyright 2007 Cirrus Logic

23.5.5 Texas Instruments® Synchronous Serial Frame Format

Figure 23-1 shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

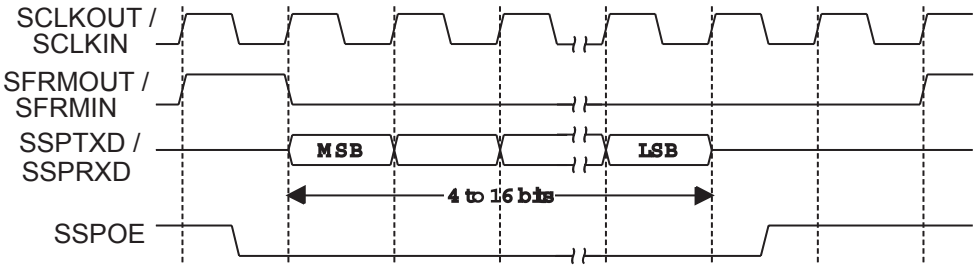


Figure 23-1. Texas Instruments Synchronous Serial Frame Format (Single Transfer)

In this mode, **SCLKOUT** and **SFRMOUT** are forced LOW, and the transmit data line **SSPTXD** is put in the high impedance state whenever the SSP is idle. Once the bottom entry of the transmit FIFO contains data, **SFRMOUT** is pulsed HIGH for one **SCLKOUT** period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of **SCLKOUT**, the MSB of the 4 to 16-bit data frame is shifted out on the **SSPTXD** pin. Likewise, the MSB of the received data is shifted onto the **SSPRXD** pin by the off-chip serial slave device.

Both the SSP and the off-chip serial slave device then clock each data bit into their serial shifter on the falling edge of each **SCLKOUT**. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of **SCLKOUT** after the LSB has been latched. Figure 23-2 shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted.

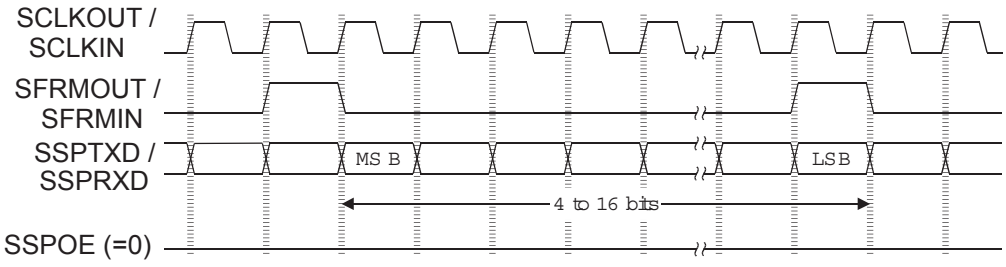


Figure 23-2. TI Synchronous Serial Frame Format (Continuous Transfer)

Copyright 2007 Cirrus Logic

23.5.6 Motorola® SPI Frame Format

The Motorola SPI interface is a four-wire interface where the **SFRMOUT** signal behaves as a slave select. The main feature of the Motorola SPI format is that the inactive state and phase of the **SCLKOUT** signal are programmable through the SPO and SPH bits within the control register, "SSPCR0" on page 23-13.

23.5.6.1 SPO Clock Polarity

When the SPO clock polarity control bit is LOW, it produces a steady state low value on the **SCLKOUT** pin. If the SPO clock polarity control bit is HIGH, a steady state high value is placed on the **SCLKOUT** pin when data is not being transferred.

23.5.6.2 SPH Clock Phase

The SPH control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge.

When the SPH phase control bit is LOW, data is captured on the first clock edge transition. If the SPH clock phase control bit is HIGH, data is captured on the second clock edge transition.

23.5.7 Motorola SPI Format with SPO=0, SPH=0

Single and continuous transmission signal sequences for Motorola SPI format with SPO=0, SPH=0 are shown in Figure 23-3 and Figure 23-4 on page 23-6.

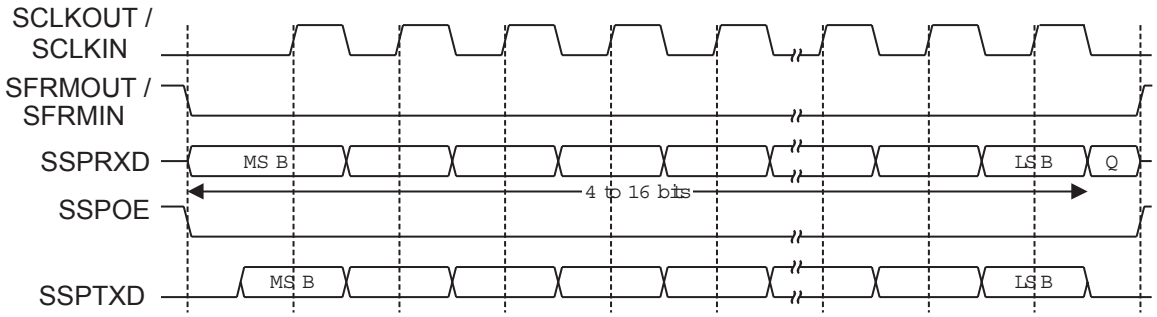


Figure 23-3. Motorola SPI Frame Format (Single Transfer) with SPO=0 and SPH=0

Copyright 2007 Cirrus Logic

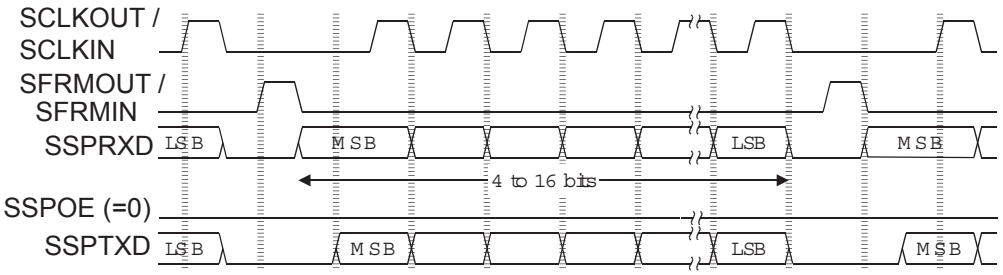


Figure 23-4. Motorola SPI Frame Format (Continuous Transfer) with SPO=0 and SPH=0

In this configuration, during idle periods:

- the **SCLKOUT** signal is forced LOW
- **SFRMOUT** is forced HIGH
- the transmit data line **SSPTXD** is arbitrarily forced LOW
- when the SSP is configured as a master, the **SSPCTLOE** line is driven LOW, enabling the **SCLKOUT** pad (active LOW enable)
- when the SSP is configured as a slave, the **SSPCTLOE** line is driven HIGH, disabling the **SCLKOUT** pad (active LOW enable).

If the SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the **SFRMOUT** master signal being driven LOW. This causes slave data to be enabled onto the **SSPRXD** input line of the master. The master **SSPTXD** output pad is enabled.

One half **SCLKOUT** period later, valid master data is transferred to the **SSPTXD** pin. Now that both the master and slave data have been set, the **SCLKOUT** master clock pin goes HIGH after one further half **SCLKOUT** period.

The data is now captured on the rising edges, and is propagated on the falling edges, of the **SCLKOUT** signal.

In the case of a single word transmission, after all bits of the data word have been transferred, the **SFRMOUT** line is returned to its idle HIGH state one **SCLKOUT** period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the **SFRMOUT** signal must be pulsed HIGH between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is logic zero. Therefore the master device must raise the **SFRMIN** pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the **SFRMOUT** pin is returned to its idle state one **SCLKOUT** period after the last bit has been captured.

Copyright 2007 Cirrus Logic

23.5.8 Motorola SPI Format with SPO=0, SPH=1

The transfer signal sequence for Motorola SPI format with SPO=0, SPH=1 is shown in Figure 23-5, which covers both single and continuous transfers.

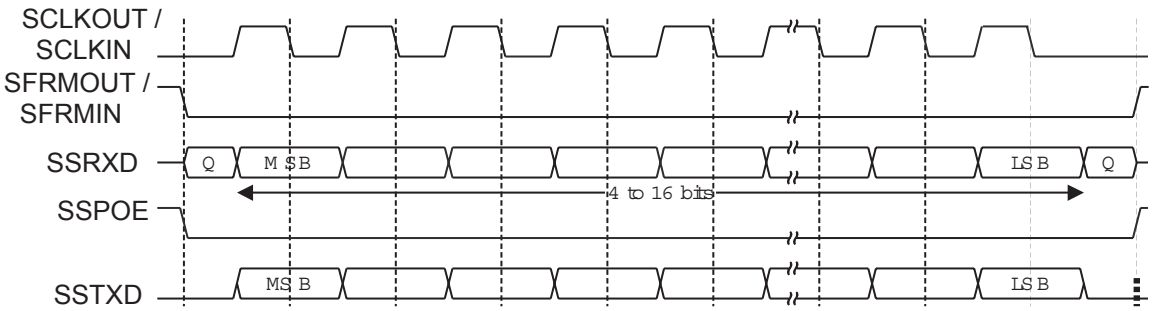


Figure 23-5. Motorola SPI Frame Format with SPO=0 and SPH=1

In this configuration, during idle periods:

- the **SCLKOUT** signal is forced LOW
- **SFRMOUT** is forced HIGH
- the transmit data line **SSPTXD** is arbitrarily forced LOW
- when the SSP is configured as a master, the **SSPCTLOE** line is driven LOW, enabling the **SCLKOUT** pad (active LOW enable)
- when the SSP is configured as a slave, the **SSPCTLOE** line is driven HIGH, disabling the **SCLKOUT** pad (active LOW enable).

If the SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the **SFRMOUT** master signal being driven LOW. The master **SSPTXD** output pad is enabled. After a further one half **SCLKOUT** period, both master and slave valid data is enabled onto their respective transmission lines. At the same time, the **SCLKOUT** is enabled with a rising edge transition.

Data is then captured on the falling edges and propagated on the rising edges of the **SCLKOUT** signal.

In the case of a single word transfer, after all bits have been transferred, the **SFRMOUT** line is returned to its idle HIGH state one **SCLKOUT** period after the last bit has been captured.

For continuous back-to-back transfers, the **SFRMOUT** pin is held LOW between successive data words and termination is the same as that of the single word transfer.

Copyright 2007 Cirrus Logic

23.5.9 Motorola SPI Format with SPO=1, SPH=0

Single and continuous transmission signal sequences for Motorola SPI format with SPO=1, SPH=0 are shown in Figure 23-6 and Figure 23-7.

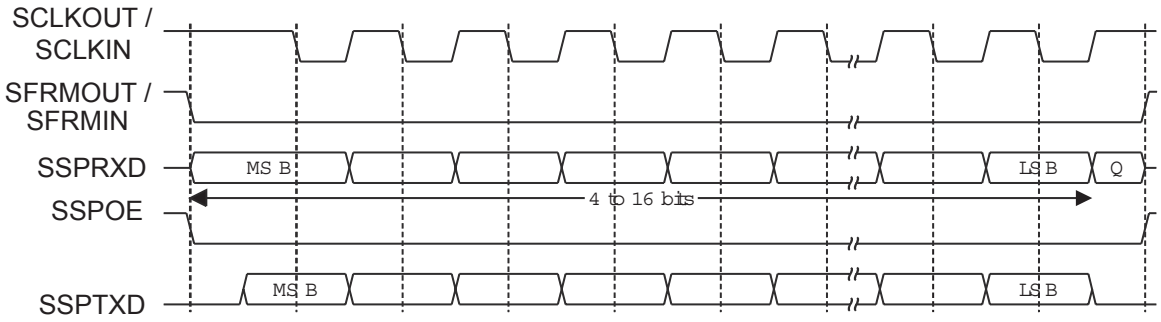


Figure 23-6. Motorola SPI Frame Format (Single Transfer) with SPO=1 and SPH=0

Note: In Figure 23-6, Q is an undefined signal.

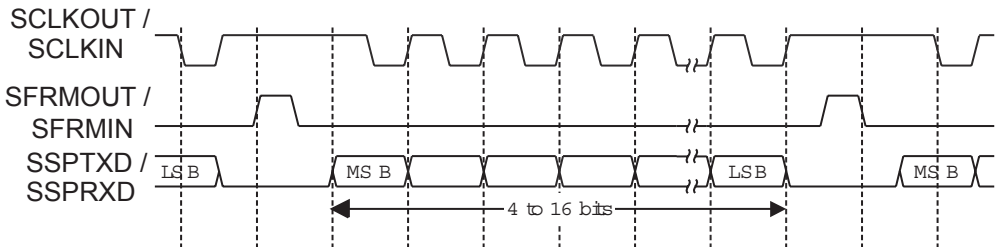


Figure 23-7. Motorola SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0

In this configuration, during idle periods

- the **SCLKOUT** signal is forced HIGH
- **SFRMOUT** is forced HIGH
- the transmit data line **SSPTXD** is arbitrarily forced LOW
- when the SSP is configured as a master, the **SSPCTLOE** line is driven LOW, enabling the **SCLKOUT** pad (active LOW enable)
- when the SSP is configured as a slave, the **SSPCTLOE** line is driven HIGH, disabling the **SCLKOUT** pad (active LOW enable).

Copyright 2007 Cirrus Logic

If the SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the **SFRMOUT** master signal being driven LOW, which causes slave data to be immediately transferred onto the **SSPRXD** line of the master. The master **SSPTXD** output pad is enabled.

One half period later, valid master data is transferred to the **SSPTXD** line. Now that both the master and slave data have been set, the **SCLKOUT** master clock pin becomes LOW after one further half **SCLKOUT** period. This means that data is captured on the falling edges and is propagated on the rising edges of the **SCLKOUT** signal.

In the case of a single word transmission, after all bits of the data word are transferred, the **SFRMOUT** line is returned to its idle HIGH state one **SCLKOUT** period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the **SFRMOUT** signal must be pulsed HIGH between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is logic zero. Therefore the master device must raise the **SFRMIN** pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the **SFRMOUT** pin is returned to its idle state one **SCLKOUT** period after the last bit has been captured.

23.5.10 Motorola SPI Format with SPO=1, SPH=1

The transfer signal sequence for Motorola SPI format with SPO=1, SPH=1 is shown in Figure 23-8, which covers both single and continuous transfers.

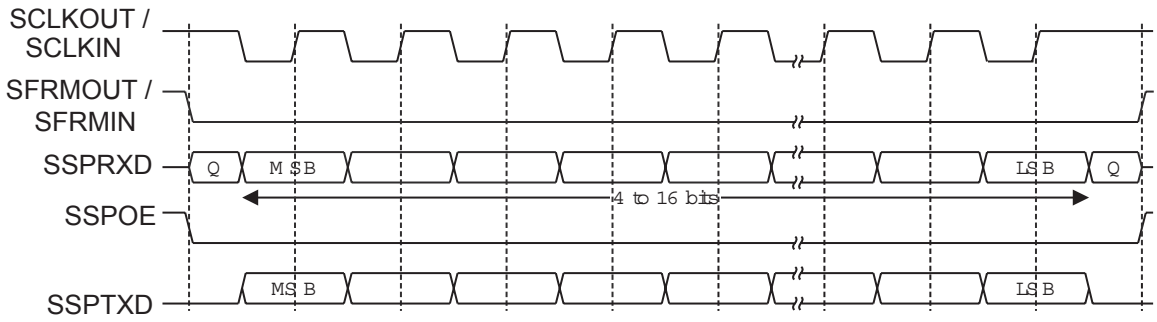


Figure 23-8. Motorola SPI Frame Format with SPO=1 and SPH=1

Note: Figure 23-8, Q is an undefined signal.

In this configuration, during idle periods:

- the **SCLKOUT** signal is forced HIGH
- **SFRMOUT** is forced HIGH
- the transmit data line **SSPTXD** is arbitrarily forced LOW

Copyright 2007 Cirrus Logic

- when the SSP is configured as a master, the **SSPCTLOE** line is driven LOW, enabling the **SCLKOUT** pad (active LOW enable)
- when the SSP is configured as a slave, the **SSPCTLOE** line is driven HIGH, disabling the **SCLKOUT** pad (active LOW enable).

If the SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the **SFRMOUT** master signal being driven LOW. The master **SSPTXD** output pad is enabled. After a further one half **SCLKOUT** period, both master and slave data are enabled onto their respective transmission lines. At the same time, the **SCLKOUT** is enabled with a falling edge transition. Data is then captured on the rising edges and propagated on the falling edges of the **SCLKOUT** signal.

After all bits have been transferred, in the case of a single word transmission, the **SFRMOUT** line is returned to its idle HIGH state one SCLKOUT period after the last bit has been captured.

For continuous back-to-back transmissions, the **SFRMOUT** pins remains in its active LOW state, until the final bit of the last word has been captured, and then returns to its idle state as described above.

For continuous back-to-back transfers, the **SFRMOUT** pin is held LOW between successive data words and termination is the same as that of the single word transfer.

23.5.11 National Semiconductor® Microwire™ Frame Format

Figure 23-9 shows the National Semiconductor Microwire frame format, again for a single frame. Figure 23-10 on page 23-12 shows the same format when back to back frames are transmitted.

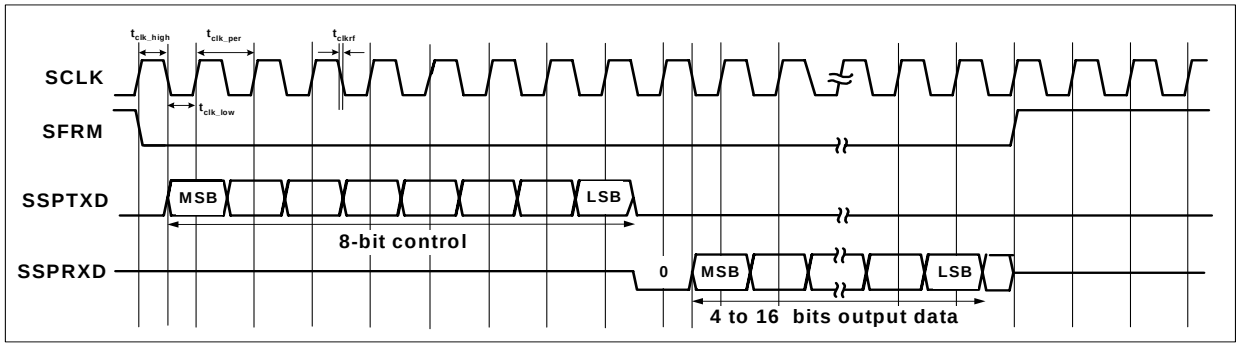


Figure 23-9. Microwire Frame Format (Single Transfer)

Copyright 2007 Cirrus Logic

Microwire format is very similar to SPI format, except that transmission is half-duplex instead of full-duplex, using a master-slave message passing technique. Each serial transmission begins with an 8-bit control word that is transmitted from the SSP to the off-chip slave device. During this transmission, no incoming data is received by the SSP. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the required data. The returned data is 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

In this configuration, during idle periods:

- the **SCLKOUT** signal is forced LOW
- **SFRMOUT** is forced HIGH
- the transmit data line **SSPTXD** is arbitrarily forced LOW

A transmission is triggered by writing a control byte to the transmit FIFO. The falling edge of **SFRMOUT** causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic, and the MSB of the 8-bit control frame to be shifted out onto the **SSPTXD** pin. **SFRMOUT** remains LOW for the duration of the frame transmission. The **SSPRXD** pin remains in a high impedance state during this transmission.

The off-chip serial slave device latches each control bit into its serial shifter on the rising edge of each **SCLKOUT**. After the last bit is latched by the slave device, the control byte is decoded during a one clock wait-state, and the slave responds by transmitting data back to the SSP. Each bit is driven onto **SSPRXD** line on the falling edge of **SCLKOUT**. The SSP in turn latches each bit on the rising edge of **SCLKOUT.** At the end of the frame, for single transfers, the **SFRMOUT** signal is pulled HIGH one clock period after the last bit has been latched in the receive serial shifter, that causes the data to be transferred to the receive FIFO.

Note: The off-chip slave device can put the receive line in a high impedance state either on the falling edge of **SCLKOUT** after the LSB has been latched by the receive shifter, or when the **SFRMOUT** pin goes HIGH.

For continuous transfers, data transmission begins and ends in the same manner as a single transfer. However, the **SFRMOUT** line is continuously asserted (held LOW) and transmission of data occurs back to back. The control byte of the next frame follows directly after the LSB of the received data from the current frame. Each of the received values is transferred from the receive shifter on the falling edge **SCLKOUT**, after the LSB of the frame has been latched into the SSP.

Copyright 2007 Cirrus Logic

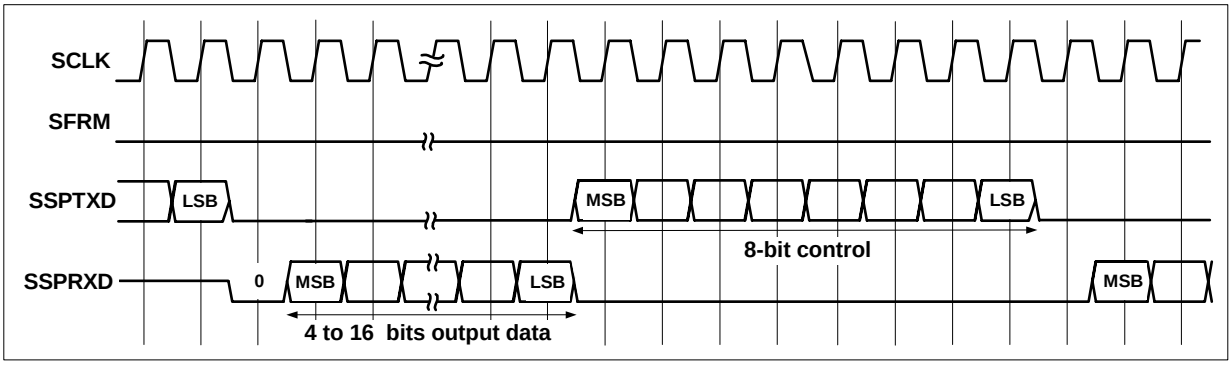


Figure 23-10. Microwire Frame Format (Continuous Transfers)

23.5.11.1 Setup and Hold Time Requirements on SFRMIN with Respect to SCLKIN in Microwire Mode

In the Microwire mode, the SSP slave samples the first bit of receive data on the rising edge of **SCLKIN** after **SFRMIN** has gone LOW. Masters that drive a free-running **SCLKIN** must ensure that the **SFRMIN** signal has sufficient setup and hold margins with respect to the rising edge of **SCLKIN**.

Figure 23-11 illustrates these setup and hold time requirements. With respect to the **SCLKIN** rising edge on which the first bit of receive data is to be sampled by the SSP slave, **SFRMIN** must have a setup of at least two times the period of **SCLKIN** on which the SSP operates. With respect to the **SCLKIN** rising edge previous to this edge, **SFRMIN** must have a hold of at least one **SCLKIN** period.

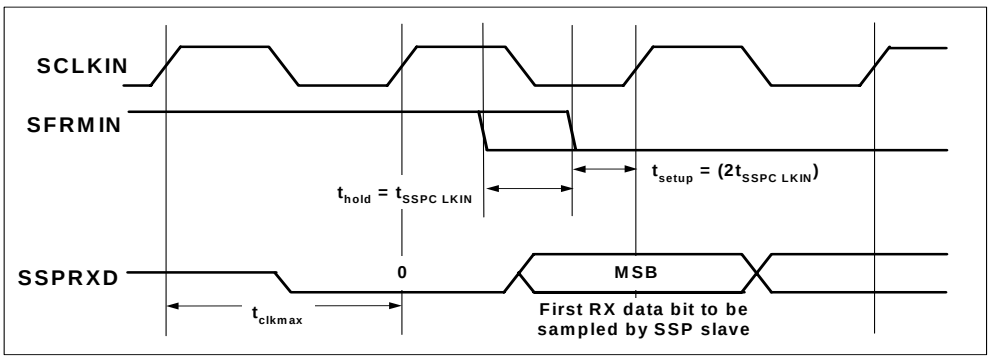


Figure 23-11. Microwire Frame Format, SFRMIN Input Setup and Hold Requirements

Copyright 2007 Cirrus Logic

23.6 Registers

The SSP registers are shown in the following table.

Table 23-1. SSP Register Memory Map Description

Address	Type	Width	Reset value	Name	Description
0x808A_0000	Read/write	16	0x0000	SSPCR0	Control register 0.
0x808A_0004	Read/write	8	0x00	SSPCR1	Control register 1.
0x808A_0008	Read/write	16	0x0000	SSPDR	Receive FIFO (Read)/ Transmit FIFO data register (Write).
0x808A_000C	Read	7	0x00	SSPSR	Status register.
0x808A_0010	Read/write	8	0x00	SSPCPSR	Clock prescale register.
0x808A_0014	Read	3	0x0	SSPIIR/ SSPICR	Interrupt identification register (read) Interrupt clear register (write).
0x808A_0018 - 0x808A_003C	-	-	-	-	Reserved
0x808A_0094 - 0x808A_00FF	-	-	-	-	Reserved

Register Descriptions

SSPCR0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCR								SPH	SPO	FRF		DSS			

Address:

0x808A_0000 - Read/Write

Default:

0x0000_0000

Definition:

SSPCR0 is the control register 0 and contains four different bit fields, which control various functions within the SSP.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

Copyright 2007 Cirrus Logic

SCR: Serial clock rate. The value SCR is used to generate the transmit and receive bit rate of the SSP. SCR is a value from 0 to 255. This provides the secondary divide of (1+SCR) after a pre divide of CPSDVSR (ranging from 2 to 254)

SPH: SCLKOUT phase (applicable to Motorola SPI frame format only).

SPO: SCLKOUT polarity (applicable to Motorola SPI frame format only).

FRF: Frame format:
00 Motorola SPI frame format
01 - TI synchronous serial frame format
10 - National Semiconductor Microwire frame format
11 - Reserved, undefined operation

DSS: Data Size Select:
0000 - Reserved, undefined operation
0001 - Reserved, undefined operation
0010 - Reserved, undefined operation
0011 - 4-bit data
0100 - 5-bit data
0101 - 6-bit data
0110 - 7-bit data
0111 - 8-bit data
1000 - 9-bit data
1001 - 10-bit data
1010 - 11-bit data
1011 - 12-bit data
1100 - 13-bit data
1101 - 14-bit data
1110 - 15-bit data
1111 - 16-bit data

SSPCR1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									SOD	MS	SSE	LBM	RORIE	TIE	RIE

Address:

0x808A_0004 - Read/Write

Copyright 2007 Cirrus Logic

Default:

0x0000_0000

Definition:

SSPCR1 is the control register 1 and contains six different bit fields, which control various functions within the SSP.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

SOD: Slave-mode output disable. This bit is relevant only in the slave mode (MS=1). In multiple-slave systems, it is possible for an SSPMS master to broadcast a message to all slaves in the system while ensuring that only one slave drives data onto its serial output line. In such systems the RXD lines from multiple slaves can be tied together. To operate in such systems, the SOD may be set if the SSP slave is not supposed to drive the **SSPTXD** line.
0 - SSP may drive the **SSPTXD** output in slave mode.
1 - SSP must not drive the **SSPTXD** output in slave modes.

MS: Master / Slave mode select. This bit can be modified only when the SSP is disabled (SSE=0).
0 - Device configured as master (default).
1 - Device configured as slave.

SSE: Synchronous serial port enable:
0 - SSP operation disabled
1 - SSP operation enabled.

LBM: Loop back mode:
0 - Normal serial port operation enabled.
1 - Output of transmit serial shifter is connected to input of receive serial shifter internally.

RORIE: Receive FIFO overrun interrupt enable:
0 - Overrun detection is disabled. Overrun condition does not generate the **SSPRORINTR** interrupt.
1 - Overrun detection is enabled. Overrun condition generates the **SSPRORINTR** interrupt.

TIE: Transmit FIFO interrupt enable:
0 - Transmit FIFO half-full or less condition does not generate the **SSPTXINTR** interrupt.
1 - Transmit FIFO half-full or less condition generates the **SSPTXINTR** interrupt.

Copyright 2007 Cirrus Logic

RIE: Receive FIFO interrupt enable:
0 - Receive FIFO half-full or more condition does not generate the **SSPRXINTR** interrupt.
1 - Receive FIFO half-full or more condition generates the **SSPRXINTR** interrupt.

SSPDR

23

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															

Address:

0x808A_0008 - Read/Write

Default:

0x0000_0000

Definition:

SSPDR is the data register and is 16-bits wide. When SSPDR is read, the entry in the receive FIFO (pointed to by the current FIFO read pointer) is accessed. As data values are removed by the SSPs receive logic from the incoming data frame, they are placed into the entry in the receive FIFO (pointed to by the current FIFO write pointer).

When SSPDR is written, the entry in the transmit FIFO (pointed to by the write pointer), is written. Data values are removed from the transmit FIFO one value at a time by the transmit logic. It is loaded into the transmit serial shifter, then serially shifted out onto the **SSPTXD** pin at the programmed bit rate.

When a data size of less than 16 bits is selected, the user must right-justify data written to the transmit FIFO. The transmit logic ignores the unused bits. Received data less than 16 bits is automatically right justified in the receive buffer.

When the SSP is programmed for National Semiconductor Microwire frame format, the default size for transmit data is eight bits (the most significant byte is ignored). The receive data size is controlled by the programmer. The transmit FIFO and the receive FIFO are not cleared even when SSE is set to zero. This allows the software to fill the transmit FIFO before enabling the SSP.

Bit Descriptions:

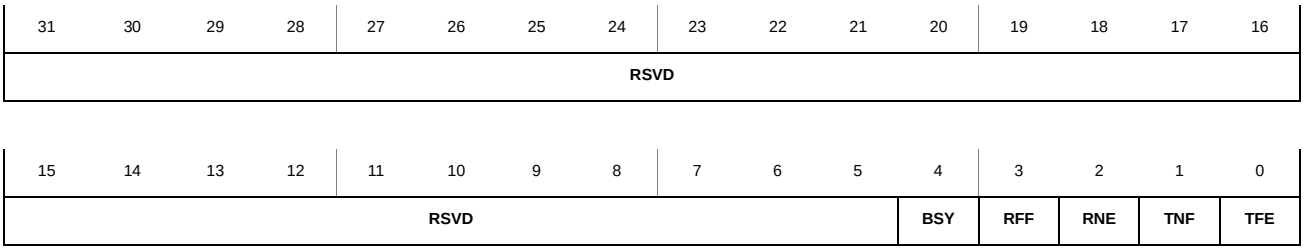
RSVD: Reserved. Unknown During Read.

Copyright 2007 Cirrus Logic

DATA: Transmit / Receive FIFO:
Read - Receive FIFO
Write - Transmit FIFO

Note: The user should right-justify data when the SSP is programmed for a data size that is less than 16 bits. Unused bits at the top are ignored by transmit logic. The receive logic automatically right justifies.

SSPSR



Address:
0x808A_000C - Read Only

Default:
0x0000_0000

Definition:
SSPSR is a read-only status register, which contains bits that indicate the FIFO fill status and the SSP busy status.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

BSY: SSP busy flag (read-only):
0 - SSP is idle.
1 - SSP is currently transmitting and / or receiving a frame or the transmit FIFO is non-empty.

RFF: Receive FIFO full (read-only):
0 - Receive FIFO is not full
1 - Receive FIFO is full

RNE: Receive FIFO not empty (read-only):
0 - Receive FIFO is empty.
1 - Receive FIFO is not empty

TNF: Transmit FIFO not full (read-only):
0 - Transmit FIFO is full.
1 - Transmit FIFO is not full.

TFE: Transmit FIFO empty (read-only):
0 - Transmit FIFO is not empty
1 - Transmit FIFO is empty

Copyright 2007 Cirrus Logic

SSPCPSR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								CPSDVSR							

23

Address:

0x808A_0010 - Read/Write

Default:

0x0000_0000

Definition:

SSPCPSR is the clock prescale register and specifies the division factor by which the input SSPCLK should be internally divided before further use.

The value programmed into this register should be an even number between 2 and 254. The least significant bit of the programmed number is hard-coded to zero. If an odd number is written to this register, data read back from this register will have the least significant bit as zero.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

CPSDVSR: Clock pre-scale divisor. Should be an even number from 2 to 254, depending on the frequency of SSPCLK. The least significant bit CPSDVSR[0] always returns zero on reads since it is hard-coded to 0

SSPIIR / SSPICR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD													RORIS	TIS	RIS

Address:

0x808A_0014 - Read Only

Note: A write to this register clears the receive overrun interrupt, regardless of the data value written.

Default:

0x0000_0000

Copyright 2007 Cirrus Logic

Definition:

The interrupt status is read from the SSP interrupt identification register (SSPIIR). A write of any value to the SSP interrupt clear register (SSPICR) clears the SSP receive FIFO overrun interrupt. Therefore, clearing the RORIE bit in the SSPCR1 register will also clear the overrun condition if already asserted. All the bits are cleared to zero when reset.

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
RORIS:	Read: SSP Receive FIFO overrun interrupt status 0 - **SSPRORINTR** is not asserted. 1 - **SSPRORINTR** is asserted. This bit is cleared by writing any value to the SSPSR register
TIS:	Read: SSP transmit FIFO service request interrupt status 0 - **SSPTXINTR** is not asserted indicating that the transmit FIFO is more than half full. 1 - **SSPTXINTR** is asserted indicating that the transmit FIFO is less than half full (space available for at least four half words).
RIS:	Read: SSP receive FIFO service request interrupt status 0 - **SSPRXINTR** is not asserted indicating that the receive FIFO is less than half full. 1 - **SSPRXINTR** is asserted indicating that the receive FIFO is more than half full (4 or more half words present in FIFO)

Copyright 2007 Cirrus Logic

23

Copyright 2007 Cirrus Logic

Chapter 24

Pulse Width Modulator

24

24.1 Introduction

Note: The EP9307 processor has one PWM with one output, PWMOUT.

Note: The EP9301, EP9302, EP9312, and EP9315 processors each have two PWMs with two outputs, PWMOUT and PWMO1. PWMO1 is an alternate function for EGPIO14.

The Pulse Width Modulators (PWMs) have the following features:

- Configurable dual output
- Separate input clocks for each PWM output
- 16-bit resolution
- Programmable synchronous mode support
  - Allows external input to start PWM
- Programmable pulse width (duty cycle), interval (frequency), and polarity
  - Static programming: PWM is stopped
  - Dynamic programming: PWM is running
  - Updates duty cycle, frequency, and polarity at end of a PWM cycle

24.2 Theory of Operation

Each PWM is an Advanced Microcontroller Bus Architecture (AMBA) compliant system-on-a chip (SOC) peripheral. Each is a configurable dual-output, dual clock input AMBA slave module, and each connects to the Advanced Peripheral Bus (APB). The PWM Interfaces comply with the AMBA Specifications (Rev.2.0). This design assumes little-endian memory organization.

Both of the PWM peripherals are programmed via the APB, receive scaled clock inputs from the clock controller and produce outputs at external pins.

The processor has two independent DC-level PWM outputs, PWMOUT and PWMO1. The PWM outputs are sourced by programable duty-cycle pulse generators. From a programmer's point of view, for each channel there are two 16-bit registers. These can be used to specify the width of the pulse cycle (in terms of number of PWM clock cycles), and the duration of high-phase of the pulse (set in terms of the number of PWM clock cycles).

Copyright 2007 Cirrus Logic

With those two counters specified, a fixed pulse is generated. The two channels are totally independent. This is a DC-level PWM.

Either PWM channel can be utilized to create reoccurring pulses at the **PWMx** output pins. Depending upon how a PWM is programmed, its output can vary from a continuous level (100% duty-cycle), to a square wave (50% duty-cycle), to a narrow pulse approaching a 0% duty-cycle. Both PWMs offer 16-bit resolution of the input clock signal.

The outputs of both PWM channels are programmed in terms of PWM input clock cycles. Each PWM may be programmed statically (when it is halted) or dynamically (while it is running). The output of either PWM may be programmed as normal or inverted. With the exception of inversion, if a PWM is programmed statically, no change to the output will occur until the PWM is enabled. If a PWM is reprogrammed while it is running (enabled), the output is updated to the new programming (total period, total period asserted) at the beginning of the next PWM cycle. The exception for inverted operation is explained below. Both PWMs are reset to the halted condition.

The output of either PWM can be programmed for either normal or inverted operation. Inversion affects the output pin when the PWM peripheral is halted and also when it is running. Both outputs are reset to the normal (non-inverted) configuration, which places the output pins in a LOW condition at reset. When the output is reprogrammed to be inverted (or to be normal), the new programming does not become effective until the rising edge of the PWM input clock signal.

Note: In the design, because of the use of clock gating on PCLK, the write enable and read enable were altered to work correctly within the design.

24.2.1 PWM Programming Examples

The reference clock for PWM is **XTALI**.

24.2.1.1 Example

To produce a PWM output of 100 kHz (10 µsec) and 20% duty cycle with a system clock of 66 MHz (15 nsec):

1. Calculate PWMxTermCnt = (66 MHz / 0.1 MHz) – 1 = 659 (decimal).

2. Calculate PWMxDutyCycle = (0.2 × (659 + 1)) – 1 = 131 (decimal).

24.2.1.2 Static Programming (PWM is Not Running) Example

Table 24-1. Static Programming Steps

Step	Register	Value
Stop PWM	PWMxEn	0x0000
Wait for PWM to finish current cycle	----------	
Program TC value with 659 (decimal)	PWMxTermCnt	0x0293
Program DC value with 131 (decimal)	PWMxDutyCycle	0x0083
Program PWM output to invert	PWMxInvert	0x0001
Enable/Start PWM	PWMxEn	0x0001

Copyright 2007 Cirrus Logic

24.2.1.3 Dynamic Programming (PWM is Running) Example

Note: Updates will take place at the end of the PWM cycle. Order of programming of the Terminal Count and Duty Cycle is important. See Section 24.2.2 on page 24-3.

Table 24-2. Dynamic Programming Steps

Step	Register	Value
Program TC value with 659	PWMxTermCnt	0x0293
Program DC value with 131	PWMxDutyCycle	0x0081

24.2.2 Programming Rules

1. Because the user can not tell the state of the PWM between cycles, care must be taken while programming on the fly (while the PWM is running). To insure proper operation observe the following rules to preserve the relationship of PWMxDutyCycle to PWMxTermCnt:

   A. If PWMxTermCnt (new) > PWMxTermCnt (current):

   Program PWMxTermCnt (new) first then PWMxDutyCycle (new)

   B. If PWMxTermCnt (new) < PWMxTermCnt (current):

   Program PWMxDutyCycle (new) first then PWMxTermCnt (new)

   If the rules "A" and "B" are not followed, a duty cycle of 100% may result until both PWMxTermCnt and PWMxDutyCycle are updated.

2. Program PWMxTermCnt and PWMxDutyCycle with values that meet the specification.

3. When PWM is stopped (PWM_EN = 0), it does not stop immediately but waits for the end of the current PWM cycle and then stops.

24.3 Registers

Table 24-3. PWM Registers Map

Address	Register	Name	Reset Value	Access	Size
0x8091_0000	PWM0TermCnt	PWM0 Terminal Count	0x0000	R/W	16
0x8091_0004	PWM0DutyCycle	PWM0 Duty Cycle	0x0000	R/W	16
0x8091_0008	PWM0En	PWM0 Enable	0x0000	R/W	16
0x8091_000C	PWM0Invert	PWM0 Invert	0x0000	R/W	16
0x8091_0020	PWM1TermCnt	PWM1 Terminal Count	0x0000	R/W	16
0x8091_0024	PWM1DutyCycle	PWM1 Duty Cycle	0x0000	R/W	16
0x8091_0028	PWM1En	PWM1 Enable	0x0000	R/W	16
0x8091_002C	PWM1Invert	PWM1 Invert	0x0000	R/W	16

Note: All pwmout outputs will drive a logical "0" during reset. Coming out of reset, it will continue to drive a logical "0", and the PWM will be in halt mode.

Copyright 2007 Cirrus Logic

Note: All undefined register bits will be read as 0.

Register Descriptions

PWMxTermCnt

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PWM_TC															

Address:

PWM0TermCnt: 0x8091_0000 - Read/Write
PWM1TermCnt: 0x8091_0020 - Read/Write

Default:

0x0000_0000

Definition:

PWMx Terminal Count

Bit Descriptions:

PWM_TC: PWMxTermCnt is used to adjust the output frequency of the PWM. PWMxTermCnt gives the PWM up to 16-bit resolution.

PWMxTermCnt is double buffered to allow it to be programed statically (PWM is stopped) or dynamically (PWM is running).

Programmed dynamically, PWMxTermCnt is updated at the end of a PWM cycle to prevent any output glitches or errors. Reading the register reflects what was written to it, not the state of the counter.

PWMxDutyCycle

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PWM_DC															

Address:

PWM0DutyCycle: 0x8091_0004 - Read/Write
PWM1DutyCycle: 0x8091_0024 - Read/Write

Default:

0x0000_0000

Definition:

PWMx Duty Cycle

Copyright 2007 Cirrus Logic

Bit Descriptions:

PWM_DC: PWM_DC is used in conjunction with PWMxTermCnt to adjust the output duty cycle of PWM. PWMxDutyCycle is double buffered to allow it to be programed statically (PWM is stopped) or dynamically (PWM is running). Programmed dynamically, PWMxDutyCycle is updated at the end of a PWM cycle to prevent any output glitches or errors. Reading the register reflects what was written to it, not the state of the counter.

PWMxEn

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															EN

Address:

PWM0En: 0x8091_0008 - Read/Write
PWM1En: 0x8091_0028 - Read/Write

Default:

0x0000_0000

Definition:

PWMx Enable

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

EN: Enable PWM

0 - Disable/Stop PWM. The PWM is actually stopped when it reaches the end of its current cycle. PWM output is:
0 - If PWM_INV = 0
1 - if PWM_INV = 1

1 - PWM is Enabled. When in normal mode writing a one will start the PWM.
PWMxTermCnt is updated with its new buffered value.
PWMxDutyCycle is updated with its new buffered value.

PWMxInvert

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															INV

Address:

PWM0Invert: 0x8091_000C - Read/Write
PWM1Invert: 0x8091_002C - Read/Write

Copyright 2007 Cirrus Logic

Default:

0x0000_0000

Definition:

PWMx Invert

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

INV: Invert PWM output
0 = Output is not inverted. **PWMOUT** will output t_{ON} first then t_{OFF}, PWMxDutyCycle controls t_{ON}
1 = Output is inverted. **PWMOUT** will output t_{OFF} first then t_{ON}, PWMxDutyCycle controls t_{OFF}.

PWM_INV is double buffered to allow it to be programmed statically (PWM is stopped) or dynamically (PWM is running).

Programmed statically, the invert takes affect after the APB write completes and CLK_PWM is running. After the update, CLK_PWM can be turned off without affecting **PWMOUT**. In this way, the PWM output can be inverted without enabling the PWM.

Programmed dynamically, PWM_INV is updated at the end of a PWM cycle to prevent any output glitches or errors. Read/write accesses to PWM_INV will read/write its buffer.

Figure 24-1 provides an example of the effect of the PWM_INV invert bit.

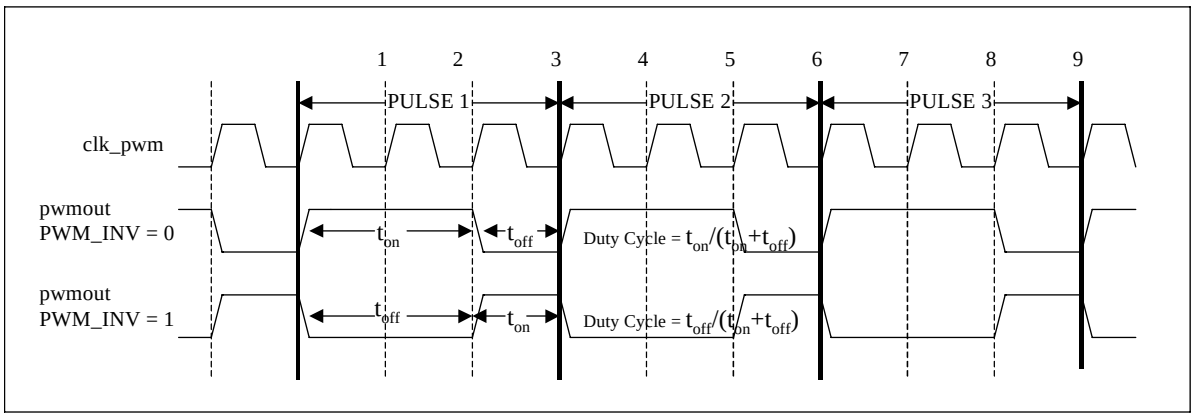


Figure 24-1. PWM_INV Example

Copyright 2007 Cirrus Logic

Chapter 25

Analog Touch Screen Interface

25

25.1 Introduction

Note: The EP9301 and EP9302 processors each support a general 5-bit ADC, but no touch screen .

Note: The EP9307, EP9312, and EP9315 processors each support up to an 8-wire touch screen or a general 12-bit ADC.

The touch screen controller is a hardware engine that controls scanning for 4, 5, 7, or 8-wire analog resistive touch screens (TS). The engine performs all sampling, averaging, and range checking for analog-to-digital converter values. The hardware engine also controls an analog switch array for controlling the different scan modes.

Through the APB interface, it is possible to disable the touch screen hardware engine and directly control the switch matrix and the analog-to-digital converter. This allows for the implementation of more complex software scanning algorithms if desired. However, the hardware engine provides all features necessary to implement a standard interface and eliminates almost all ARM Core intervention in the scanning process.

For both X and Y axes, the touch screen controller engine does the following:

- Takes 4, 8, 16, or 32 sample sets.
- Checks to see if the deviation in the sample set is too great.
- Averages the samples.
- Checks the average to see if the move is too great to be realistic.
- Checks to see if the move is large enough to bother the ARM Core.

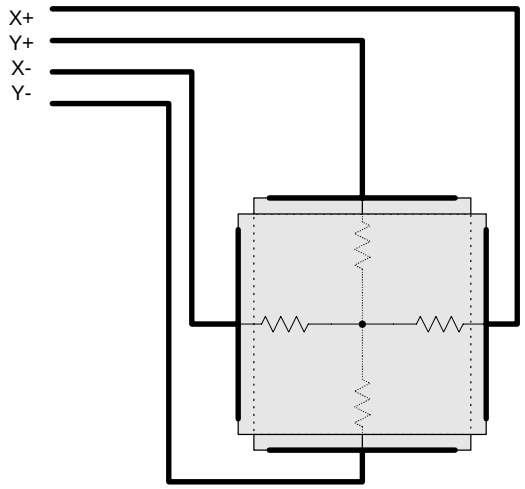
Many touch screen controllers send a continuous information stream to the processor when the touch screen is in use. This controller, in contrast, only interrupts the ARM Core when the stimulus change is significant.

25.2 Touch Screen Controller Operation

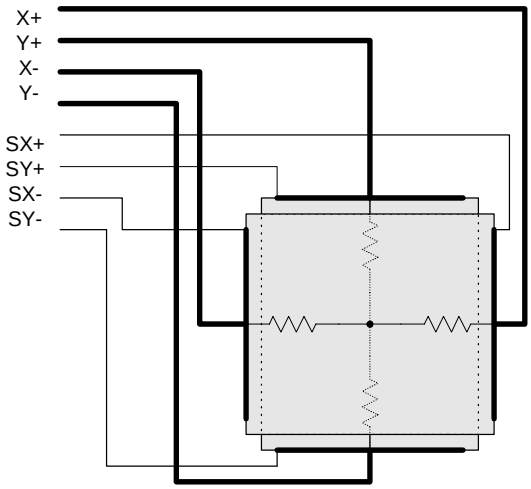
To understand the controller operation, it is first important to understand the electrical implementation of touch screen technologies. The schematics for various resistive touch screen technologies are shown in Figure 25-1. When the front and back indium-tin-oxide layers are pressed together, a resistive contact is made.

Copyright 2007 Cirrus Logic

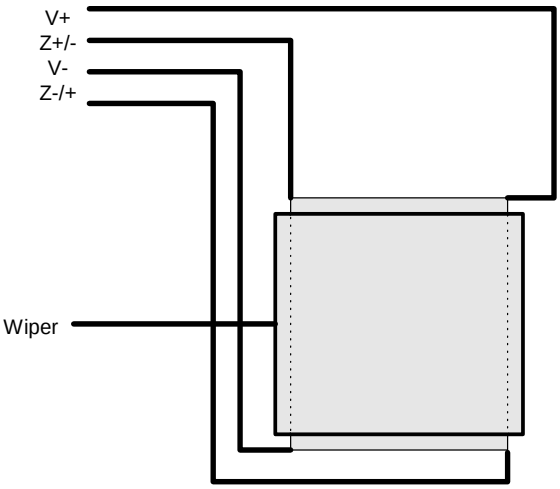
25



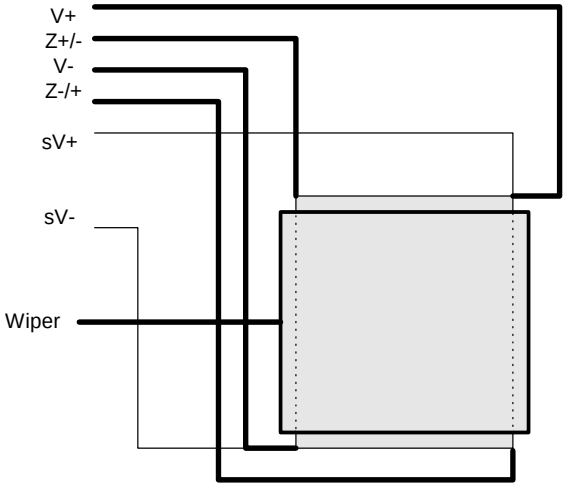
4 WIRE ANALOG RESISTIVE TOUCH SCREEN SCHEMATIC



8 WIRE ANALOG RESISTIVE TOUCH SCREEN SCHEMATIC



5 WIRE ANALOG RESISTIVE TOUCH SCREEN SCHEMATIC



7 WIRE (5 WIRE WITH FEEDBACK) ANALOG RESISTIVE TOUCH SCREEN SCHEMATIC

Figure 25-1. Different Types of Touch Screens

For 8- and 4-wire touch screens, the point at which the contact is made can be determined by measuring a voltage driven between the bus bars on the X-axis layer through either or both of the Y lines, and by then measuring a voltage driven between the bus bars on the Y-axis layer through either or both of the X lines. By comparing these voltages to values determined during calibration, the location of the touch can be determined.

Copyright 2007 Cirrus Logic

For 8-wire touch screens, the SX and SY lines are used as the measurement reference for the analog-to-digital converter to provide better resolution during a reading and compensate for any drift in samples due to other factors. A 4-wire analog resistive touch screen is the same as an 8-wire without the SX and SY feedback lines. A 4-wire analog resistive touch screen may be adequate for non-industrial use or small touch screens.

For 5-wire touch screens and 7-wire (5-wires with feedback) touch screens, a constant voltage is applied from corner to corner of the lower layer. Switching of the X and Y axis is performed by driving appropriate voltages on Z+/- and Z-/+. The location values during both the X and Y scans are then read by sampling the Wiper input. A 7-wire touch screen (5-wires with feedback), provides reference feedback voltages to the analog-to-digital converter to eliminate analog switch and other circuit resistances.

The changes in connection for sampling are performed by a set of separately controlled analog switches. These switches may be connected in a variety of ways which allows a high degree of flexibility in using the analog-to-digital converter. To avoid contention, each switch drive circuit has a much faster turn-off than turn-on to provide an overall break-before-make array function.

Logic safeguards are included to condition the control signals for power connection to the matrix to prevent part damage. In addition, a software lock register is included that must be written with 0xAA before each register write to change the values of the four switch matrix control registers.

Table 25-1 provides switch definitions and the logical safeguards that are implemented to prevent physical part damage. A "1" in the register bit position closes the corresponding switch.

Table 25-1. Switch Definitions and Logical Safeguards to Prevent Physical Damage

Switch CTL	Connection From	Connection To	Only allowed if
Bit 0	X+	ADC input	-
Bit 1	X-	ADC input	-
Bit 2	Y+	ADC input	-
Bit 3	Y-	ADC input	-
Bit 4	sY-	ADC input	-
Bit 5	sY+	ADC input	-
Bit 6	sX-	ADC input	-
Bit 7	sX+	ADC input	-
Bit 8	VBAT	ADC input	SW8 = HIGH and the bus SW[7-0] = 0x00
Bit 9	AVDD	ADC REF+	SW9 = HIGH and SW26 = LOW and SW24 = LOW
Bit 10	AGND	ADC REF-	-
Bit 11	X+	AGND	-
Bit 12	X-	AGND	-
Bit 13	Y+	AGND	-
Bit 14	Y-	AGND	-
Bit 15	sX+	AGND	-
Bit 16	sX-	AGND	-
Bit 17	sY+	AGND	-

Copyright 2007 Cirrus Logic

Table 25-1. Switch Definitions and Logical Safeguards to Prevent Physical Damage (Continued)

Bit 18	sY-	AGND	-
Bit 19	X+	AVDD	SW19 = HIGH and SW11 = LOW and SW0 = LOW
Bit 20	Y+	AVDD	SW20 = HIGH and SW13 = LOW and SW2 = LOW
Bit 21	Y-	AVDD	SW21 = HIGH and SW14 = LOW and SW3 = LOW
Bit 22	X+	Pull-up to AVDD	SW22 = HIGH and SW11 = LOW and (SW0 = LOW or the bus SW[7-0] = 0x01)
Bit 23	Y+	Pull-up to AVDD	SW23 = HIGH and SW13 = LOW and (SW2 = LOW or the bus SW[7-0] = 0x04)
Bit 24	sX+	ADC REF+	-
Bit 25	sX-	ADC REF-	-
Bit 26	sY+	ADC REF+	-
Bit 27	sY-	ADC REF-	-
Bit 28	TOUCH_PRESS gate	Touch_Press NAND gate	Gates input to prevent oscillations
Bit 29	DAC	ADC input	DAC to ADC feedback test switch
Bit 30	VBAT ADC input	Load Resistor	Also gates PRSTn for measuring battery with load

25.2.1 Touch Screen Scanning: Four-wire and Eight-wire Operation

Figure 25-2 shows control for an 8-wire analog resistive touch screen. The register values TSDetect, TSDischarge, TSXSample, and TSYSample are derived from the switch positions in the diagram. These values are listed in Table 25-2.

The left most part of the diagram shows how the switches are driven to detect a touch press. This configuration is controlled by the TSDetect register.

The second part of the diagram shows all lines of the touch screen switched to ground for discharging as controlled by the TSDischarge register.

The third example shows a voltage driven across the X-axis with the A / D sample input connected to both Y-axis lines and the A / D referenced to the SX feedback lines as controlled by the TSXSample register.

The right most part of the diagram shows a voltage driven across the Y-axis with the A / D sample input connected to both X-axis lines and the A / D referenced to the SY feedback lines for digitization as controlled by the TSYSample register.

Each time the analog switching configuration is changed, an appropriate time interval must be provided for the circuit to stabilize due to the touch screen capacitance, and any EMI filtering provided on the signals. This is controlled by the DLY field in the TSSetup register.

The analog touch screen interface circuitry is internally connected to a signed 12-bit analog-to-digital converter. The 12-bit digital result is held stable until another sample is requested. The controller reads the digital value by issuing a read pulse. This read pulse signal from the controller is also used as a convert command for the analog-to-digital converter to begin conversion of its next sample. The output of the ADC is a signed value. For the comparison functions of the touch controller to work correctly, the TSSetup2.SIGND bit should be set.

The flow chart in Figure 25-4 demonstrates the sample process used for determining touch input on a touch screen. The ARM Core must load all of the setup registers for the touch

Copyright 2007 Cirrus Logic

array scanning and enable the state machine. In determining a touch point, the first axis to be scanned is the X-axis. X and Y axis definitions are arbitrary and must only be coordinated with the code when determining a screen position. For 8-wire and 4-wire implementations, the touch screen X and Y axis positioning should be linear for all checking algorithms to work linearly. The algorithm and the returned values will not be linear for a 5-wire touch screens or 7-wire (5-wires with feedback) touch screens and must be adjusted by software to determine screen position. Some newer technology 5-wire touch screens approach linearity and will need to be adjusted differently in software.

The same algorithm for sampling is used by 4, 5, 7, and 8-wire touch screens. However, the switch combinations controlled by the algorithm are determined by configuration registers. The configuration registers are set up by software according to the touch screen type. Table 25-2 details the configuration values required for 4-, 5-, 6-, and 8-wire touch screens.

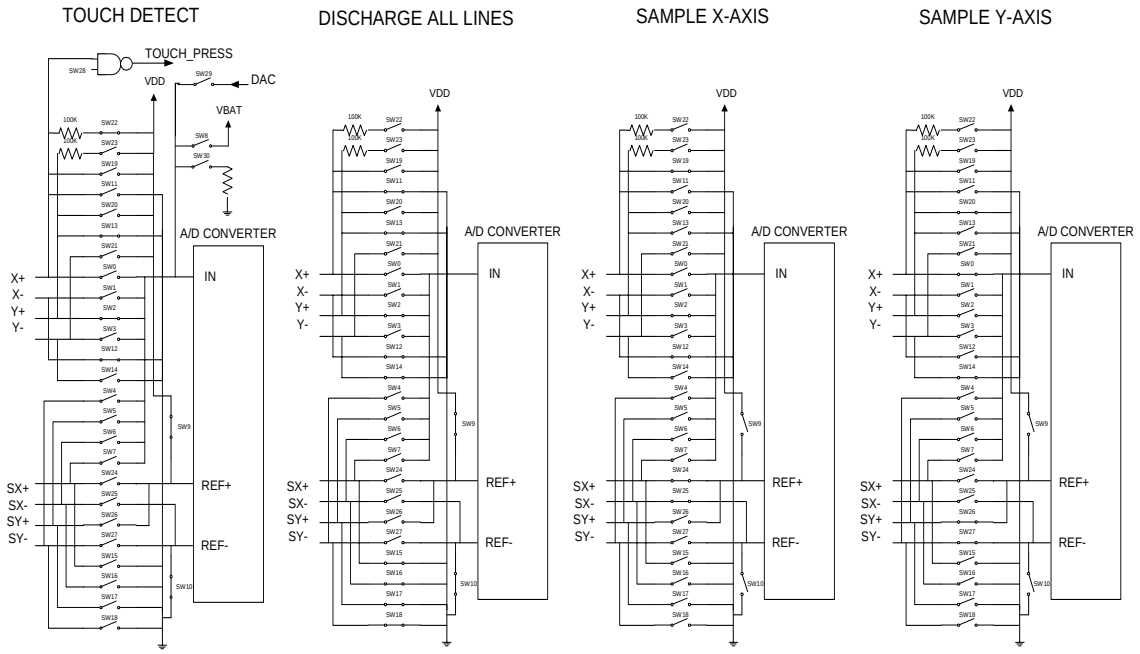


Figure 25-2. 8-Wire Resistive Interface Switching Diagram

Copyright 2007 Cirrus Logic

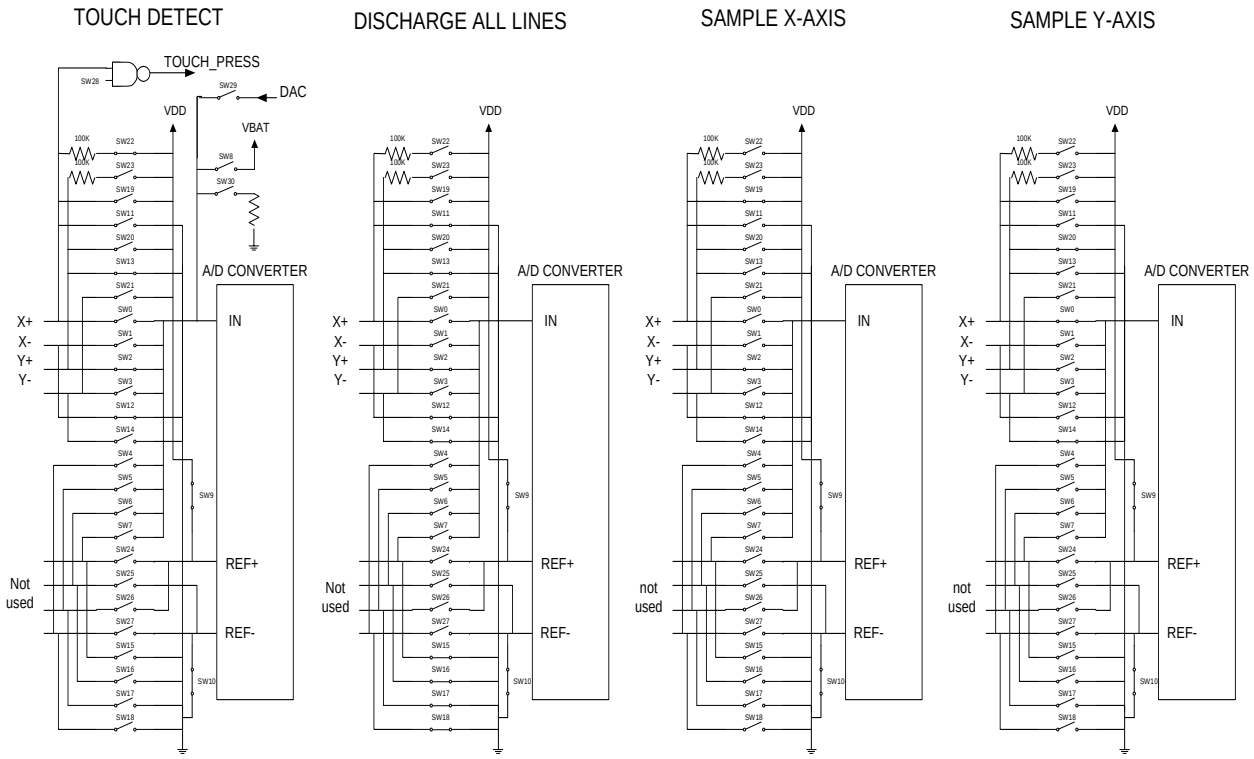


Figure 25-3. 4-Wire Analog Resistive Interface Switching Diagram

Copyright 2007 Cirrus Logic

The algorithm begins by putting the touch screen into its touch detect settling state for up to 1024 µsec as determined by the DLY value in the TSSetup register. After the delay value, the algorithm moves to the touch detect state. The switches in the settling and touch detect states are controlled by the TSDetect register value. The algorithm stays in the touch detect state until a touch is detected.

Table 25-2. Touch Screen Switch Register Configurations

Register Name	4-Wire	5-Wire	7-Wire	8-Wire
TSDetect (If TSSetup2.S28EN = 0)	0x0040_3604	0x0042_0620	0x0042_0620	0x0040_3604
TSDetect (If TSSetup2.S28EN = 1)	0x1040_3604	0x1042_0620	0x1042_0620	0x1040_3604
TSDischarge	0x0007_FE04	0x0002_2E20	0x030A_3020	0x0007_FE04
TSXSample	0x0008_1604	0x001D_D620	0x0318_5020	0x0308_1004
TSYSample	0x0010_4601	0x002D_B620	0x0328_3020	0x0C10_4001

Once a touch is detected, the algorithm moves to the discharge X state and stays there for up to 1024 µsec as determined by the DLY value in the TSSetup register. The switches in the discharge state are controlled by the TSDischarge register value. After the delay value, the algorithm moves to the sample X state. In this state, the analog switches are set to the TSXSample register. For example, for an 8-wire touch screen, VDD voltage is applied to X+, with X- held at ground potential, and the SX+ and SX- lines are connected as the reference for the A / D. The sample X state is also held for up to 1024 µsec as determined by the DLY value in the TSSetup register before any samples are taken.

At the end of the delay, the logic begins to take A / D samples. The initial read is a convert command and data from the initial sample is discarded. The number of samples taken is 4, 8, 16, or 32 as determined by the NSMP value in the TSSetup register. Each sample is compared with a min and max register to determine the range of samples taken. The min register is initialized to a value of 4095 and the max register is initialized to a value of 0. Any data points sampled will fall within this range and the min and max stored sample values will be adjusted based on the comparison. In addition, as the samples are taken, a running accumulator adds each 12-bit sample to a 17-bit total. After all samples are taken, the stored min is subtracted from the stored max for the sample set. The difference is compared to 4, 8, 12, 16, 24, 32, 64, or 128 as determined by the DEV value in the TSSetup register. If the range exceeds the deviation allowed, the results are scrapped and the logic starts over with initialization and detection of a valid touch. This allows a data set to be screened for bad points (possibly caused by noise or removing a press) that would adversely affect an average value. If the range does not exceed the maximum deviation allowed, the resulting value in the accumulator register is shifted by 2, 3, 4, or 5 places to divide by the number of samples as determined by the NSMP value in the TSSetup register. This generates the average for the sample set for a new X value.

Copyright 2007 Cirrus Logic

The difference between this new X value and the last valid X value is then compared against the XMIN value stored in the TSXYMaxMin register (unless the X interrupt pending flag is set). If the difference is less than this value no action is taken, and the algorithm continues by discharging and scanning the Y-axis. If the difference between the new X value and the last X value is greater than the XMIN value, the algorithm continues by comparing the difference between the new X value and the last X value to the XMAX value in the TSXYMaxMin register. If the difference is greater than XMAX, it is assumed that this distance is too far for a touch input to possibly move in the short scan time interval and that the key press is invalid. However, it is stored as the last recorded X location in case a truly new location is being determined. The X move interrupt pending flag is also set at this point to cause the algorithm to skip over the comparison to XMIN on consecutive sample sets. This flag will also cause a ARM Core interrupt after valid X and Y samples have been established.

The algorithm then starts again, discharging and detecting a valid press. If the difference between the new X value and the last stored X value is less than the XMAX value, the algorithm stores the new X value, sets the X move interrupt pending flag and continues by discharging and scanning a Y-axis value. Figure 25-4 is the flow chart demonstrating the scanning process.

The Y-axis scan proceeds in the same fashion as the X-axis scan, except that the switch matrix is now controlled by the TSYSample register. When the new Y value has been calculated, the difference between this new Y value and the last valid Y value is then compared against the YMIN value stored in the TSXYMaxMin register (unless the Y interrupt pending flag is set). If the difference is less than this value, no Y action is taken and the algorithm continues by checking the X interrupt pending flag. If this flag is set the Y value is stored in the Y last register and the interrupt to the ARM Core is generated.

Copyright 2007 Cirrus Logic

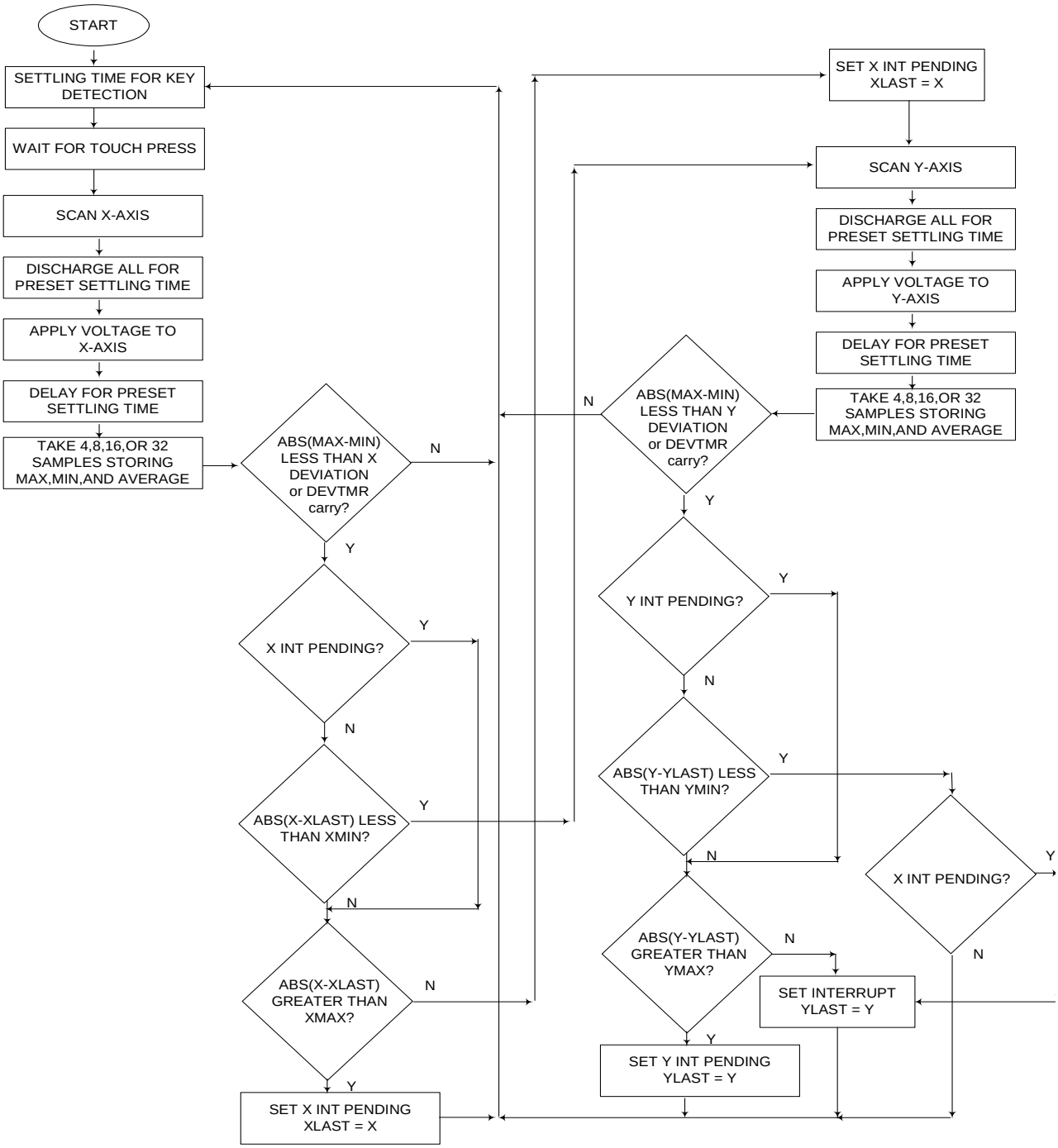


Figure 25-4. Analog Resistive Touch Screen Scan Flow Chart

Copyright 2007 Cirrus Logic

The algorithm then would continue by discharging and detecting a valid touch. With no X interrupt pending flag, the algorithm also continues by discharging and detecting a valid touch, but without interrupt generation. If the difference between the new Y value and the last Y value is greater than the YMIN value, the algorithm continues by comparing the difference between the new Y value and the last Y value to the YMAX value in the TSXYMaxMin register. If the difference is greater than YMAX, it is assumed that this distance is too far for a touch input to possibly move in the short scan time interval and that the key press is invalid. However, it is stored as the last recorded Y location in case a truly new location is being determined (that is, no contact to contact or contact to no contact).

The Y move interrupt pending flag is also set at this point to cause the algorithm to skip over the comparison to YMIN on consecutive sample sets. This flag will also cause an ARM Core interrupt as soon as a valid Y sample can be established. The algorithm then starts over discharging and detecting a valid press. If the difference between the new Y value and the last stored Y value is less than the YMAX value, the algorithm stores the new Y value in the Y last register, and the stored X value is moved to the X last register. The interrupt to the ARM Core is generated, and the algorithm continues by moving to the settle state and detecting a valid press. The interrupt is cleared when the TSXYResult register is read or by clearing the TSSetup2.TINT bit.

As can be seen from Figure 25-3, 4-wire touch screens are operated in much the same way a 8-wire touch screens, except that the reference for the A / D converter is internal instead of a feedback. The register values TSDetect, TSDischarge, TSXSample, and TSYSample are derived from the switch positions in the diagram. They are documented in Table 25-2.

25.2.2 Five-wire and Seven-wire Operation

Five-wire touch screens require a different connection scheme than 4- or 8-wire touch screens. The X+ and X- lines become static V+ and V- lines. The Y+ and Y- lines are Z+/- and Z-/+ and are used to switch between X and Y axis. The reference for the A / D converter is internal. One of the feedback lines, for example, sY+ is used as a Wiper input to the A / D converter. The register values TSDetect, TSDischarge, TSXSample, and TSYSample are derived from the switch positions in the diagram. They are provided in Table 25-2.

Seven-wire touch screens (5-wires with feedback) are very similar to a normal 5-wire touch screen, except that the A / D reference is taken from the external circuit. The X+ and X- lines are still static V+ and V-lines. The Y+ and Y- lines are still Z+/- and Z-/+ and are used to switch between X and Y axis. However, the reference for the A / D converter is on the sX+ and sX- lines relabelled as sV+ and sV-. The sY+ feedback line is still used as a Wiper input to the A / D converter.

The register values TSDischarge, TSXSample, and TSYSample are derived from the switch positions in the diagram. They are provided in Table 25-2.

Copyright 2007 Cirrus Logic

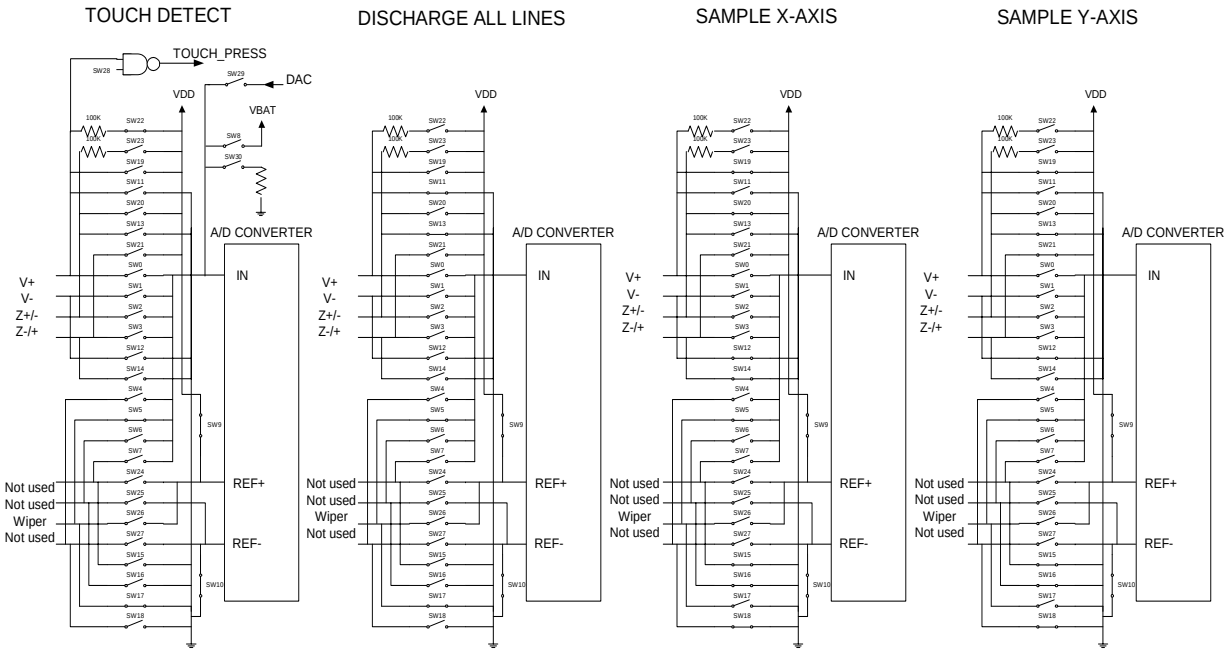


Figure 25-5. 5-Wire Analog Resistive Interface Switching Diagram

Copyright 2007 Cirrus Logic

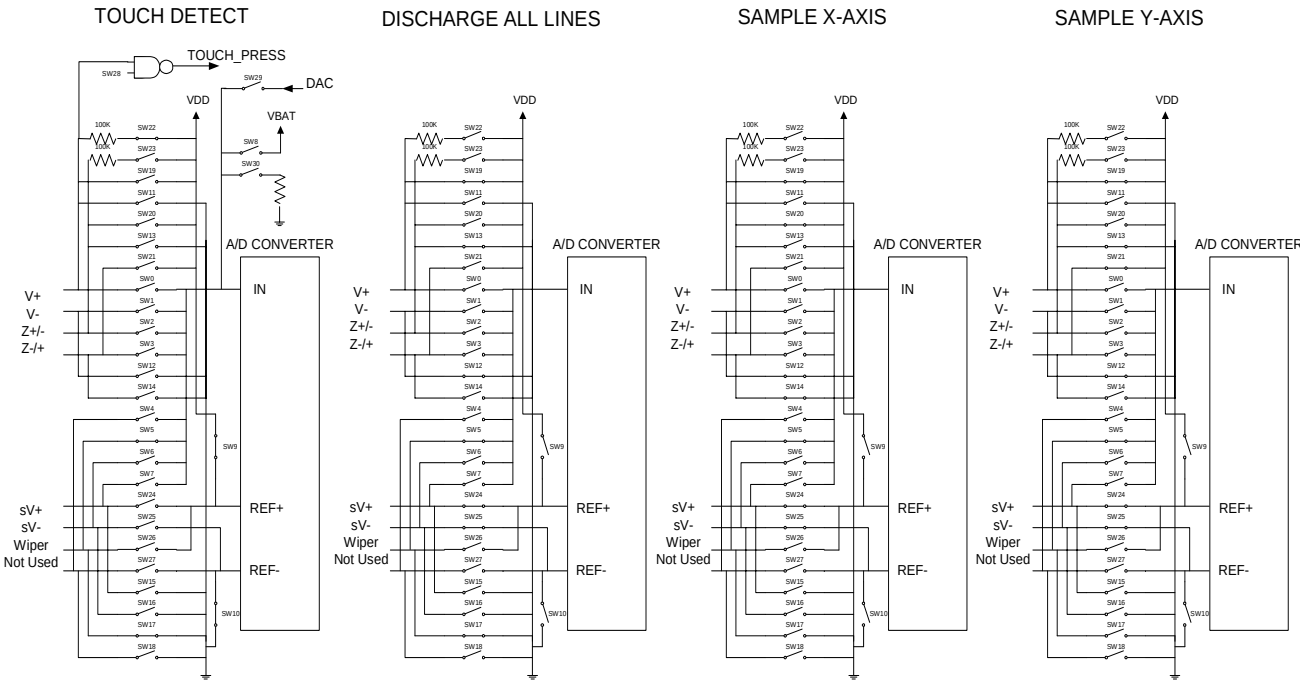


Figure 25-6. 5-Wire Feedback (7-Wire) Analog Resistive Interface Switching Diagram

25.2.3 Direct Operation

When the touch screen controller is disabled (TSSetup.ENABLE low), the ARM Core has direct control of the analog switch array through the TSDirect register. The full 12-bit output of the analog-to-digital converter can also be read from the TSXYResult register when the touch screen engine is disabled. **Please note that the initial read value should be viewed as a convert command where the data provided is stale and should be discarded.** After the conversion time for the A / D converter, the actual value may be read. This also sets off another convert command.

The next few figures demonstrate some special functions that can be implemented with direct ARM Core control of the switch array. The touch screen can be set up in a way that it can be disabled for low power mode and still have the ability to interrupt the ARM Core. Figure 25-7 shows how to detect a key press in either 4 / 8-wire or 5-wire installations with the touch screen controller disabled. The **TOUCH_PRESS** signal in the figure is gated into the interrupt logic when the touch screen controller is disabled and in low power mode. In this mode, the clock to the module can be disabled and interrupts will still be generated. The low-power mode should be entered and exited with the touch screen interrupt disabled, as the asynchronous operation of this logic could cause glitches on the interrupt line. Entering the low-power state with interrupts enabled may trigger false interrupts.

Copyright 2007 Cirrus Logic

The register values for TSDirect can be derived from the switch positions in the diagram. A "1" in the register bit position indicates that the switch is made or closed. When TSSetup.S28EN is low, the TSDirect value for 4- or 8-wire touch press detection should be set to 0x0040_4601. Otherwise, when TSSetup.S28EN is high, the TSDetect value should be 0x1040_4601. And, when TSSetup.S28EN is low, the TSDirect value for 5-wire touch press detection should be set to 0x0042_0601. Otherwise, when TSSetup.S28EN is high, the TSDirect value should be 0x1042_0601.

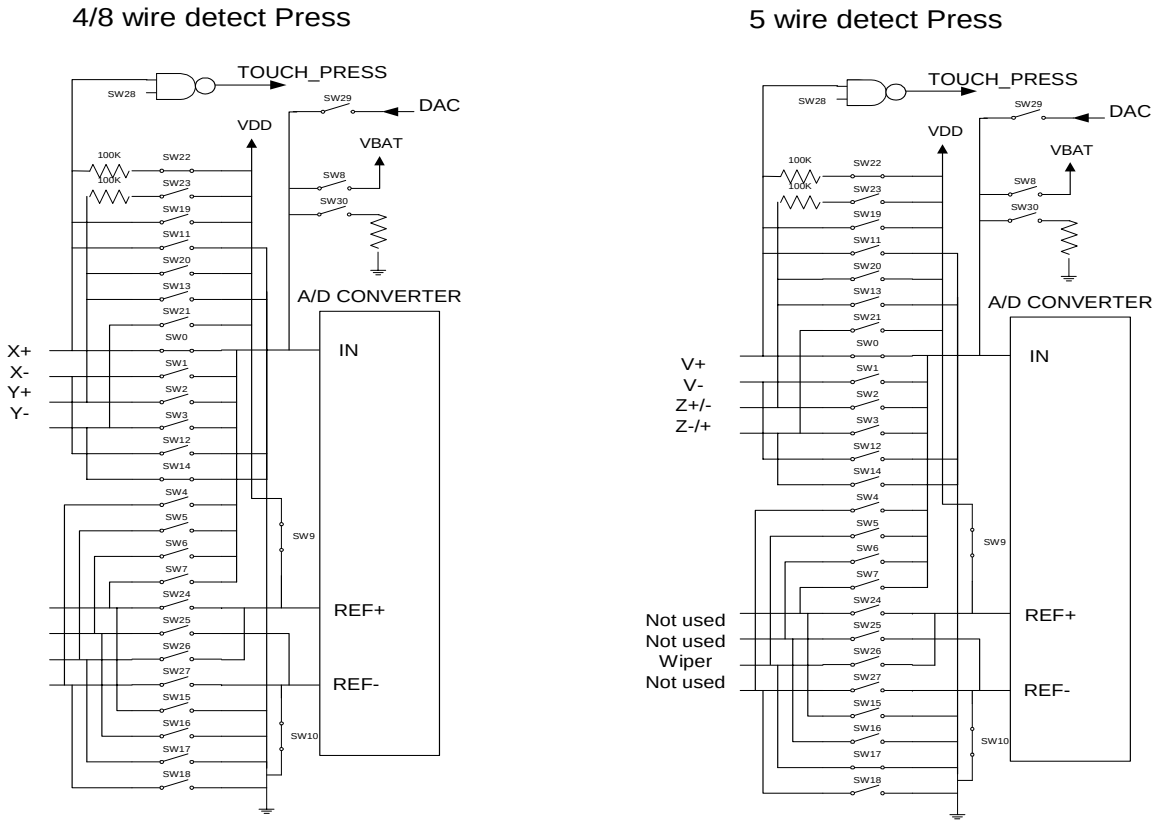


Figure 25-7. Power Down Detect Press Switching Diagram

25.2.4 Measuring Analog Input with the Touch Screen Controls Disabled

The analog switch array can be used to measure the chip battery voltage, Digital to Analog Converter feedback, or other miscellaneous analog inputs when the touch screen controller is disabled. Figure 25-8 shows the switch configuration for reading these values. Note that any extra reference lines that are not used for 4-wire or 5-wire touch screens can be read from the APB bus by temporarily disabling the touch screen controller. Please note that the initial read should be viewed as a convert command where the data provided is stale and should be discarded.

Copyright 2007 Cirrus Logic

The register values for TSDirect can be derived from the switch positions shown in Figure 25-8. A "1" in the register bit position indicates that the switch is made or closed. Therefore, the TSDirect value for battery sampling should be set to 0x4000_0700, the TSDirect value for the measuring the DAC feedback example should be set to 0x2000_0600, and the TSDirect value for measuring a miscellaneous input example should be set to 0x0000_0601.

To use the miscellaneous measurement input, TSDIRECT should be set to 0x0000_0601 and the input should be on **Xp** pin. By using this configuration, we can measure the FULL range, from 0 to 3.3 V. Also, to use this function we need to configure 2 items in Syscon:

- Enable the Clock for the Touch Screen (KTDIV register in Syscon)
- Set the TIN bit in DeviceCfg in Syscon. This sets the Touch Screen controller to an inactive state.

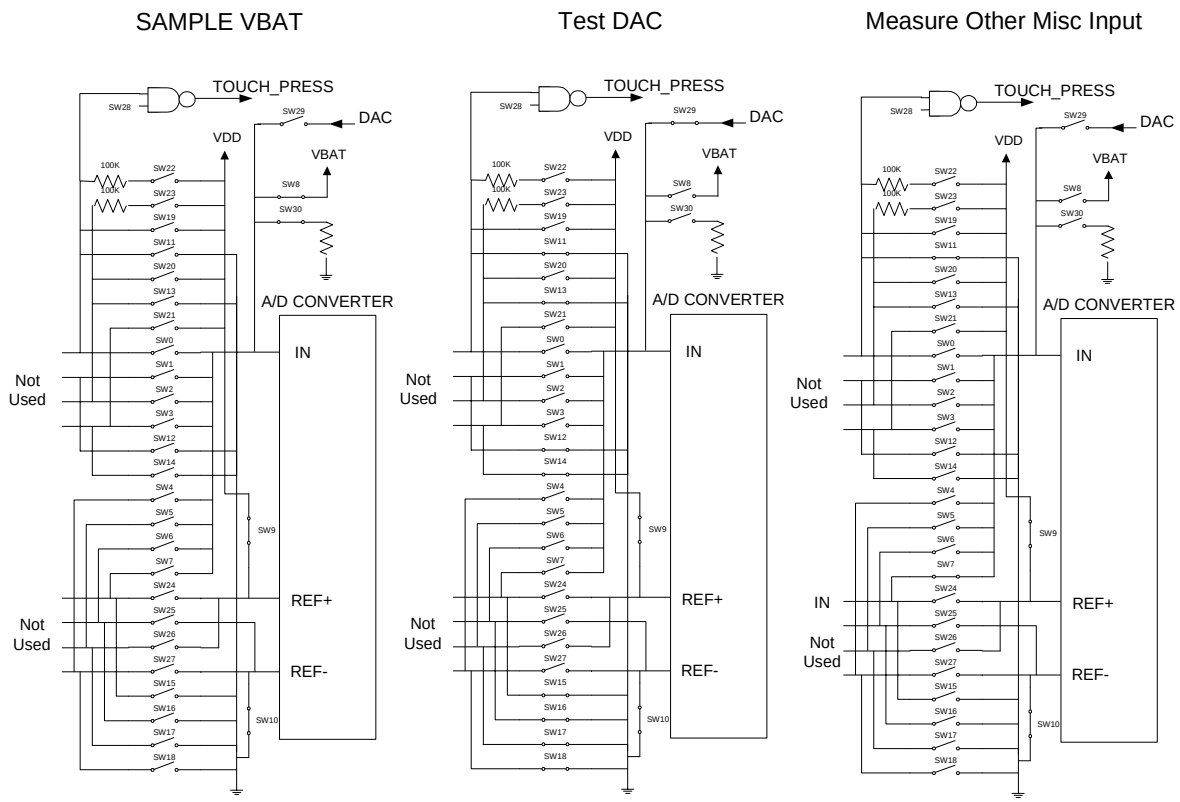


Figure 25-8. Other Switching Diagrams

Copyright 2007 Cirrus Logic

25.2.5 Measuring Touch Screen Resistance

The analog switch array can be configured to get an approximation of touch screen resistance. This may be useful for advanced touch screen algorithms that either try to determine how hard a touch screen is being pressed, or if it is being pressed in more than one location. The touch screen controller built-in algorithm does not use this feature. More advanced algorithms would need to perform either primary or additional scanning through the APB interface. Please note that the initial A / D converter read should be viewed as a convert command where the data provided is stale and should be discarded.

The register values for TSDirect can be derived from the switch positions in the diagram. A "1" in the register bit position indicates that the switch is made or closed. Therefore, the TSDirect value for 4- or 8-wire X-axis resistance measurement should be set to 0x0040_1601, the TSDirect value for 4- or 8-wire Y-axis resistance measurement should be set to 0x0080_4604, and the TSDirect value for 5-wire resistance measurement should be set to 0x0080_4604. See Figure 25-9.

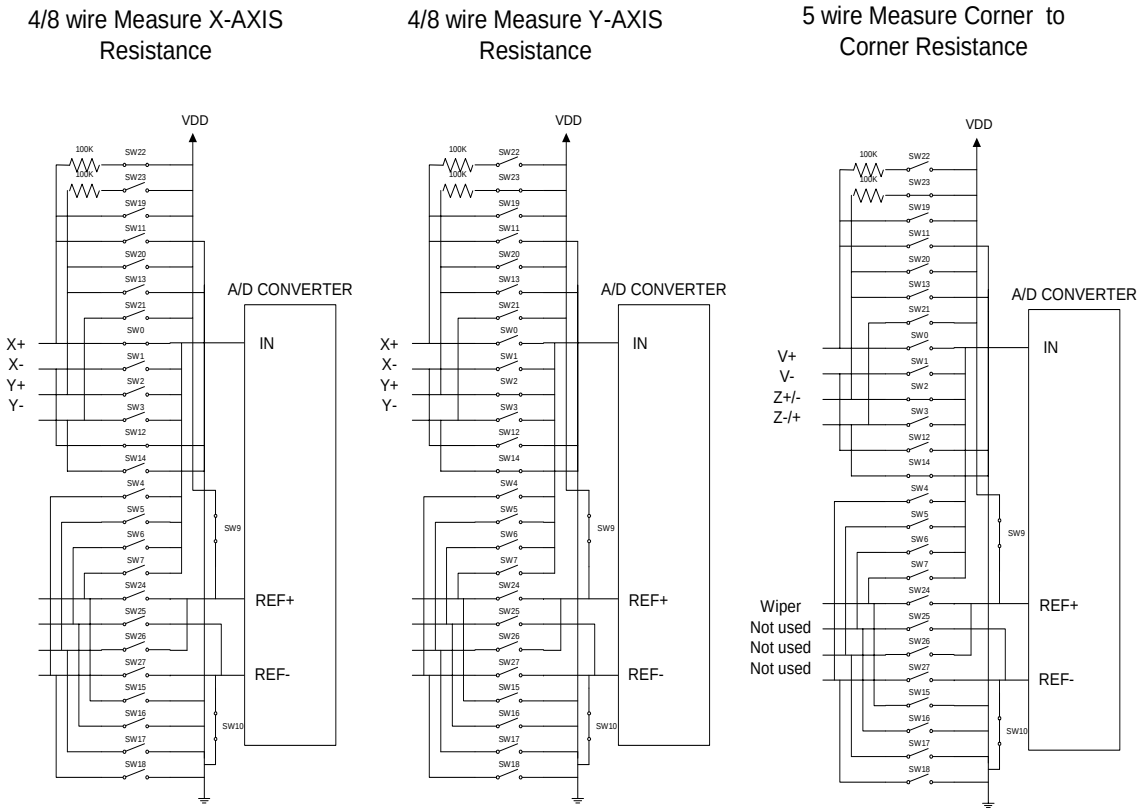


Figure 25-9. Measure Resistance Switching Diagram

Copyright 2007 Cirrus Logic

25.2.6 Polled and Interrupt-Driven Modes

The ADC provides support for synchronous sampling, in both polled mode and interrupt-driven mode. In either mode, the touch screen scanning state machine should be disabled by setting bit 15 of the TSSetup register to zero.

The ADC decimation filter conversion value appears in the TSXYResult register. With the touch screen scanning state machine disabled, bit 31 of this register is the SDR, or Synchronous Data Ready, bit. This bit is set when a new valid conversion value appears in this register, and is cleared when this register is read. Hence, in polled mode, the TSXYResult register may be read repeatedly. If two consecutive reads show the bit clear and then set, the second read has a new valid value.

Conversion data may also be processed using interrupts from this module. If bit 11 in the TSSetup2 register (the RINTEN bit) is set, an interrupt occurs whenever the SDR bit in the TSXYResult register is set. Therefore, an interrupt handler can read the TSXYResult register whenever a new valid sample appears in the register, which both returns a new conversion value and clears the interrupt.

25.2.7 Touch Screen Package Dependency

The Touch block uses the following external pins

Table 25-3. External Signal Functions

Signals	Function
DAC_VDD	Should be connected to ADC_VDD on the board.
ADC_VDD	Touch Screen ADC power, nominal 3.3V. This supply is internally diode connected to the DAC_VDD supply to provide additional ESD protection. Should be connected to DAC_VDD on the board.
Xp **Xm**	Touch Screen ADC X axis analog bias output.
Yp **Ym**	Touch Screen ADC Y axis analog bias output.
sXp **sXm**	Touch Screen ADC X axis voltage feedback inputs.
sYp **sYm**	Touch Screen ADC Y axis voltage feedback inputs.
ADC_GND	Touch Screen ADC ground

Copyright 2007 Cirrus Logic

25

25.3 Registers

Table 25-4. Analog Touch Screen Register Memory Map

Address	Name	SW locked	Type	Size	Description
0x8090_0000	TSSetup	No	Read/Write	26 bits	Analog Resistive Touch Screen controller setup control register.
0x8090_0004	TSXYMaxMin	No	Read/Write	32 bits	Analog Resistive Touch Screen controller max/min register.
0x8090_0008	TSXYResult	No	Read Only	32 bits	Analog Resistive Touch Screen controller result register.
0x8090_000C	TSDischarge	Write	Read/Write	28 bits	Analog Resistive Touch Screen controller Switch Matrix control register.
0x8090_0010	TSXSample	Write	Read/Write	28 bits	Analog Resistive Touch Screen controller Switch Matrix control register.
0x8090_0014	TSYSample	Write	Read/Write	28 bits	Analog Resistive Touch Screen controller Switch Matrix control register.
0x8090_0018	TSDirect	Write	Read/Write	28 bits	Analog Resistive Touch Screen controller Switch Matrix control register.
0x8090_001C	TSDetect	Write	Read/Write	28 bits	Analog Resistive Touch Screen controller Switch Matrix control register.
0x8090_0020	TSSWLock	NA	Read/Write	1-bit read 8-bit write	Analog Resistive Touch Screen controller software lock register.
0x8090_0024	TSSetup2	No	Read/Write	9 bits	Analog Resistive Touch Screen controller setup control register #2.

Note: The touch screen controller block decodes APB address bits PADR[6:2] only. If the decode for the PSEL_HATSH APB block select is a larger block size in the APB decoder, the registers will be repeated through memory. Touch screen controller registers are intended to be word accessed only. Since the least significant bytes of the address bus are not decoded, byte and half word accesses are not allowed and may have unpredictable results.

Register Descriptions

TSSetup

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDTCT	RSVD					DLY									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN	DEV			NSMP		SDLY									

Address:

0x8090_0000

Default:

0x0000_0000

Copyright 2007 Cirrus Logic

Mask:

03FF_FFFF

Definition:

Analog Touch screen Setup and Deviation Register.

Bit Descriptions:

RSVD:	Reserved. Unknown during read.
TDTCT:	TouchDetect Read only bit. Allows the ARM Core the ability to read the state of the **TOUCH_DETECT** line.
SDLY:	Defines the amount of settling time between A / D samples from 3 to 1024 µsec assuming a 1 MHz clock.
NSMP:	Defines the number of samples averaged per X or Y reading. 00 - 4 samples 01 - 8 samples 10 - 16 samples 11 - 32 samples
DEV:	Defines the amount of max to min range deviation in a sample set allowed before rejection. 000 - +/- 4 LSBs 001 - +/- 8 LSBs 010 - +/- 12 LSBs 011 - +/- 16 LSBs 100 - +/- 24 LSBs 101 - +/- 32 LSBs 110 - +/- 64 LSBs 111 - +/- 128 LSBs
ENABLE:	Enables the touch screen scanning state machine. 0 - Disabled. 1 - Enabled.
DLY:	Defines the amount of settling time between changes to the touch screen drive conditions from 3 to 1024 µs assuming a 1 MHz clock.

Copyright 2007 Cirrus Logic

TSXYMaxMin

Address:

0x8090_0004

Default:

0x0000_0000

Definition:

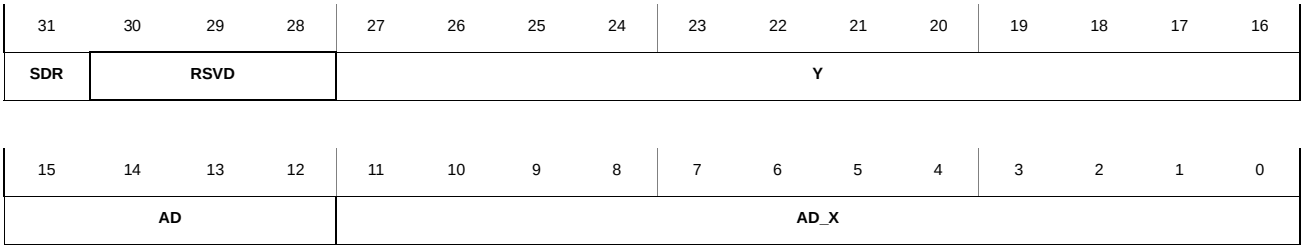
Analog Touch screen MAX and MIN move Register.

Bit Descriptions:

YMAX[11:4],XMAX[11:4]:Defines the amount of x-y distance from a previous touch that represents an invalid data point. The user could not move to a new location this many pixels away within the scan time limit. The definition can be from 16 to 4096, in increments of 16 decode locations, in a 4096 x 4096 decode array.

YMIN[7:0],XMIN[7:0]:Defines the amount of x-y distance from a previous touch value to a new touch value for a touch detection to initiate another interrupt to the ARM Core. The x-y box definition can be up to 512 x 512 (+/-256).

TSXYResult



Address:

0x8090_0008

Default:

0x0000_0000

Definition:

Analog Touch screen X and Y result Register.

Copyright 2007 Cirrus Logic

Bit Descriptions:

RSVD:	Reserved. Unknown during read.
SDR:	Synchronous Data Ready. This bit is set when new conversion data from the ADC digital filter appears the TSXYResult register. The bit is cleared when the TSXYResult register is read. The bit is always clear unless the touch screen scanning state machine is disabled (TSSetup bit 15 is zero).
Y, AD_X:	X and Y controller calculated reading when touch screen controller is enabled at 12-bit resolution. The interrupt output is cleared when this register is read.
AD, AD_X:	Direct analog-to-digital controller output when touch screen controller is disabled at 16-bit resolution.

TSDischarge, TSXSample, TSYSample, TSDirect, TSDetect

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD		SCTL[29]		SCTL											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCTL															

Address:

TSDischarge: 0x8090_000C
TSXSample: 0x8090_0010
TSYSample: 0x8090_0014
TSDirect: 0x8090_0018
TSDetect: 0x8090_001C

Default:

0x0000_0000

Definition:

Analog switch control Registers.

Bit Descriptions:

RSVD:	Reserved. Unknown during read.
SCTL[29]:	Analog switch control value for direct analog switch control only (TSDirect) when the touch screen controller is disabled. Controls DAC routing to ADC input.

Copyright 2007 Cirrus Logic

SCTL: Analog switch control values for the touch controller touch detect, discharge, sample X, and sample Y states and direct analog switch control when the touch screen controller is disabled. A "1" indicates that the switch is made or closed. A "0" indicates that the switch is open.

Table 25-2 contains the values that must be loaded into the switch registers, depending on the type of touch screen being used.

TSSWLock

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								SWLCK							

Address:
0x8090_0020

Default:
0x0000_0000

Definition:
Software lock register.

Bit Descriptions:

RSVD: Reserved. Unknown during read.

SWLCK: Software lock bits.

WRITE: The Unlock value for this feature is 0xAA. Writing 0xAA to this register will unlock all locked registers until the next block access. The ARM lock instruction prefix should be used for the two consecutive write cycles when writing to locked chip registers.

READ: During a read operation SWLCK[0] has the following meaning:
1 = Unlocked for current bus access.
0 = Locked

The Read feature of the SWLOCK register is used for testing the locking function. Since the software lock only remains unlocked for the next block cycle, this test must be performed on two consecutive cycles using the ARM lock instruction prefix. The contents of SWLCK[7:1] are unknown during a read operation.

Copyright 2007 Cirrus Logic

TSSetup2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				RINTEN	S28EN	NSIGND	DISDEV	DTMEN	DINTEN	DEVINT	PINTEN	PENSTS	PINT	NICOR	TINT

Address:

0x8090_0024

Default:

0x0000_0000

Definition:

Touch screen Setup Register #2.

Bit Descriptions:

RSVD: Reserved. Unknown during read.

RINTEN: Synchronous Data Ready Interrupt Enable. Setting this bit results in an interrupt whenever the Synchronous Data Ready (SDR) bit in the TSXYResult register is set. The SDR bit will never be set unless the EN bit of the TSSetup register is clear.

S28EN: Switch 28 Enable. The touch detect NAND gate can be triggered separately by bit 28 of the switch registers or in conjunction with bit 22 which also controls the X+ pullup.
0 = NAND gate controlled by bit 22.
1 = NAND gate controlled by bit 28.

NSIGND: Unsigned ADC output type. The touch input information can be processed as signed or unsigned integers. The default bit value is "0" for signed.

DISDEV: Disable Deviation check for both X and Y ADC sampling. Setting this bit high causes the deviation test to always pass and forces a sample set.

DTMEN: Deviation Timer Enable. Setting this bit high enables the timeout for the deviation check. If the deviation check fails 255 times for either the X or Y axis the algorithm will skip this check and force a sample set anyway.

DINTEN: Deviation Error Interrupt Enable. Setting this bit high causes an interrupt when the sample deviation check fails 255 times for either the X or Y axis. The DTMEN bit must be high for this bit to be effective.

Copyright 2007 Cirrus Logic

DEVINT: Deviation Interrupt. This is the deviation error interrupt. When the DINTEN and DTMEN bits are set high and an axis fails the deviation test 255 times causing an interrupt, this bit must be written to a "0" to clear the interrupt.

PINTEN: Pen up Interrupt Enable. Setting this bit high causes an interrupt when the algorithm first detects a pen up condition.

PENSTS: Pen Status. This bit allows access to directly read the status of the pen up / down indicator. Read only.
0 - pen up.
1 - pen down.

PINT: Pen up Interrupt. This is the Pen up interrupt. When the PINTEN bit is set and a pen up condition is detected after a pen down, this bit will be set high and cause the interrupt output to go high. This bit may be written high for test purposes and written low to clear the interrupt.

NICOR: No Interrupt Clear on Read. This bit controls clearing of the touch interrupt.
0 - TINT clears when reading TSXYResult register.
1 - The TINT bit must be written low to clear the interrupt.

TINT: Touch Interrupt. This is the touch screen activation interrupt. When a new stable set of X and Y ADC values is resolved, this bit will activate. It may be triggered from an off-screen unpressed stable value. Writing a "0" to this bit will clear the touch interrupt. This bit may be written high for test purposes.

Copyright 2007 Cirrus Logic

25

Copyright 2007 Cirrus Logic

Chapter 26

Keypad Interface

26.1 Introduction

26

Note: This chapter applies only to the EP9307, EP9312, and EP9315 processors.

The keypad interface has the following features:

- A maximum 8x8 array of normally open, single pole contacts
- A back drive feature to minimize capacitance effects
- A typical scan count limit of 3 consecutive scans
- A maximum mechanical bounce time for a key press of 20 milliseconds
- A typical interrupt interval between 24 and 44 milliseconds
- A low-power wakeup mode
- A three-key reset

If the system does not use a keyboard, the Row[7:0] and Column[7:0] pins can be remapped to General Purpose Input/Output (GPIO) pins. For details, see Chapter 5, "DeviceCfg" on page 5-25 and Chapter 28, Table 28-4 on page 28-8.

A block diagram for the key array scanning circuitry is shown in Figure 26-1.

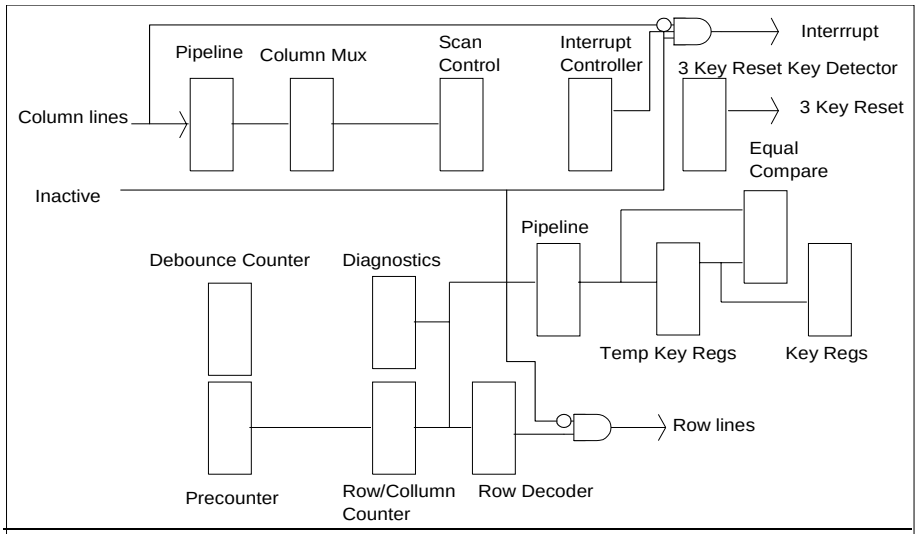


Figure 26-1. Key Array Block Diagram

Copyright 2007 Cirrus Logic

26.2 Theory of Operation

The circuitry scans an array of up to 64 keys. The keys are normally open, single pole contacts arranged in an array of 8 rows by 8 columns. The array may be partially filled. The key array rows are designated as ROW0 through ROW7, and the columns are designated as COL0 through COL7.

Any 1 or 2 keys in the array that are pressed are de-bounced and decoded. If more than 2 keys are pressed, only the keys or apparent keys in the array with the lowest address will be decoded.

Keys or apparent keys with address values greater than that of the lowest two will be ignored. An apparent key is a condition that may occur when more than 2 keys are pressed. Apparent keys are caused by alternate current paths in the key array.

A key address is the binary row number concatenated with the binary column address. Key addresses range from 0x00 to 0x3F. A diagram of the key array is shown in Figure 26-2.

The circuitry scans the key array by driving each ROW line low, one line at a time. At the end of each ROW time period, column data is read. The key array column lines are registered and decoded by a multiplexer. The column address selects the column multiplexer input. Each of the column lines is passively pulled up by the chip.

When a key is pressed, the column line for the key will be driven low when the row which contains the key is driven low. On the next key array scan, the output of the multiplexer in the chip will be asserted active when the row and column is the same as the key address. When the multiplexer output is active then the key is detected and the address is stored, if it is one of the first two from the start of a key array scan.

When more than 2 keys are pressed, it is possible to detect apparent keys. Apparent keys look like actual pressed keys to the device but are not. An example of an apparent key is described in "Apparent Key Detection".

Copyright 2007 Cirrus Logic

Figure 26-2. 8 x 8 Key Array Diagram

26.2.1 Apparent Key Detection

When more than two keys are pressed, the key array controller may detect "apparent" keys. An apparent key detection is caused by misinterpreting the basic electrical signals. For example, in Figure 26-3, three keys are pressed:

- (ROW0, COL3) with address 0x03
- (ROW3, COL0) with address 0x18
- (ROW3, COL3) with address 0x1B

The controller's instruction is to decode the two keys with the lowest addresses. Therefore the system interprets the electrical signals as:

- An apparent key address of 0x00 at (ROW0, COL0)]
- An actual key address of 0x03 at (ROW0, COL3)
- No press for address 0x18 at (ROW3, COL0)

Copyright 2007 Cirrus Logic

- No press for address 0x1B at (ROW3, COL3)

The ignored addresses, 0x18 and 0x1B, are greater than the addresses of the two keys detected. The following controller actions occur:

- ROW0 is driven low.
- When ROW0 is low, COL0 and COL3 are also low due to the current paths formed by the keys pressed.
- During the time that ROW1 is low each of the columns (COL0 through COL7) is scanned.
- Since COL0 is low the key 0x00 appears to be pressed.

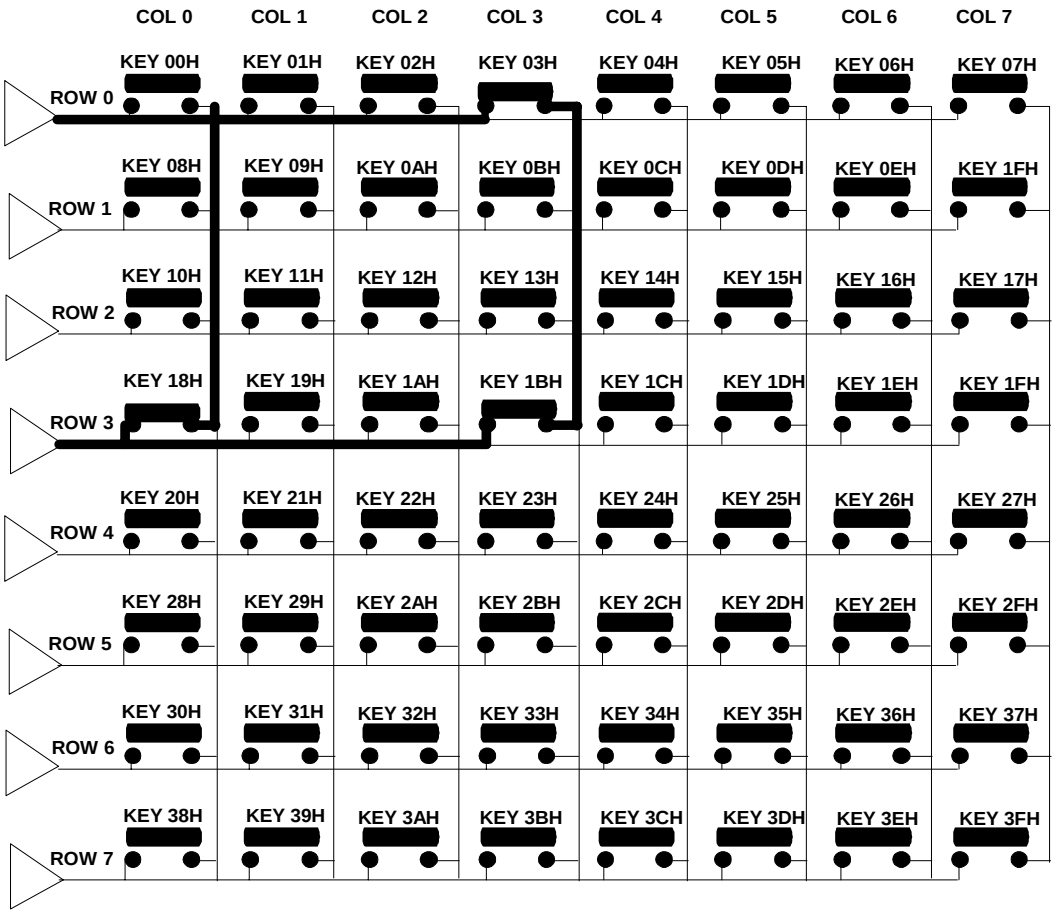


Figure 26-3. Apparent Key 00H

Copyright 2007 Cirrus Logic

26.2.2 Scan and Debounce

Products are scanned based on the KEY_SCAN register value. Each complete array scan starts with ROW7 and then progresses to ROW0, 1, and so on because of the pipelined nature of the key scan controller. Keys in this ROW have precedence and are considered first in the scan order because ROW7 is scanned first. When a key is pressed, it may mechanically bounce for a up to 20 msec. The key array scan circuit will count the number of consecutive scans that decode to the same 1 or 2 first keys encountered from the start of a scan.

When a preset scan count limit is reached (counted by the DEBOUNCE_COUNT counter), the key is considered to be de-bounced and an interrupt will occur. If a scan does not decode the same 1 or 2 first keys as the previous scan, then the scan count will be reset. The scan count limit is adjustable from 0 to 255 scans by writing to the DEBOUNCE_COUNT register in the chip. The register is written with the complement of the desired scan count limit. Typically a scan count limit of 3 is used.

Key arrays may have significant capacitance. If a key is pressed at a location ROWY, COLX, the capacitance associated with COLX is discharged when the ROWY line is driven low during the scan. When ROWY + 1 is driven, the capacitance associated with COLX must then charge to a logic 1 passively before the COL inputs are sampled. If not, an erroneous key press will be detected. For fast scan times, the time to charge the key array between column samples is reduced. To reduce the time to charge the key array, the back drive feature may be used. When back drive is enabled, the column lines and row lines are all driven high for a short period of time between ROW scanning time to charge the array capacitance.

26.2.3 Interrupt Generation

An interrupt is generated whenever the key scanner detects a new stable set of keys. This means that an interrupt occurs after a key is pressed and then after the key is released. An interrupt will also occur if the first or second pressed keys in the array change to different keys. When the interrupt occurs, the number of decoded keys pressed and the array coordinates of the pressed keys are stored. The interrupt signal is maintained by a flip flop. The interrupt flip flop is cleared when the key register is read by the ARM Core. The interrupt condition can also be read by the ARM Core. Interrupt conditions are 1 key, 2 keys, and no keys.

Assume that the keys may bounce for up to 20 msec and each scan is roughly 8 msec and the scan count limit is set to 3 then an interrupt will occur between 24 and 44 msec after a key is pressed or released. If a scan count limit of 0 is set by writing 0xFF to the de-bounce count pre-load register then an interrupt will only occur the first time a key is pressed. No further interrupts will occur because the DEBOUNCE_COUNT counter will always be reset to its terminal count.

If an interrupt is ignored, then a subsequent interrupt will be pending until the first interrupt is serviced. If further keypad activity occurs after an interrupt is pending then the most recent de-bounced and decoded event will become pending and the previous pending conditions will be lost.

Copyright 2007 Cirrus Logic

26.2.4 Low Power Mode

The key scanning block also supports a low power wake-up mode. In this mode, a key press generates a wake up interrupt. The key scan interrupt should be masked. Because the wake up interrupt is asynchronous, and depends on external keypad lines which may have a large capacitance value, glitches may occur on the interrupt when transitioning to low power mode. After transitioning, all clocks to the key scanning circuitry can be shut down. In the low power mode, all of the column line drivers should be in input mode in a high state due to the pad pull up resistors. The column inputs are ANDed together to detect any key presses. This signal directly toggles the interrupt output. The detect condition is not de-bounced.

26.2.5 Three-key Reset

The key scanning circuitry provides a three-key reset output by detecting keys (columns) 2, 4, and 7 activated in row 0. The three-key reset detect is used by the watchdog circuit to generate a three-key initiated reset to the system.

The output **RESET_KEYS_DETECTED** goes to the Watchdog block to indicate that a three-key reset is being requested.

26.3 Registers

Table 26-1. Keypad Interface Register Memory Map

Address	Name	SW locked	Type	Size	Description
0x808F_0000	KeyScanInit	No	Read/Write	24 bits	Key Scan Initialization Register
0x808F_0004	KeyDiagnostic	No	Read/Write	6 bits	Key Scan Diagnostic Register
0x808F_0008	KeyRegister	No	Read Only	16 bits	Key Value Capture Register

Note: Key scan controller registers are intended to be word accessed only. Since the least significant bytes of the address bus are not decoded, byte and half word accesses are not allowed and may have unpredictable results.

Register Descriptions

KeyScanInit

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD								DBNC							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIS3KY	DIAG	BACK	T2	NA				PRSCL							

Copyright 2007 Cirrus Logic

Address:

0x808F_0000

Default:

0x0000_0000

Definition:

Key scan initialization register.

Bit Descriptions:

RSVD:	Reserved. Unknown during read.
DBNC:	De-bounce start count. This value is used to pre-load the de-bounce counter. The de-bounce counter counts the number of consecutive scans that decoded the same keys. Terminal count for the de-bounce counter is 0xFF. Terminal count indirectly generates a key scan interrupt. A pre-load value of 0xFC will cause the key scan circuitry to count 3 identical consecutive keypad scans.
DIS3KY:	Disable 3 Key reset. Setting this bit high disables the three key reset output to the watchdog reset block. Setting it back low re-enables it.
DIAG:	Key scan diagnostic mode. Setting this bit high allows key scanning to be directly controlled through the key register by writes from the ARM Core. The DIAG.KEY[5:0] value is written by the ARM Core. Then the KeyRegister.K bit is read to determine if there is a key press. The result from reading the KeyRegister.K bit is not hardware de-bounced.
BACK:	Key scan back driving enable. Setting this bit high enables the key scanning logic to back drive the row and column pins of the chip high during the first two column counts in the row/column counter.
T2:	Test mode bit. When this bit is set to "1", the counter RC_COUNT is advanced by 8 counts when EN is active. The effect is that only column 0 is checked in each row. This test mode allows a faster test of the ROW pins.
NA:	Not Assigned. These bits will read back the value written.

Copyright 2007 Cirrus Logic

26

PRSCL: Row/Column counter pre-scaler load value. This value is used to pre-load the RC pre-scale counter. The pre-scale down counter counts the number of 1 MHz clocks for every step of the RC counter. When the pre-scale counter reaches 0, the RC counter steps. A pre-load value of 0x002 will cause the RC counter to step every three clocks. The PRSCL value should never be set to 0x000 or 0x001, except for high speed counter tests, as the key state machine will not resolve keys properly for these values.

Key array scan time = clock period * (PRSCL+1)[64keys]
Example scan time for PRSCL[9:0] = 0x0FA
Scan time = 1 µs * (249 +1) = 16 ms
if de-bounce = 0xFC, key detection interrupt is fired in approximately 48 ms.
Array scan time can range from 64 µsec. to 65.536 ms.

KeyDiagnostic

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												DIAG			

Address:
0x808F_0004

Default:
0x0000_0000

Definition:
Diagnostic key value register.

Bit Descriptions:

RSVD: Reserved. Unknown during read.

DIAG: Diagnostic key value. When diagnostic mode is enabled (KeyScanInit.DIAG high) and this register is written, the Row Column scan value is used to directly control the chip key matrix scan drivers and receivers. Results are read back via the KeyRegister.K bit.

Copyright 2007 Cirrus Logic

Chapter 27

IDE Interface

27.1 Introduction

27

Note: This chapter applies only to the EP9312 and EP9315 processors.

The IDE interface provides an industry standard connection to ATA/ATAPI compliant devices. A single IDE port is provided which will attach to master or slave devices. The interface will support up to:

- 2 Devices
- PIO Mode 4
- Multi-word DMA Mode 2
- Ultra DMA Mode 4

The IDE block will use the internal DMA controller to do the data transfers in Multiword DMA and Ultra DMA modes. The interface will support only 16 bit devices. The data transfer is always 16-bit wide, even for a non-data transfer in PIO mode, when only the lower 8 bits are valid.

27.2 Theory of Operation

The IDE host has one request line, **DMAide** to the DMA controller, used to request DMA service. It has an external interrupt line, **INTRQ**, from the device for interrupt service. It also has an internally generated interrupt signal INTide for reporting internal errors in the IDE host to the ARM Core.

The IDE port is connected to the external ATAPI device through a 28-pin interface. Of these 28 signals, 25 use dedicated pins, 2 share EGPIO pins (**EGPIO[2]** for **DMARQ** and **EGPIO[15]** for **DASPn**), and the device interrupt request uses one of the INT pins (**INT[3]**) for **INTRQ**.

The IDE interface hardware is composed of several elements: a GPIO like Pin Interface, a MDMA Transfer State Machine, a UDMA Transfer State Machine, a pair of Read and Write Data Buffer FIFOs, and a pair of CRC generation circuits.

The interface between the IDE host and the IDE device is defined in Table 27-1. The column labeled Type identifies the block associated with the processor pin. The GPIO type indicates the 2 pins that are shared with the EGPIO block. The INT type indicates the pin using one of the INT pins. The NI type indicates an IDE signal that is not supported. All others are dedicated pins.

Copyright 2007 Cirrus Logic

27

27.2.1 Diagrams and State Machines

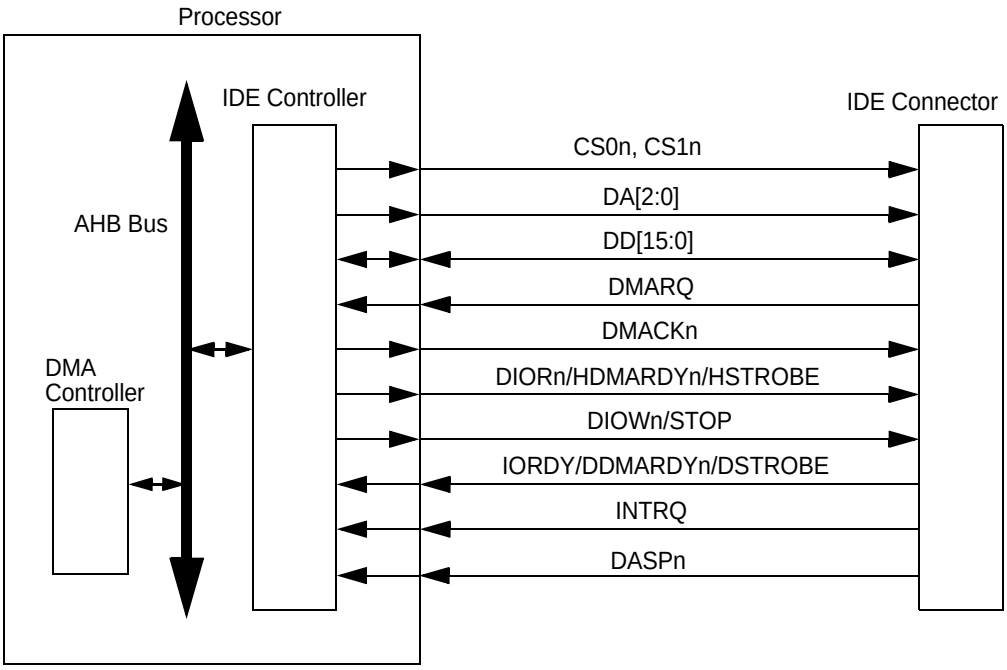


Figure 27-1. IDE Interface Signal Connections

Table 27-1. IDE Host to IDE Interface Definition

IDE Pin	Type	No. of Pins	Description
CS0n	-	1	Chip select for device registers with base address 0x1F0
CS1n	-	1	Chip select for device registers with base address 0x3F0
DA[2:0]	-	3	3-bit binary encoded address
DIORn/ HDMARDYn/ HSTROBE	-	1	Strobe signal to read device regs or data port/ Flow control signal for Ultra DMA data-in burst/ Flow control signal for Ultra DMA data-out burst
DIOWn/ STOP	-	1	Strobe signal to write device regs or data port/ Terminates an Ultra DMA burst
DMACKn	-	1	DMA acknowledge to DMARQ to initiate DMA transfers
DASPn	GPIO	1	Signal to indicate that a device is active, or that Device 1 is present
DMARQ	GPIO	1	DMA request for DMA to and from the controller
INTRQ	INT	1	Device interrupt

Copyright 2007 Cirrus Logic

Table 27-1. IDE Host to IDE Interface Definition (Continued)

IDE Pin	Type	No. of Pins	Description
IORDY/ DDMARDYn/ DSTROBE	-	1	Negate to extend the host transfer cycle of any host read or write access/ Flow control signal for Ultra DMA data-out burst/ Flow control signal for Ultra DMA data-in burst
IOCS16n	NI	1	Device indicates it supports 16-bit I/O bus cycles
PDIAGn/ CBLIDn	NI	1	Asserted by device 1 to indicate to device 0 that it has finished diagnostic/ Cable assembly type identifier
DD[15:0]	-	16	16-bit data interface between controller and device

Note: NI = Not supported at this time.

27.2.2 PIO Operations

For PIO operations, the Pin Interface unit handles all the operations. Register read and write operations by the host are sufficient to operate the IDE interface in PIO mode for both non-data and data transfer in both directions.

Most IDE controllers handle this automatically, but this IDE controller does not. The diagrams are located in Information Technology AT Attachment with Packet Interface (ATA/ATAPI-5), Section 10.2.2, Figure 44. See "Preface" chapter, "Reference Documents" on page P-3 for additional information.

Initial state: pins **DIORn** and **DIOWn** low.

For a Read operation.

1. Write out the register value.
2. Delay as follows, based on the PIO mode.
   PIO Mode 0 - Delay for 70 ns
   PIO Mode 1 - Delay for 50 ns.
   PIO Mode 2 - Delay for 30 ns.
   PIO Mode 3 - Delay for 30 ns
   PIO Mode 4 - Delay for 25 ns
3. Bring **DIORn** high.
4. Based on the PIO mode, delay as follows before the next read or write can occur.
   PIO Mode 0 - Delay for 290 ns.
   PIO Mode 1 - Delay for 290 ns
   PIO Mode 2 - Delay for 290 ns.
   PIO Mode 3 - Delay for 80 ns
   PIO Mode 4 - Delay for 70 ns
5. Bring **DIORn** low.
6. Read IDE Data in the register.

Copyright 2007 Cirrus Logic

For a Write Operation.

1. Write out the register value.
2. Delay as follows, based on the PIO mode.
   PIO Mode 0 - Delay for 70 ns.
   PIO Mode 1 - Delay for 50 ns.
   PIO Mode 2 - Delay for 30 ns
   PIO Mode 3 - Delay for 30 ns
   PIO Mode 4 - Delay for 25 ns
3. Bring **DIOWn** high.
4. Delay as follows, based on the PIO mode.
   PIO Mode 0 - Delay for 290 ns.
   PIO Mode 1 - Delay for 290 ns
   PIO Mode 2 - Delay for 290 ns
   PIO Mode 3 - Delay for 80 ns
   PIO Mode 4 - Delay for 70 ns
5. Bring **DIOWn** low.
6. Delay as follows, based on the PIO mode before the next read or write can occur.
   PIO Mode 0 - 240 ns.
   PIO Mode 1 - 50 ns
   PIO Mode 2 - 20 ns
   PIO Mode 3 - 70 ns
   PIO Mode 4 - 25 ns

Minimum total cycle time for the various PIO modes is as follows:
PIO Mode 0 - 600 ns
PIO Mode 1 - 383 ns
PIO Mode 2 - 330 ns
PIO Mode 3 - 180 ns
PIO Mode 4 - 120 ns

You must also setup IDECFG and WST as follows, according to the PIO mode:
PIO Mode 0 - Delay for 30 ns
PIO Mode 1 - Delay for 20 ns
PIO Mode 2 - Delay for 15 ns
PIO Mode 3 - Delay for 10 ns
PIO Mode 4 - Delay for 5 ns

27.2.3 MDMA Operations

For MDMA operations, DMA commands are set up using PIO operations by the host. The registers IDEMDMADataOut and IDEMDMADataIn act as the 1-deep buffer for write and read operations respectively. The state machine sets up the necessary signals including the DMA request to the DMA controller.

Copyright 2007 Cirrus Logic

In a write operation, when the DMA controller writes to IDEMDMADataOut for completing the DMA transfer, the state machine toggles **DIOWn** and drives the data onto the DD bus. In a read operation, when data is filled into IDEMDMADataIn by the host latching in the DD bus at the **DIORn** rising edge, the state machine sends the DMA request. The DMA transfer is completed when the IDEMDMADataIn register is read by the DMA controller. These two registers should only be written or read by the DMA controller.

The registers IDEDataOut and IDEDataIn are aliased to IDEMDMADataOut and IDEMDMADataIn during MDMA operations. The host can read IDEDataOut and IDEDataIn registers at any time. All data transfers are 32-bit wide, with 2 16-bit wide data transfers to or from the DD bus executed before the next DMA request is sent.

27.2.4 UDMA Operations

For UDMA operations, DMA commands are set up using PIO operations by the host. There is a 32-bit, 12-deep output write buffer and a 32-bit, 12-deep input read buffer. These buffers are circular buffers with head and tail pointers. The state machine set up the necessary signals including the DMA request to the DMA controller.

In a write operation, when the write buffer has less than 4 entries, a DMA request will be sent to fill 4 32-bit entries in the buffer. At the same time, the state machine does the handshaking with the device and sends out the data in 16-bit pieces. Flow control is achieved by the host through controlling when to toggle **HSTROBE** and by the device through temporarily deasserting **DDMARDYn**.

In a read operation, the state machine does the handshaking and starts to receive data from the device. When the read buffer has 4 or more entries filled, a DMA request is sent to the DMA controller. Flow control is achieved by the host through temporarily deasserting **HDMARDYn** or by the device through controlling when to toggle **DSTROBE**.

In both write and read, either the host or the device can terminate the transfer and the state machine handles the termination handshaking mechanism.

A 16-bit CRC result is always sent from the host to device in both write and read operations for checking. The CRC registers are "seeded" or pre-loaded with the value of 0x4ABA at the beginning of the transfer. The "ping-pong" method is used and a "grace" area is provided in the buffers in case the handshaking required for pausing comes more slowly than the data. All data transfers are in chunks of four 32-bit words. Pieces of 16-bit wide data to or from the DD bus are consumed or collected. All data to be transferred through the DMA controller must be on word boundaries. In case the last chunk contains less than 4 32-bit words, a non-4-word transfer is allowed.

27.2.5 Performance Considerations

IDE data transfer performance depends on many factors. All PIO operations are expected to complete at the normal speed of the IDE interface when configured for the fastest PIO mode. Payload data transfers will normally use one of the DMA modes. For host read operations the DMA controller will try to keep the input read buffer empty. For host write operations the DMA controller will try to keep the output write buffer at least half full. If the DMA Transfer State

Copyright 2007 Cirrus Logic

Machine sees that the incoming versus outgoing data rate is out of balance, it will signal the controlling device to pause the transfer. For both read and write operations it is expected that the DMA controller will get behind and not be able to keep up with the device transfer rate. Thus the net transfer rate is determined by the available DMA controller bandwidth and how fast the DMA completion is acknowledged by interrupt or by reading some DMA transfer counters. The DMA controller does a DMA data transfer by:

- Requesting the AHB
- Reading the source data to a local buffer
- Requests the AHB for the write to the destination.

No burst transfer is allowed.

27.2.6 UDMA Example

For an estimation of the speed of operation, consider a UDMA read operation. After being granted master status on the AHB bus, the DMA controller access to the IDEUDMADataIn register will complete in 2 AHB clock cycles. Typical data transfers are to system dynamic memory through the SDRAM controller. Single cycle writes to SDRAM get posted and are completed in 2 AHB cycles, provided that the SDRAM controller is not busy. Consecutive SDRAM single transfer writes will take 8 AHB cycles. If we assume one clock cycle for bus arbitration, we end up with a maximum sustained DMA transfer rate of one 32-bit transfer every 11 AHB clocks. For a 100 MHz AHB (10 ns cycle time), and two 16-bit IDE transfers per DMA transfer, this example works out to 55 ns per IDE transfer peak rate.

A more typical DMA exists when the DMA request conflicts with ARM Core cache line fills or raster display memory access. Cache line fills use quad word bursts and raster accesses use 16 word bursts. The worst case is the raster, which will hold the SDRAM controller for 20 AHB clocks. Assuming a worst case system load where raster is getting 50% of the memory bandwidth, and each raster burst in between is a cache line fill, the DMA controller can only get 12 of the 40 available clocks. In this case, the DMA write would get posted, and flushed, but a read would use 8 of the 12 cycles. Either case would be able to complete one DMA transfer every 40 AHB clocks. The IDE transfer rate for this example is 400 ns per DMA transfer, which is 200 ns per IDE transfer. This still nets 10 megabytes per second (MBps) in a heavily loaded system.

In this last example, the DMA controller would not keep up with the IDE device and the transfers would rely on proper signaling to pause the IDE transfers until the DMA catches up. An additional overhead would be how fast the DMA controller is configured to do another DMA transfer after finishing one. There might be the possibility that the request line has been pulled high even before the DMA controller is set to service this request after the completion of the previous request is acknowledged.

The device operates asynchronously to the host and all input signals to the host are synchronized to the AHB clock. In a UDMA read operation, there is a possibility that the device is transmitting the data and toggling **DSTROBE** too quickly for the host to keep up with latching the data from DD bus, based on the synchronized version of **DSTROBE**. There is a lower limit for AHB clock speed, where lowering the speed further cannot guarantee correct

Copyright 2007 Cirrus Logic

latching of the data. It is calculated that the cycle time of AHB clock has to be smaller than (IDE cycle time)*2/3. For different UDMA speed modes, the minimum AHB clock speeds are listed below. There is no special speed constraint imposed on the design for PIO and MDMA modes.

Table 27-2. IDE Cycle Times and Data Transfer Rates

UDMA Speed Mode	Min. IDE Cycle Time	Max. AHB Cycle Time	Min. AHB Clock Frequency
0	112 ns	74.7 ns	13.4 MHz
1	73 ns	48.7 ns	20.5 MHz
2	54 ns	36.0 ns	27.8 MHz
3	39 ns	26.0 ns	38.5 MHz
4	25 ns	16.7 ns	59.8 MHz

27.2.7 DMA Request Latency

27.2.7.1 DMA Request Deassertion

Multi-word DMA Write to IDE Controller:

The **DMAide** signal deassertion is generated based on the AHB write logic. The act of writing to the Multi-word DMA write-FIFO causes the deassertion to appear on the following bus cycle.

Multi-word DMA Read from IDE Controller:

The **DMAide** signal deassertion is generated based on the AHB read logic. The act of reading from the Multi-word DMA read-FIFO causes the deassertion to appear on the following bus cycle.

Ultra DMA Write to IDE Controller:

The **DMAide** signal deassertion is generated based on the contents of the Ultra DMA write FIFO. If the FIFO contains four or more elements, the DMAide signal deasserts.

Ultra DMA Read from IDE Controller:

The **DMAide** signal deassertion is generated based on an internal counter. The **DMAide** signal will deassert if four DMA reads have occurred or if the FIFO is now empty (which only occurs at the end of a non-quad word aligned read from the IDE device)

27.2.7.2 DMA Request Latency Overview

The IDE controller requires a certain number of cycles to deassert the DMA request line **DMAide** after a DMA access for Multi-word DMA and Ultra DMA modes. The number of wait-states required are listed below in addition to the pipeline breakdown of the signal propagation. The assumption is that the deassertion should follow an AHB bus command (read or write) in HCLK cycle 1.

Copyright 2007 Cirrus Logic

Table 27-3. Wait State Value for the DMA M2M Register Control.PWSC

Wait States:	Multi-Word DMA Request	Ultra DMA Request
Read	0	1
Write	3	2

Note: This is the number of wait states required by the IDE Controller to deassert the DMA Controller request line after each word transfer is complete.

Table 27-4. HCLK Cycles to De-assert DMA Request

Operation	HCLK Cycle	Event
Multi-word DMA Write to IDE Controller:	0	AHB write command
	1	DMAide deasserts
Multi-word DMA Read from IDE Controller:	0	AHB read command
	1	AHB read data, DMAide deasserts
Ultra DMA Write to IDE Controller:	0	AHB write command
	1	Data stored in IDE register
	2	Data stored in FIFO, FIFO status updates
	3	DMAide deasserts
Ultra DMA Read from IDE Controller:	0	AHB read command
	1	AHB read data, word counter updates
	2	DMAide deasserts

27.2.7.3 IDE DMA Programming Considerations

This is a general guideline for programming the DMA controller to properly interact with the IDE controller without receiving malformed data. All cases assume the DMA controller is capable of burst read or burst write operations. Non-ideal DMA controllers may be able to avoid wait-states due to less than optimal bus utilization.

27.2.7.3.1 General Note

Please verify that the DMA controller will ignore DMA requests if it's transfer counter register has gone to zero. If this is not the case, the DMA controller must be configured to time out based on the wait-state table in Table 27-3 and Table 27-4. Quad-word bursts are not allowed.

27.2.7.3.2 Multi-word DMA

Follow the wait-state number listed in the wait-state table in Table 27-3 and Table 27-4. Quad-word bursts are not allowed.

Copyright 2007 Cirrus Logic

27.2.7.3.3 Ultra DMA Read from IDE Controller

Follow the wait-state number listed in the wait-state table in Table 27-3 and Table 27-4. However, the DMA request will not assert unless there are 4 words present in the read FIFO or the transfer is non-quad aligned and has the last remaining bits of data, so quad-word bursts are permissible if the total Ultra DMA transfer size is quad-word aligned.

27.2.7.3.4 Ultra DMA Write to IDE Controller

Although the DMA request line has a non-insignificant DMAIDE latency, the DMA write FIFO is of sufficient size to absorb any overage incurred during the DMA request latency period. The DMA controller can be run without wait-states. Quad-word bursts are permissible if the Ultra DMA transfer size is quad-word aligned.

27.2.8 IDE Package Dependency

The block uses the following external pins:

IDECS0n, IDECS1n, IDEDA, DIORn, DIOWn, DMACKn, DD, IORDY, INT[3], EGPIO[2], and **EGPIO[15].**

27.2.8.1 System Configuration Constraints

The following system configuration modes force the disabling of the IDE controller:

- GPIOEonIDE
- GPIOFonIDE
- GPIOGonIDE
- GPIOHonIDE

27.2.8.2 Bus Bandwidth Requirements

The block does not have any hard bandwidth constraints because it can throttle performance to the available bandwidth without data corruption. The maximum free bandwidth that the block will consume is limited by the IDE mode the device is in. Maximum theoretical bandwidths are listed in Table 27-5.

Table 27-5. Maximum Theoretical Bandwidths for Various Operating Modes

Mode	MAX IDE Device Bandwidth
PIO Mode 0	3.33 MBps
PIO Mode 1	5.22 MBps
PIO Mode 2	8.33 MBps
PIO Mode 3	11.11 MBps
PIO Mode 4	16.67 MBps
MDMA Mode 0	4.17 MBps
MDMA Mode 1	13.33 MBps
MDMA Mode 2	16.67 MBps

Copyright 2007 Cirrus Logic

For both PIO and MDMA modes, the actual throughput is limited by the ARM Core's ability to service requests, not raw bandwidth. For UDMA, the throughput is dependent on the bandwidth available to the DMA controller.

27.3 Registers

Table 27-6. IDE Interface Register Map

Address	Name	Description
0x800A_0000	IDECtrl	IDE Control Register
0x800A_0004	IDECfg	IDE Configuration Register
0x800A_0008	IDEMDMAOp	IDE MDMA Operation Register
0x800A_000C	IDEUDMAOp	IDE UDMA Operation Register
0x800A_0010	IDEDataOut	IDE PIO Data Output Register
0x800A_0014	IDEDataIn	IDE PIO Data Input Register
0x800A_0018	IDEMDMADataOut	IDE MDMA Data Output Register
0x800A_001C	IDEMDMADataIn	IDE MDMA Data Input Register
0x800A_0020	IDEUDMADataOut	IDE UDMA Data Output Register
0x800A_0024	IDEUDMADataIn	IDE UDMA Data Input Register
0x800A_0028	IDEUDMASts	IDE UDMA Status Register
0x800A_002C	IDEUDMADebug	IDE UDMA Debug Register
0x800A_0030	IDEUDMAWrBufSts	IDE UDMA Write Buffer Status Register
0x800A_0034	IDEUDMARdBufSts	IDE UDMA Read Buffer Status Register

Register Descriptions

IDECtrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD					IORDY	INTRQ	DMARQ	DASPn	DIOWn	DIORn	DA			CS1n	CS0n

Address:

0x800A_0000 - Read/Write

Default:

0x0000_0063

Definition:

IDE Control Register. This register is used for IDE PIO control operations. **IORDY**, **INTRQ**, **DMARQ**, and **DASPn** reflect external pins. Their reset state can vary depending on system implementation and system configuration.

Copyright 2007 Cirrus Logic

Bit Descriptions:

RSVD:	Reserved. Unknown during read, ignored during write.
CS0n:	Chip Select 0 pin output control.
CS1n:	Chip Select 1 pin output control.
DA:	Device address output control.
DIORn:	**DIORn** pin output control.
DIOWn:	**DIOWn** pin output control.
DASPn:	**DASPn** pin input state. This signal comes in on the **EGPIO[15]** pin. Read only.
DMARQ:	**DMARQ** pin input state. This signal comes in on the **EGPIO[2]** pin. Read only.
INTRQ:	**INTRQ** pin input state. This input comes from the **INT[3]** input pin and is routed to the Interrupt Controller. Read only.
IORDY:	IORDY pin input state. Read only.

IDECfg

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD						WST		MODE				UDMA	MDMA	PIO	IDEEN

Address:
0x800A_0004 - Read/Write

Default:
0x0000_0000

Definition:
IDE Configuration Register.

Bit Descriptions:

RSVD:	Reserved. Unknown during read, ignored during write.
IDEEN:	IDE master enable.
PIO:	Polled IO operation selection.
MDMA:	Multiword DMA operation selection.
UDMA:	Ultra DMA operation selection.

Copyright 2007 Cirrus Logic

Note: At most, one of the above 3 bits should be set to 1 at any time. If more than one is set, the results will be unpredictable, and the data invalid.

MODE: Speed mode number. (0 to 4 defined for PIO, 0 to 2 defined for MDMA, 0 to 4 defined for UDMA).

WST: Wait State for Turn. Number of HCLK cycles to hold the data bus after a PIO write operation.

27

IDEMDMAOp

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD														RWOP	MEN

Address:

0x800A_0008 Read/Write

Default:

0x0000_0000

Definition:

IDE MDMA Configuration Register.

Bit Descriptions:

RSVD: Reserved. Unknown during read, ignored during write.

MEN: Enable Multiword DMA operation.
1 - to start MDMA
0 - to terminate MDMA by the host.

RWOP: Read or write operation selection:
0 - Read
1 - Write.

IDEUDMAOp

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD														RWOP	UEN

Address:

0x800A_000C - Read/Write

Copyright 2007 Cirrus Logic

Default:

0x0000_0000

Definition:

IDE UDMA Configuration Register.

Bit Descriptions:

RSVD:	Reserved. Unknown during read, ignored during write.
RWOP:	Read or write operation selection: 0 - Read 1 - Write.
UEN:	Enable Ultra DMA operation. 1 - to start UDMA 0 - to terminate UDMA by the host.

Note: Before setting the UEN bit to enable UDMA operation:
1 - Set or Clear the RWOP bit to configure for a Write or Read operation.
2 - Perform a dummy read of the IDEUDMAOp register.
3 - Set the UEN bit to enable UDMA operation.

IDEDataOut

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDEDD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDEDD															

Address:

0x800A_0010 - Read/Write

Default:

0x0000_0000

Definition:

In PIO mode write operation, this register is the Output Data Registers, containing the register contents or the data to be written to the device. The register is driven onto the **DD** pins when **DIOWn** is low. The register is both read write in this operation. In MDMA and UDMA data-out operations, this register is an exact copy of the data in the output buffer to be transferred to the device. The register should only be read in these operations for checking the output data. Any write in these two operation modes is ignored.

Bit Descriptions:

IDEDD:	IDE output data in PIO writes (read write), data in output buffer in MDMA and data at the tail of output buffer in UDMA mode (read only).

Copyright 2007 Cirrus Logic

IDEDataIn

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDEDD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDEDD															

27

Address:

0x800A_0014 - Read Only

Default:

0x0000_0000

Definition:

In PIO mode read operation, this register is the Input Data Registers, containing the register contents or the data read from the device. The register is loaded from the **DD** pins at the positive edge of the **DIORn** signal. The register is read only in this operation. In MDMA and UDMA data-in operations, this register is an exact copy of the data in the input buffer to be transferred through DMA. The register is read only in these operations. Any write to this register is ignored.

Bit Descriptions:

IDEDD: IDE input data in PIO read, data in input buffer in MDMA and data at the head of input buffer in UDMA mode.

IDEMDMADataOut

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDEDD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDEDD															

Address:

0x800A_0018 - Write Only (should be written by the DMA controller only)

Default:

0x0000_0000

Definition:

In MDMA data-out operations, this register contains the data in the output buffer to be transferred to the device. The data is written into this register by the DMA controller. This register should only be addressed and written by the

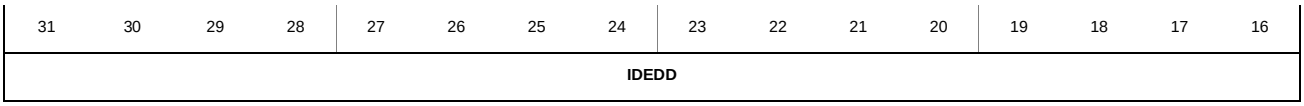
Copyright 2007 Cirrus Logic

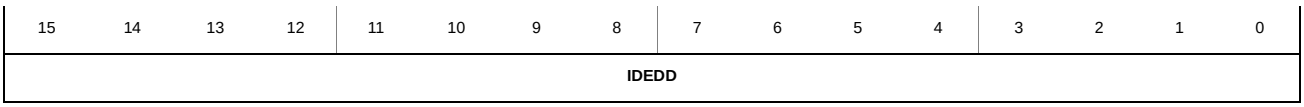
DMA controller. A write by the host during MDMA data-out operation will erroneously interfere with the MDMA state machine. Any read will return zero.

Bit Descriptions:

IDEDD: IDE output data in the output buffer in MDMA mode.

IDEMDMADataIn





Address:

0x800A_001C - Read Only (should be read by the DMA controller only)

Default:

0x0000_0000

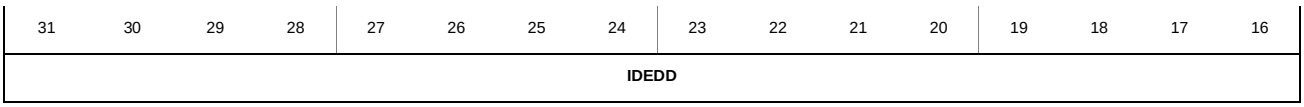
Definition:

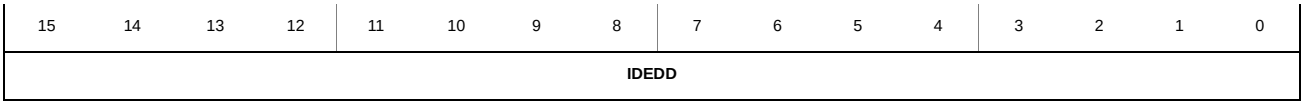
In MDMA data-in operations, this register contains the data in the input buffer just transferred from the device. The data is read from this register by the DMA controller. This register should only be addressed and read by the DMA controller. A read by the host during MDMA data-in operation will erroneously interfere with the MDMA state machine. Any write is ignored.

Bit Descriptions:

IDEDD: IDE input data in the input buffer in MDMA mode.

IDEUDMADataOut





Address:

0x800A_0020 - Write Only (should be written by the DMA controller only)

Default:

0x0000_0000

Definition:

In UDMA data-out operations, this register contains the data at the tail of the output buffer to be written by the DMA controller. This register should only be

Copyright 2007 Cirrus Logic

addressed and written by the DMA controller. A write by the host during UDMA data-out operation will erroneously interfere with the UDMA state machine. Any read will return zero.

Bit Descriptions:

IDEDD: IDE output data at the tail of the output buffer in UDMA mode.

27

IDEUDMADataIn

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDEDD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDEDD															

Address:

0x800A_0024 - Read Only (should be read by the DMA controller only)

Default:

0x0000_0000

Definition:

In UDMA data-in operations, this register contains the data at the head of the input buffer to be transferred by the DMA controller. The data is read from this register by the DMA controller. This register should only be addressed and read by the DMA controller. A read by the host during UDMA data-in operation will erroneously interfere with the UDMA state machine. Any write is ignored.

Bit Descriptions:

IDEDD: IDE input data at the head of the input buffer in UDMA mode.

IDEUDMASts

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD					N4X	NDI	NDO	RSVD					SBUSY	INTide	DMAide

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD					DSDD	DMARQ	DDOE	DM	STOP	HSHD	DA			CS1n	CS0n

Address:

0x800A_0028 - Read Only

Default:

0x0000_0000

Copyright 2007 Cirrus Logic

Definition:

In UDMA data-out and data-in operations, this register contains status about the output and input signals, state machine status and error reporting. Several bits reflect external pins. Their reset state can vary depending on system implementation and system configuration.

Bit Descriptions:

RSVD:	Reserved. Unknown during read, ignored during write.
CS0n:	Chip select pin0 status. Should be driven to 1 (deasserted) in UDMA.
CS1n:	Chip select pin1 status. Should be driven to 1 (deasserted) in UDMA.
DA:	Device address status. Should be driven to 0x0 (de-asserted) in UDMA.
HSHD:	**HSTROBE** (during data-out) and **HDMARDYn** (during data-in) status. Driven by UDMA state machine.
STOP:	**STOP** (during data-out) status. Driven by UDMA state machine.
DM:	**DMACKn** (both data-out and data-in) status. Driven by UDMA state machine.
DDOE:	**DD** bus output enable as controlled by UDMA state machine.
DMARQ:	Synchronized version of **DMARQ** input from device.
DSDD:	**DSTROBE** (during data-in) and **DDMARDYn** (during data-out) status from device.
DMAide:	DMA request signal from UDMA state machine.
INTide:	**INT** line generated by UDMA because of errors in the state machine.
SBUSY:	UDMA state machine busy, not in idle state.
NDO:	Error for data-out not completed.
NDI:	Error for data-in not completed.
N4X:	Error for data transferred not multiples of four 32-bit words.

Copyright 2007 Cirrus Logic

IDEUDMADebug

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD										RRDR	RRPTR	RROE	RWDR	RWPTR	RWOE

Address:

0x800A_002C - Read/Write

Default:

0x0000_0000

Definition:

Debug register to reset some internal signals in the UDMA state machine for debug purpose.

Bit Descriptions:

RSVD: Reserved. Unknown during read, ignored during writes.

RWOE: Reset UDMA write data-out error.

RWPTR: Reset UDMA write buffer pointer to 0.

RWDR: Reset UDMA write DMA request.

RROE: Reset UDMA read data-in error.

RRPTR: Reset UDMA read buffer pointer to 0.

RRDR: Reset UDMA read DMA request.

IDEUDMAWrBufSts

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				FULL	NFULL	HOM	EMPTY	TPTR				HPTR			

Address:

0x800A_0030 - Read Only

Default:

0x0000_0100

Definition:

Status register for UDMA write buffer.

Copyright 2007 Cirrus Logic

Bit Descriptions:

RSVD:	Reserved. Unknown during read, ignored during writes.
HPTR:	Head pointer in the write buffer.
TPTR:	Tail pointer in the write buffer.
EMPTY:	Write buffer empty status.
HOM:	Half or more entries in write buffer filled status.
NFULL:	Write buffer near full status.
FULL:	Write buffer full status.
CRC:	CRC result for data-out operation. Reset to 0x4ABA after finishing UDMA operation.

IDEUDMARdBufSts

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				FULL	NFULL	HOM	EMPTY	TPTR				HPTR			

Address:

0x800A_0034 - Read Only

Default:

0000_0100

Definition:

Status register for UDMA read buffer.

Bit Descriptions:

RSVD:	Reserved. Unknown during read, ignored during writes.
HPTR:	Head pointer in the read buffer.
TPTR:	Tail pointer in the read buffer.
EMPTY:	Read buffer empty status.
HOM:	Half or more entries in read buffer filled status.
NFULL:	Read buffer near full status.
FULL:	Read buffer full status.
CRC:	CRC result for data-in operation. Reset back to 0x4ABA after finishing UDMA operation.

Copyright 2007 Cirrus Logic

27

Copyright 2007 Cirrus Logic

Chapter 28

GPIO Interface

28

28.1 Introduction

Note: The EP9301 and EP9302 processors each have 18 standard GPIOs and 19 enhanced GPIOs.

Note: The EP9307 processor has 30 standard GPIOs and 18 enhanced GPIOs.

Note: The EP9312 processor has 31 standard GPIOs and 16 enhanced GPIOs.

Note: The EP9315 processor has 31 standard GPIOs and 24 enhanced GPIOs.

The General Purpose Input/Output (GPIO) is an Advanced Peripheral Bus (APB) slave module. The GPIO block is the primary controller for the **EGPIO, RDLED, GRLED, SLA[1:0], EECLK,** and **EEDAT** pins. It is a secondary controller for the **ROW[7:0], COL[7:0]**, PCMCIA and **IDE** control pins.

There are two types of GPIOs, standard and enhanced. The enhanced GPIO, called EGPIO, have interrupt generation capability.

The GPIO block has eight ports, named Port A through Port H. Ports C, D, E, G, and H are standard GPIO ports. Ports A, B, and F are enhanced GPIO ports.

Each GPIO port controls eight individual pins. Each port has an 8-bit data register and an 8-bit data direction register. The EGPIO ports each have additional 8-bit registers for interrupt configuration and status. The control of an individual pin is determined in a bit-slice fashion across all registers for that port; only a single bit at a particular index from each register affects or is affected by that pin.

Copyright 2007 Cirrus Logic

28

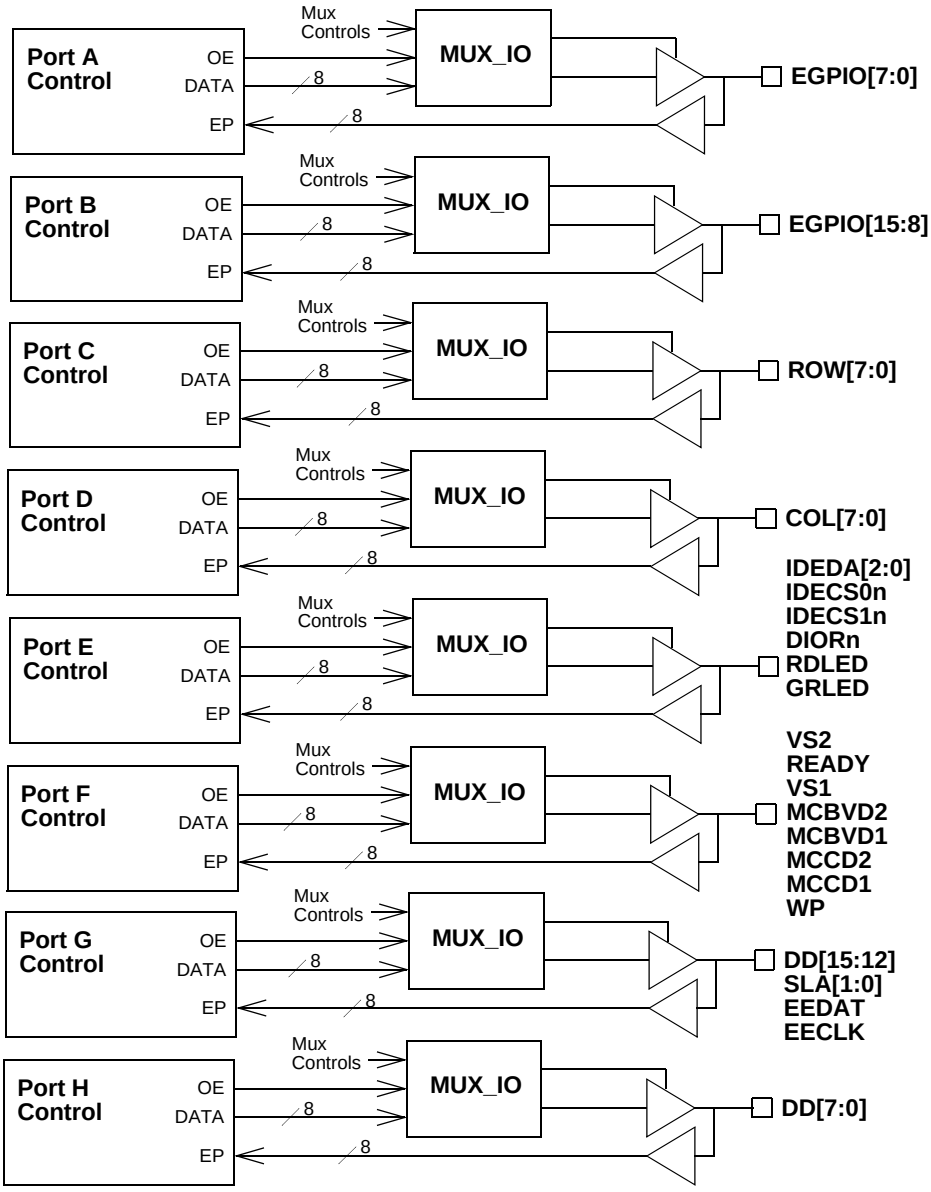


Figure 28-1. System Level GPIO Connectivity

Copyright 2007 Cirrus Logic

28.1.1 Memory Map

The GPIO base address is 0x8084_0000. All registers are 8 bits wide and are aligned on word boundaries. For all registers, the upper 24 bits are not modified when written and always read zeros.

28.1.2 Functional Description

Each port has an 8-bit data register and an 8-bit direction register. The data direction register controls whether each individual GPIO pin is an input or output. Writing to a data register only affects the pins that are configured as outputs. Reading a data register returns the value on the corresponding GPIO pins.

Ports A, B, and F also provide interrupt capability. The 16 interrupt sources from Ports A and B are combined into a single signal **GPIOINTR** which is connected to the system interrupt controller. All eight individual interrupt signals on Port F are available to the system interrupt controller as **GPIO0INTR** through **GPIO7INTR**.

The interrupt properties of each of the GPIO pins on ports A, B, and F are individually configurable. Each interrupt can be either high or low level sensitive or either positive or negative edge triggered. It is also possible to enable debouncing on the Port A, B, and F interrupts. Debouncing is implemented using a 2-bit shift register clocked by a 128 Hz clock.

There are seven additional registers for port A, B, and F:

- *GPIO Interrupt Enable* registers (GPIOAIntEn, GPIOBIntEn, GPIOFIntEn) control which bits are to be configured as interrupts. Setting a bit in this register configures the corresponding pin as an interrupt input.
- *GPIO Interrupt Type 1* registers (GPIOAIntType1, GPIOBIntType1, GPIOFIntType1) determines interrupt type. Setting a bit in this register configures the corresponding interrupt as edge sensitive; clearing it makes it level sensitive.
- *GPIO Interrupt Type 2* registers (GPIOAIntType2, GPIOBIntType2, GPIOFIntType2) determines interrupt polarity. Setting a bit in this register configures the corresponding interrupt as rising edge or high level sensitive; clearing it configures the interrupt as falling edge or low level sensitive.
- *GPIO End-Of-Interrupt* registers (GPIOAEOI, GPIOBEOI, GPIOFEOI) are used to clear specific bits in the interrupt Status Register. Writing a one to a bit will clear the corresponding interrupt; writing a zero has no effect.
- *GPIO Debounce* registers (GPIOADB, GPIOBDB, GPIOFDB) enable debouncing of specific interrupts signals.
- *Interrupt Status* registers (IntStsA, IntStsB, IntStsF) provide the status of any pending unmasked interrupt.
- *Raw Interrupt Status* registers (RawIntStsA, RawIntStsB, RawIntStsF) provide the status of any pending interrupt regardless of masking.

Copyright 2007 Cirrus Logic

In order to stop any spurious interrupts that may occur during the programming of the GPIOxINTTYPEx registers, the following sequence should be observed:

1. Disable interrupt by writing to GPIO Interrupt Enable register.
2. Set interrupt type by writing GPIOxINTTYPE1/2 register.
3. Clear interrupt by writing to GPIOxEOI register.
4. Enable interrupt by writing to GPIO Interrupt Enable register.

28

Figure 28-2 and Figure 28-3 illustrate the signal connections for GPIO and EGPIO.

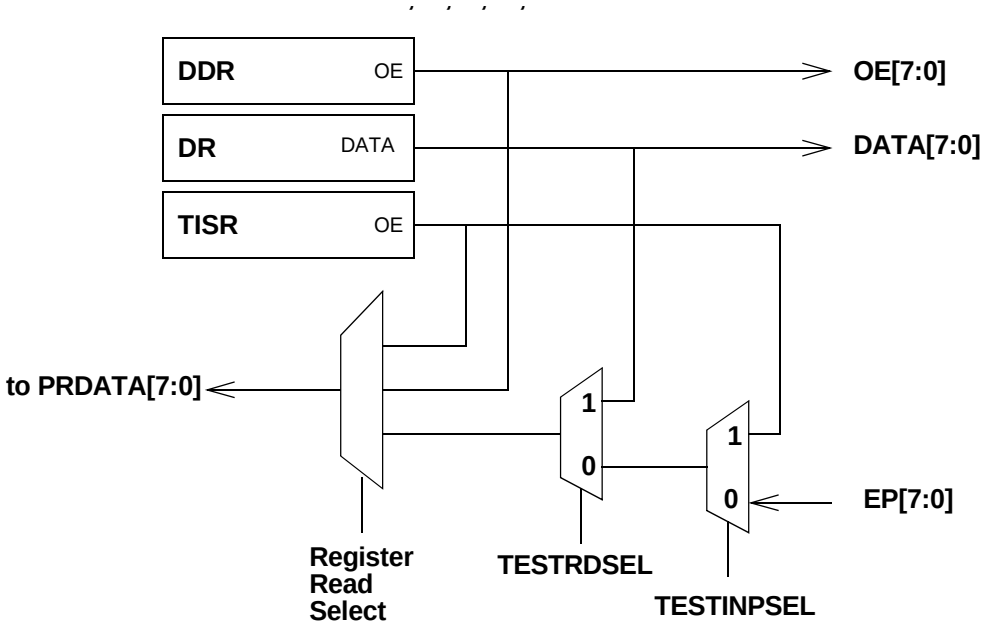


Figure 28-2. Signal Connections Within the Standard GPIO Port Control Logic (Ports C, D, E, G, H)

Copyright 2007 Cirrus Logic

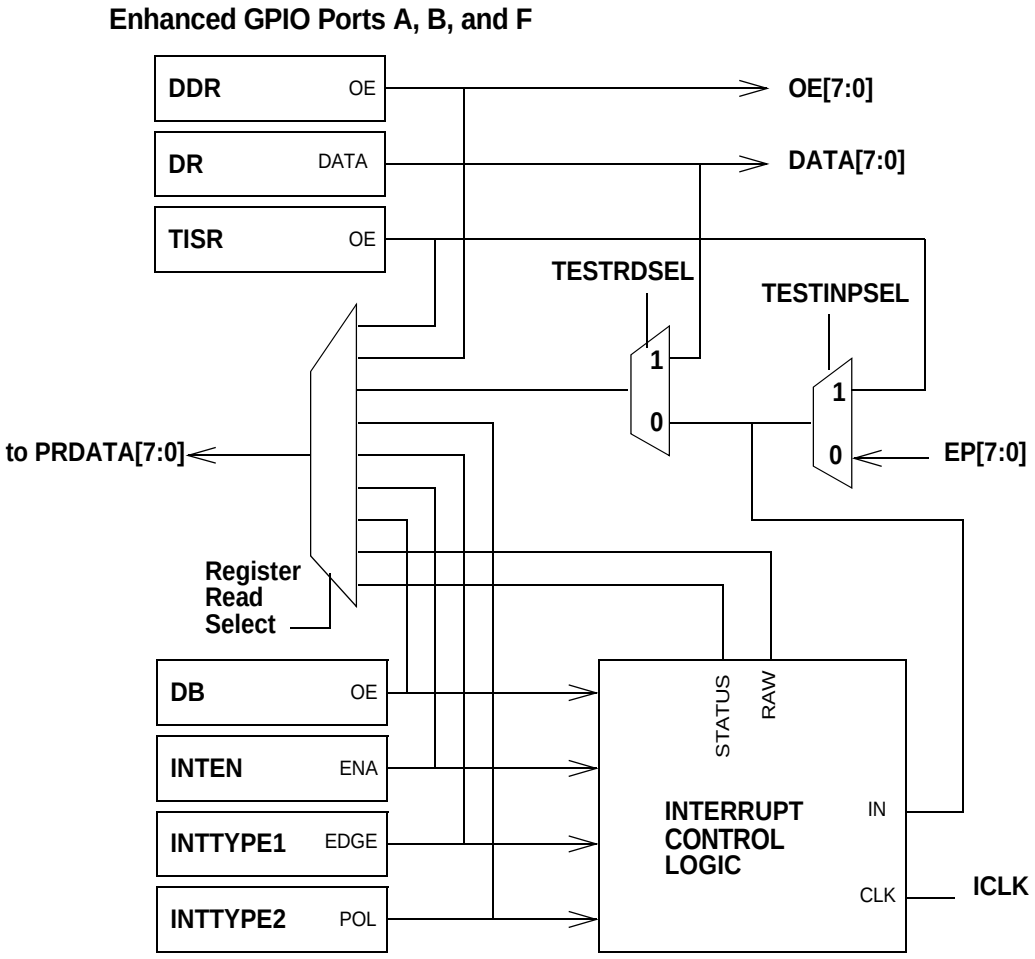


Figure 28-3. Signal Connections Within the Enhanced GPIO Port Control Logic (Ports A, B, F)

28.1.3 Reset

All GPIO registers are initialized on system reset. The data and data direction registers for all ports (except as noted below) are cleared, configuring them as inputs. Port E[1:0] bits are used for the LED outputs **RDLED** and **GRLED** respectively and are set to drive high. Port G[3:2] bits are used for **SLA[1:0]** outputs and are set to drive low. Port G[1:0] bits are used for **EEDAT** and **EECLK** respectively and are set up as inputs. All interrupt control and debounce registers are cleared.

Copyright 2007 Cirrus Logic

28.1.4 GPIO Pin Map

All GPIO signals are mapped to device pins. The Syscon DeviceCfg register contains four bits that control mapping of the GPIO Ports to device pins: GonK, EonIDE, GonIDE, and HonIDE. Table 28-1, Table 28-2, Table 28-3, and Table 28-4 show how the GPIO ports map to EP93xx pins depending on these control signals.

28

Table 28-1. EP9301 and EP9302 GPIO Port to Pin Map

Pin Name	Default Function
EGPIO[7:0]	Port A
EGPIO[15:8]	Port B
GRLED[1]	Port E0
RDLED[2]	Port E1
EECLK[3]	Port G0
EEDAT[4]	Port G1

1. GRLED is the Green LED pin.
2. RDLED is the Red LED pin.
3. EECLK is the EEPROM clock pin.
4. EEDAT is the EEPROM data pin.

Table 28-2. EP9307 GPIO Port to Pin Map

Pin Name	Default Function	Function in GonK Mode
EGPIO[7:0]	Port A	Port A
EGPIO[13:8]	Port B	Port B
EGPIO[15]	Port B	Port B
GRLED[1]	Port E0	Port E0
RDLED[2]	Port E1	Port E1
EECLK[3]	Port G0	Port G0
EEDAT[4]	Port G1	Port G1
ROW[7:0][5]	ROW[7:0]	Port C
COL[7:0]6	COL[7:0]	Port D

1. GRLED is the Green LED pin.
2. RDLED is the Red LED pin.
3. EECLK is the EEPROM clock pin.

Copyright 2007 Cirrus Logic

4. EEDAT is the EEPROM data pin.
5. ROW[7:0] are the Key Matrix row pins.
6. COL[7:0] are the Key Matrix column pins.

Table 28-3. EP9312 GPIO Port to Pin Map

Pin Name	Default Function	Function in GonK Mode	Function in EonIDE Mode	Function in GonIDE Mode	Function in HonIDE Mode
EGPIO[7:0]	Port A	Port A	Port A	Port A	Port A
EGPIO[15:8]	Port B	Port B	Port B	Port B	Port B
GRLED[3]	Port E0	Port E0	Port E0	Port E0	Port E0
RDLED[4]	Port E1	Port E1	Port E1	Port E1	Port E1
EECLK[5]	Port G0	Port G0	Port G0	Port G0	Port G0
EEDAT[6]	Port G1	Port G1	Port G1	Port G1	Port G1
SLA[1:0]	Port G3:2	Port G3:2	Port G3:2	Port G3:2	Port G3:2
ROW[7:0][7]	ROW[7:0]	Port C	ROW[7:0]	ROW[7:0]	ROW[7:0]
COL[7:0][8]	COL[7:0]	Port D	COL[7:0]	COL[7:0]	COL[7:0]
IDEDA[2:0][1]	IDEDA[2:0]	IDEDA[2:0]	Port E7:5	IDEDA[2:0]	IDEDA[2:0]
IDECS1n[1]	IDECS1n	IDECS1n	Port E4	IDECS1n	IDECS1n
IDECS0n[1]	IDECS0n	IDECS0n	Port E3	IDECS0n	IDECS0n
DIORn[1]	DIORn	DIORn	Port E2	DIORn	DIORn
DD[15:12][2]	DD[15:12]	DD[15:12]	DD[15:12]	Port G7:4	DD[15:12]
DD[11:8][2]	DD[11:8]	DD[11:8]	DD[11:8]	DD[11:8]	DD[11:8]
DD[7:0][2]	DD[7:0]	DD[7:0]	DD[7:0]	DD[7:0]	Port H

1. IDEDA[2:0], IDECS0n, IDECS1n, and DIORn are IDE control pins.
2. DD[15:0] are the IDE data pins. DD[11:8] has no GPIO pin mapping.
3. GRLED is the Green LED pin.
4. RDLED is the Red LED pin.
5. EECLK is the EEPROM clock pin.
6. EEDAT is the EEPROM data pin.
7. ROW[7:0] are the Key Matrix row pins.
8. COL[7:0] are the Key Matrix column pins.

Copyright 2007 Cirrus Logic

28

Table 28-4. EP9315 GPIO Port to Pin Map

Pin Name	Default Function	Function in GonK Mode	Function in EonIDE Mode	Function in GonIDE Mode	Function in HonIDE Mode
EGPIO[7:0]	Port A	Port A	Port A	Port A	Port A
EGPIO[15:8]	Port B	Port B	Port B	Port B	Port B
GRLED[5]	Port E0	Port E0	Port E0	Port E0	Port E0
RDLED[6]	Port E1	Port E1	Port E1	Port E1	Port E1
VS2[2]	Port F7	Port F7	Port F7	Port F7	Port F7
READY[2]	Port F6	Port F6	Port F6	Port F6	Port F6
VS1[2]	Port F5	Port F5	Port F5	Port F5	Port F5
MCBVD[2:1][2]	Port F4:3	Port F4:3	Port F4:3	Port F4:3	Port F4:3
MCD[2:1][2]	Port F2:1	Port F2:1	Port F2:1	Port F2:1	Port F2:1
WP[2]	Port F0	Port F0	Port F0	Port F0	Port F0
EECLK[7]	Port G0	Port G0	Port G0	Port G0	Port G0
EEDAT[8]	Port G1	Port G1	Port G1	Port G1	Port G1
SLA[1:0][3]	Port G3:2	Port G3:2	Port G3:2	Port G3:2	Port G3:2
ROW[7:0][9]	ROW[7:0]	Port C	ROW[7:0]	ROW[7:0]	ROW[7:0]
COL[7:0][10]	COL[7:0]	Port D	COL[7:0]	COL[7:0]	COL[7:0]
IDEDA[2:0][1]	IDEDA[2:0]	IDEDA[2:0]	Port E7:5	IDEDA[2:0]	IDEDA[2:0]
IDECS1n[1]	IDECS1n	IDECS1n	Port E4	IDECS1n	IDECS1n
IDECS0n[1]	IDECS0n	IDECS0n	Port E3	IDECS0n	IDECS0n
DIORn[1]	DIORn	DIORn	Port E2	DIORn	DIORn
DD[15:12][4]	DD[15:12]	DD[15:12]	DD[15:12]	Port G7:4	DD[15:12]
DD[11:8][4]	DD[11:8	DD[11:8]	DD[11:8]	DD[11:8]	DD[11:8]
DD[7:0]	DD[7:0]	DD[7:0]	DD[7:0]	DD[7:0]	Port H

1. IDEDA[2:0], IDECS0n, IDECS1n, and DIORn are IDE control pins.
2. VS2, VS1, MCBVD[2:1], MCD[2;1], READY, and WP are PCMCIA pins.
3. SLA[1:0] are the PCMCIA voltage control pins.
4. DD[15:0] are the IDE data pins. DD[11:8] has no GPIO pin mapping.
5. GRLED is the Green LED pin.
6. RDLED is the Red LED pin.
7. EECLK is the EEPROM clock pin.
8. EEDAT is the EEPROM data pin.
9. ROW[7:0] are the Key Matrix row pins.

Copyright 2007 Cirrus Logic

10. COL[7:0] are the Key Matrix column pins.

Note: The various functional modes described in Table 28-4 are selected via bits set in the DeviceCfg register in Syscon. See Chapter 5, "DeviceCfg" on page 5-25 for additional register information.

When the GPIO port signals are not explicitly mapped to a device pin, the inputs will continue to monitor the pin while outputs are disconnected. For example, when the Key Matrix block has control of the ROW pins, GPIO port C inputs still monitor the state of the ROW pins.

Another level of functional muxing is applied to several EGPIO pins. The Syscon DeviceCfg register bits RonG, MonG, TonG, HC3EN, HC1EN, and map different functionality to the EGPIO pins:

- MonG maps **RI** (modem Ring Indicator) onto **EGPIO[0].**
- RonG maps **CLK32K**, the 32 KHz clock monitor output for RTC calibration, onto **EGPIO[1]**.
- TonG maps **TENn**, the RS485 transmit enable output, onto **EGPIO[3]**.
- Both HC3EN and HC1EN map the synchronous HDLC clock onto **EGPIO[3]**.

Some GPIO signals are used as inputs by other functional blocks. **EGPIO[2:1]** are routed to the DMA controller to allow for external DMA requests. IDE interface input signals **DMARQ** and **DASPn** are **EGPIO[2]** and **EGPIO[15]**, respectively.

28.2 Registers

Table 28-5. GPIO Register Address Map

Address	Read Location	Type	Write Location	Reset Value
0x8084_0000	PADR	R/W	PADR	Note 1
0x8084_0004	PBDR	R/W	PBDR	Note 1
0x8084_0008	PCDR	R/W	PCDR	Note 1
0x8084_000C	PDDR	R/W	PDDR	Note 1
0x8084_0010	PADDR	R/W	PADDR	0x00
0x8084_0014	PBDDR	R/W	PBDDR	0x00
0x8084_0018	PCDDR	R/W	PCDDR	0x00
0x8084_001C	PDDDR	R/W	PDDDR	0x00
0x8084_0020	PEDR	R/W	PEDR	Note 2
0x8084_0024	PEDDR	R/W	PEDDR	0x03
0x8084_0028	RSVD	-	RSVD	-
0x8084_002C	RSVD	-	RSVD	-
0x8084_0030	PFDR	R/W	PFDR	Note 1
0x8084_0034	PFDDR	R/W	PFDDR	0x00
0x8084_0038	PGDR	R/W	PGDR	Note 1
0x8084_003C	PGDDR	R/W	PGDDR	0x0C
0x8084_0040	PHDR	R/W	PHDR	Note 1
0x8084_0044	PHDDR	R/W	PHDDR	0x00
0x8084_0048	RSVD	-	RSVD	-
0x8084_004C	GPIOFIntType1	R/W	GPIOFIntType1	0x00

Copyright 2007 Cirrus Logic

28

Table 28-5. GPIO Register Address Map (Continued)

Address	Read Location	Type	Write Location	Reset Value
0x8084_0050	GPIOFIntType2	R/W	GPIOFIntType2	0x00
0x8084_0054	Reserved, Read undefined	Write Only	GPIOFEOI	0x00
0x8084_0058	GPIOFIntEn	R/W	GPIOFIntEn	0x00
0x8084_005C	IntStsF	Read only	-	0x00
0x8084_0060	RawIntStsF	Read only	-	Note 3
0x8084_0064	GPIOFDB	R/W	GPIOFDB	0x00
0x8084_0090	GPIOAIntType1	R/W	GPIOAIntType1	0x00
0x8084_0094	GPIOAIntType2	R/W	GPIOAIntType2	0x00
0x8084_0098	-	Write Only	GPIOAEOI	-
0x8084_009C	GPIOAIntEn	R/W	GPIOAIntEn	0x00
0x8084_00A0	IntStsA	Read only	-	0x00
0x8084_00A4	RawIntStsA	Read only	-	Note 3
0x8084_00A8	GPIOADB	R/W	GPIOADB	0x00
0x8084_00AC	GPIOBIntType1	R/W	GPIOBIntType1	0x00
0x8084_00B0	GPIOBIntType2	R/W	GPIOBIntType2	0x00
0x8084_00B4	-	Write Only	GPIOBEOI	-
0x8084_00B8	GPIOBIntEn	R/W	GPIOBIntEn	0x00
0x8084_00BC	IntStsB	Read only	-	0x00
0x8084_00C0	RawIntStsB	Read only	-	Note 3
0x8084_00C4	GPIOBDB	R/W	GPIOBDB	0x00
0x8084_00C8	EEDrive	R/W	EEDrive	0x00

1. A read from the data register returns the value of the GPIO module input port. These ports have a default pin assignment. The read value default is the pin state based on the default pin map.

2. Port E bits 1 and 0 provide the LED driver function. The Port E[1:0] defaults to drive high. A read from the Port E data register would be expected to return 0x03, if the other pins mapped to Port E inputs are zero. However since the Port E[7:2] inputs are mapped to IDE control signals, the default read value will depend on the default action of the IDE controller and the external interface.

3. The RAWSTATUSx registers have pin dependent default read states. The interrupt control registers default to low level sensitive interrupt on reset. Therefore the external pin state will ripple through the interrupt logic to determine the RAWSTATUSx default.

Register Descriptions

PxDR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								PxDATA							

Address:

Copyright 2007 Cirrus Logic

PADR: 0x8084_0000 - Read/Write
PBDR: 0x8084_0004 - Read/Write
PCDR: 0x8084_0008 - Read/Write
PDDR: 0x8084_000C - Read/Write
PEDR: 0x8084_0020 - Read/Write
PFDR: 0x8084_0030 - Read/Write
PGDR: 0x8084_0038 - Read/Write
PHDR: 0x8084_0040 - Read/Write

Definition:

Port x Data Register. Values written to this 8-bit read/write register will be output on port x pins if the corresponding data direction bits are set HIGH (port output). Values read from this register reflect the external state of Port x inputs. All bits are cleared by a system reset. ("X." stands for a letter, A through H.)

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

PxDATA: Port x 8-bit data.

PxDDR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								PxDIR							

Address:

PADDR: 0x8084_0010 - Read/Write
PBDDR: 0x8084_0014 - Read/Write
PCDDR: 0x8084_0018 - Read/Write
PDDDR: 0x8084_001C - Read/Write
PEDDR: 0x8084_0024 - Read/Write
PFDDR: 0x8084_0034 - Read/Write
PGDDR: 0x8084_003C - Read/Write
PHDDR: 0x8084_0044 - Read/Write

Definition:

Port x Data Direction Register. Bits cleared in this 8-bit read/write register will select the corresponding pin in port x to become an input, setting a bit sets the pin to output. All bits are cleared by a system reset. ("X." stands for a letter, A through H.)

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

Copyright 2007 Cirrus Logic

PxDIR: Port x direction bits.

GPIOxIntEn

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								PxINT							

Address:

GPIOAIntEn: 0x8084_009C - Read/Write
GPIOBIntEn: 0x8084_00B8 - Read/Write
GPIOFIntEn: 0x8084_0058 - Read/Write

Definition:

The GPIO Interrupt Enable register controls which bits of port A/B/F are to be configured as interrupts. A "1" written to a bit in this register will configure the bit on port A/B/F to become an interrupt. The user must make sure that the direction of port A/B/F is set to input (PxDDR defaults to input on reset). Writing a "0" (default on reset) to a bit in the register will configure that bit on port A/B/F as a normal GPIO port and the interrupt output corresponding to that bit will be zeroed. The user can read the inputs on port A/B/F in either mode via the PxDR.

The interrupt type is controlled by the GPIOxINTTYPE1/2 registers described in the following sections.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

PxINT: Port x interrupt enables.

GPIOxIntType1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								PxINTE							

Address:

GPIOAIntType1: 0x8084_0090 - Read/Write
GPIOBIntType1: 0x8084_00AC - Read/Write
GPIOFIntType1: 0x8084_004C - Read/Write

Definition:

Copyright 2007 Cirrus Logic

The INTTYPE1 register controls what type of INTERRUPT can occur on Port A/B/F. Level sensitive when "0" is written to a bit location ("0" default on reset), edge sensitive when "1" is written to a bit location (the type of edge/level is controlled by the INTTYPE2 register). The user must make sure that the direction of port A/B/F is set to input and the corresponding bit in the GPIO INTERRUPT ENABLE register is set to allow the interrupt.

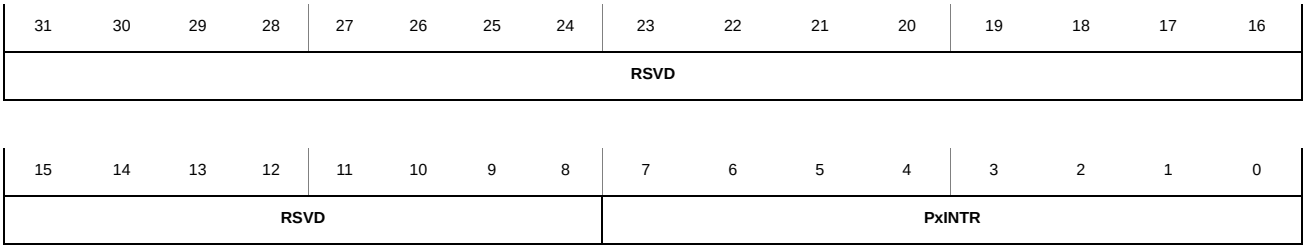
All bits are cleared by a system reset.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

PxINTE: Determines which type of interrupt may occur.

GPIOxIntType2



Address:

GPIOAIntType2: 0x8084_0094 - Read/Write
GPIOBIntType2: 0x8084_00B0 - Read/Write
GPIOFIntType2: 0x8084_0050 - Read/Write

Definition:

The GPIOxINTTYPE2 registers controls the type of edge/level sensitive interrupt that can occur on the bits in Ports A/B/F.

The interrupt is rising edge or high level sensitive if a "1" is written to the corresponding bit in GPIOxINTTYPE2 and falling edge or low level sensitive if a "0" is written to the corresponding bit in GPIOxINTTYPE2. The user must make sure that the direction of port A/B/F is set to input and the corresponding bits in the GPIO Interrupt Enable register and GPIOxINTTYPE1 are set correctly in order for this register to have any effect. For edge sensitive interrupts the GPIOxINTTYPE1 bit should set high and low for level sensitive interrupts.

All bits are cleared by a system reset.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

PxINTR: Determines which type of edge or level sensitive interrupt may occur.

Copyright 2007 Cirrus Logic

GPIOxEOI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								PxINTC							

Address:

GPIOAEOI: 0x8084_0098 - Write Only
GPIOBEOI: 0x8084_00B4 - Write Only
GPIOFEOI: 0x8084_0054 - Write Only

Definition:

In order to clear an edge sensitive interrupt that can occur over port A/B/F, the user must write a data value of "1" to the corresponding bit in the GPIOxEOI register bit. The user must clear an interrupt before changing Port A/B/F from interrupt mode to GPIO mode as the interrupts are cleared once this change has occurred. Once an interrupt has occurred and the interrupt service routine has started, one of the first instructions should be a write to this location in order to clear the interrupt so that subsequent interrupts on the same line are not missed.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

PxINTC: Clears Interrupts

GPIOxDB

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								PxINTDB							

Address:

GPIOADB: 0x8084_00A8 - Read/Write
GPIOBDB: 0x8084_00C4 - Read/Write
GPIOFDB: 0x8084_0064 - Read/Write

Definition:

For each port, if interrupts are enabled, it is possible to debounce the input signal. Setting a bit in this register enables debouncing for the corresponding interrupt signal; clearing the bit disables debouncing. Debouncing is implemented by passing the input signal through a 2-bit shift register clocked by a 128 Hz clock.

Copyright 2007 Cirrus Logic

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

PxINTDB: Interrupt debounce enable.

RawIntStsX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								PxINTRS							

Address:

RawIntStsA: 0x8084_00A4 - Read Only
RawIntStsB: 0x8084_00C0 - Read Only
RawIntStsF: 0x8084_0060 - Read Only

Definition:

Each bit in the register reports whether an interrupt would be signalled if the interrupt were enabled for the corresponding port; a set bit indicates that an interrupt would be signalled. The value reported is unaffected by whether interrupts are enabled or disabled. How a bit is set depends on the interrupt type. If the interrupt is level sensitive active high, it reflects the pin value. If level sensitive active low, it reflects the inverse of the pin value. If the interrupt is edge triggered, the bit latches a one whenever the proper level change occurs. How a bit is cleared also depends on the interrupt type. When an interrupt is level sensitive, it is cleared when not asserted. When edge triggered, it is cleared by writing the corresponding bit in GPIOxEOI. Note that the value of a bit is a debounced value if debouncing is enabled.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

PxINTRS: Raw Interrupt Status.

IntStsX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								PxINTS							

Address:

Copyright 2007 Cirrus Logic

IntStsA: 0x8084_00A0 - Read Only
IntStsB: 0x8084_00BC - Read Only
IntStsF: 0x8084_005C - Read Only

Definition:

For each port, this register reports the same value as the RawIntStsX register for each bit whose corresponding interrupt is enabled. Bits whose corresponding interrupt is not enabled report "0".

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

PxINTS: Masked Interrupt Status.

EEDrive

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD														DATOD	CLKOD

Address:

0x8084_00C8 - Read/Write

Copyright 2007 Cirrus Logic

Definition:

EEPROM interface pin drive type control. Defines the driver type for the **EECLK** and **EEDAT** pins. When set, the corresponding pin is open drain, so that the pin will require an external pull-up. When clear, the corresponding pin is a normal CMOS driver. DATOD controls the **EEDAT** pin. CLKOD controls the **EECLK** pin.

Bit Descriptions:

RSVD:	Reserved. Unknown During Read.
DATOD:	Defines the EEDAT pin output driver.
CLKOD:	Defines the EECLK pin output driver.

Copyright 2007 Cirrus Logic

28

Copyright 2007 Cirrus Logic

Chapter 29
Security

29

29.1 Introduction

Security is a generalized architecture consisting of Boot ROM, laser fuses and proprietary circuitry for secure hardware initialization. In the context of this environment, the chip supports multiple digital-rights-management content-protection from several security vendors, (such as Microsoft and InterTrust). It exceeds all the requirements set forth by the Secure Digital Music Initiative and allows for protection of object code as well as content.

29.2 Features

Key features include:

- 256 bits of laser fuse for permanent IDs and passwords.
- Security boot firmware and private passwords are “invisible” except when the IC is “locked”.
- Each instantiation of the system software may be uniquely encoded and protected using the private ID.
- Multiple security vendors can co-exist in the same system.
- JTAG functionality is disabled when security is enabled.
- External boot is disabled when security is enabled.

29.3 Contact Information

Contact Cirrus Logic at *www.cirrus.com* for additional information regarding security features.

Copyright 2007 Cirrus Logic

29.4 Registers

This section contains the detailed register descriptions for some of the registers in the Security block. Table 29-1 shows the address map for the registers in this block, followed by a detailed listing for each register.

Note: Most Security registers are not documented in this Guide. Please contact Cirrus Logic at www.cirrus.com for additional information regarding security features.

29

Table 29-1. Security Register List

Address	Name	SW Locked	Type	Size	Description
0x8083_2714	ExtensionID	No	R	32	PartID for EP93XX devices

Register Descriptions

ExtensionID

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								PartID							RSVD

Address:

0x8083_2714 - Read Only

Definition:

This register contains the PartID for EP93XX processors.

Bit Descriptions:

RSVD: Reserved. Unknown During Read.

PartID: Identification number for each type of EP93XX processors. See the respective EP93xx processor data sheet to determine the PARTID for a specific EP93xx processor.

Copyright 2007 Cirrus Logic

Chapter 30
Glossary

Table 30-1. Glossary

Term	Definition
AC'97	Serial Audio data transmission standard
ADC	Analog to Digital Converter
AMBA	Advanced Micro-controller Bus Architecture
APB	Advanced Peripheral Bus
ARM920T	ARM9 is the general purpose processor core in the EP93xx processors.
ATAPI	AT Advanced Packet Interface
Buffer	A "buffer" refers to the area in system memory that is characterized by a buffer descriptor, that is, a start address and the length of the buffer in bytes.
CODEC	Coder/Decoder
CRC	Cyclic Redundancy Check
DAC	Digital to Analog Converter
DMA	Direct Memory Access
EEPROM	Electronically Erasable Programmable Read Only Memory
FIQ	Fast Interrupt Request
FIR	Fast Infrared
FLASH	FLASH memory
GPIO	General Purpose Input Output
HDLC	High-level Data Link Control
I2C	See I^2S
I^2S	Inter-IC Sound, also known as I2S
ICE	In-circuit Emulator
IDE	Integrated Drive Electronics
Ir or IR	Infrared
IrDA	Infrared Data Association
IRQ	Standard Interrupt Request
ISO	International Standards Organization
JTAG	Joint Test Action Group
LCDDAC	Liquid Crystal Display Digital to Analog Converter

Copyright 2007 Cirrus Logic

Table 30-1. Glossary (Continued)

Term	Definition
LED	Light Emitting Diode
MAC	Media Access Controller - Ethernet
MII	Media Independent Interface
MIR	Medium Infrared
MMU	Memory Management Unit
OHCI	Open Host Controller Interface
PHY	Physical layer
PIO	Programmed I/O
PLL	Phase Locked Loop
PPM	Pulse Position Modulation
RISC	Reduced Instruction Set Computer
RTC	The ARM Real Time Clock
RTL	Register Transfer Level
RTZ	Return-to-zero
RZI	Return-to-zero Inverted
SDLC	Synchronous Data Link Control
SDMI	Secure Digital Music Initiative
SDRAM	Synchronous Dynamic Random Access Memory
SIR	Slow Infrared
SPI	Serial Peripheral Interface. Also known as SSP. Synchronous Serial Interface supporting the Motorola SPI format.
SRAM	Static Random Access Memory
Syscon	System control registers
TFT	Thin Film Transistor
TLB	Translation Lookaside Buffer
TTB	Translation Table Base
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
VIC	Vectored Interrupt Controller
Watchdog	A count down timer designed to restart the ARM Core if the processor hangs.

Copyright 2007 Cirrus Logic

Chapter 31

EP93XX Register List

Table 31-1 provides an alphabetical list of the EP93XX registers discussed in this manual. Click on the register name to view a detailed discussion of that register.

Table 31-1. EP93xx Register List

Register Name	Page Number
AC97RISRx	22-13
GrySclLUTB	7-73
AC97DRx	22-6
AC97EOI	22-20
AC97GCIS	22-24
AC97GCR	22-21
AC97GIS	22-19
AC97IEx	22-15
AC97IM	22-20
AC97ISRx	22-14
AC97Reset	22-22
AC97RGIS	22-18
AC97RXCRx	22-7
AC97S12Data	22-17
AC97S1Data	22-15
AC97S2Data	22-16
AC97SRx	22-12
AC97SYNC	22-23
AC97TXCRx	22-10
ACRate	7-56
AFP	9-52
APBWait	5-22
BACKGROUND	8-35
BASEx	10-29
BCRx	10-41
BkgrndOffset	7-65

Copyright 2007 Cirrus Logic

31

Table 31-1. EP93xx Register List

Register Name	Page Number
BlinkMask	7-63
BlinkPattrn	7-64
BlinkRate	7-63
BLKDESTHEIGHT	8-28
BLKDESTSTRT	8-25
BLKDESTWIDTH	8-27
BLKSRCWIDTH	8-26
BLOCKCTRL	8-30
BLOCKMASK	8-34
BMCtl	9-67
BMSts	9-70
BootModeClr	5-24
BootSts	13-17
Brightness	7-50
BusMstrArb	5-23
CHIP_ID	5-33
ClkSet1	5-18
ClkSet2	5-20
ColorLUT	7-77
CONTROL	10-22
CONTROL	10-31
CURRENTx	10-30
CursorAdrReset	7-67
CursorAdrStart	7-66
CursorBlinkColor1,	7-69
CursorBlinkColor2	7-69
CursorBlinkRateCtrl	7-72
CursorColor1,	7-69
CursorColor2,	7-69
CursorDScanLHYLoc	7-71
CursorSize	7-68
CursorXYLoc	7-70
DAR_BASEx	10-43
DAR_CURRENTx	10-44
DESTLINELENGTH	8-29
DESTPIXELSTRT	8-23

Copyright 2007 Cirrus Logic

31

Table 31-1. EP93xx Register List

Register Name	Page Number
DeviceCfg	5-25
DiagAd	9-48
DiagDa	9-49
DMAChArb	10-45
DMAGlInt	10-44
EEDrive	28-16
EOLOffset	7-49
ExtensionID	29-2
FCF	9-51
FCT	9-50
FIFOLevel	7-56
FIIR	17-37
FIMR	17-36
FISR	17-35
GlConfig	13-14
GlIntFrc	9-64
GlIntMsk	9-63
GlIntROSts	9-64
GlIntSts	9-62
GPIOxDB	28-14
GPIOxEOI	28-14
GPIOxIntEn	28-12
GPIOxIntType1	28-12
GPIOxIntType2	28-13
GrySclLUTG,	7-73
GrySclLUTR,	7-73
GT	9-49
HActiveStrtStop	7-43
HashTbl	9-54
HBlankStrtStop	7-44
HcBulkCurrentED	11-23
HcBulkHeadED	11-22
HcCommandStatus	11-15
HcControl	11-12
HcControlCurrentED	11-22
HcControlHeadED	11-21

Copyright 2007 Cirrus Logic

31

Table 31-1. EP93xx Register List

Register Name	Page Number
HcDoneHead	11-24
HcFmInterval	11-24
HcFmNumber	11-26
HcFmRemaining	11-25
HcHCCA	11-20
HcInterruptDisable	11-19
HcInterruptEnable	11-18
HcInterruptStatus	11-17
HClkStrtStop	7-45
HClkTotal	7-42
HcLSThreshold	11-27
HcPeriodCurrentED	11-20
HcPeriodicStart	11-26
HcRevision	11-12
HcRhDescriptorA	11-28
HcRhDescriptorB	11-29
HcRhPortStatusx	11-32
HcRhStatus	11-30
HSigStrtStop	7-80
HSyncStrtStop	7-42
I2SClkDiv	5-31
I2SGlCtrl	21-31
I2SGlSts	21-29
I2SRX0En	21-24
I2SRX0Lft	21-19
I2SRX0Rt	21-20
I2SRX1En	21-24
I2SRX1Lft	21-20
I2SRX1Rt	21-21
I2SRX2En	21-25
I2SRX2Lft	21-21
I2SRX2Rt	21-22
I2SRXClkCfg	21-27
I2SRXCtrl	21-23
I2SRXLinCtrlData	21-22
I2SRXWrdLen	21-23

Copyright 2007 Cirrus Logic

31

Table 31-1. EP93xx Register List

Register Name	Page Number
I2STX0En	21-17
I2STX0Lft	21-13
I2STX0Rt	21-13
I2STX1En	21-18
I2STX1Lft	21-14
I2STX1Rt	21-14
I2STX2En	21-18
I2STX2Lft	21-15
I2STX2Rt	21-15
I2STXClkCfg	21-26
I2STXCtrl	21-16
I2STXLinCtrlData	21-16
I2STXWrdLen	21-17
IDECfg	27-11
IDECtrl	27-10
IDEDataIn	27-14
IDEDataOut	27-13
IDEMDMADataIn	27-15
IDEMDMADataOut	27-14
IDEMDMAOp	27-12
IDEUDMADataIn	27-16
IDEUDMADataOut	27-15
IDEUDMADebug	27-18
IDEUDMAOp	27-12
IDEUDMARdBufSts	27-19
IDEUDMASts	27-16
IDEUDMAWrBufSts	27-18
IndAd	9-53
IntEn	9-57
INTERRUPT	10-35
INTERRUPT	10-25
IntStsP/IntStsC	9-60
IntStsX	28-15
IrAdrMatchVal	17-25
IrCtrl	17-24
IrData	17-27

Copyright 2007 Cirrus Logic

Table 31-1. EP93xx Register List

Register Name	Page Number
IrDataTail	17-28
IrDMACR	17-30
IrEnable	17-23
IrFlag	17-26
IrRIB	17-28
IrTR0	17-30
KeyDiagnostic	26-8
KeyScanInit	26-6
KeyTchClkDiv	5-32
LineCarry	7-49
LINEINC	8-36
LINEINIT	8-36
LineLength	7-47
LINEPATTRN	8-37
LUTSwCtrl	7-76
MAXCNTx	10-29
MaxFrmLen	9-91
MIICmd	9-65
MIIData	9-66
MIIR	17-34
MIISts	9-66
MIMR	17-33
MIRClkDiv	5-30
MISR	17-32
ParllIfIn	7-61
ParllIfOut	7-60
PattrnMask	7-65
PCAttribute	12-13
PCCommon	12-14
PCIO	12-15
PCMCIACtrl	12-17
PixelMode	7-57
PPALLOC	10-23
PWMxDutyCycle	24-4
PWMxEn	24-5
PWMxInvert	24-5

Copyright 2007 Cirrus Logic

31

Table 31-1. EP93xx Register List

Register Name	Page Number
PWMxTermCnt	24-4
PwrCnt	5-15
PwrSts	5-14
PxDDR	28-11
PxDR	28-10
RasterSWLock	7-55
RawIntStsX	28-15
Receive Descriptor Format - First Word	9-15
Receive Descriptor Format - Second Word	9-15
Receive Status - First Word	9-18
Receive Status - Second Word	9-20
RefrshTimr	13-16
REMAIN	10-28
RTCCtrl	20-6
RTCData	20-4
RTCLoad	20-6
RTCMatch	20-5
RTCSts	20-5
RTCSWComp	20-7
RXBCA	9-74
RXBufThrshld	9-85
RXCtl	9-41
RXDCurAdd	9-73
RXDEnq	9-74
RXDQBAdd	9-71
RXDQBLen	9-72
RXDQCurLen	9-72
RXDThrshld	9-89
RXHdrLen	9-78
RXMissCnt	9-55
RXRuntCnt	9-56
RXStsEnq	9-78
RXStsQBAdd	9-75
RXStsQBLen	9-76
RXStsQCurAdd	9-77
RXStsQCurLen	9-76

Copyright 2007 Cirrus Logic

31

Table 31-1. EP93xx Register List

Register Name	Page Number
RXStsThrshld	9-87
SAR_BASEx	10-42
SAR_CURRENTx	10-43
ScratchReg0, ScratchReg1	5-22
ScrnLines	7-47
SDRAMDevCfg[3:0]	13-18
SelfCtl	9-46
SigClrStr	7-81
SIRTR0	17-31
SMCBCR[7:0}	12-10
SRCLINELENGTH	8-26
SRCPIXELSTRT	8-23
SSPCPSR	23-18
SSPCR0	23-13
SSPCR1	23-14
SSPDR	23-16
SSPIIR / SSPICR	23-18
SSPSR	23-17
Standby and Halt	5-17
STATUS	10-26
STATUS	10-37
STFClr	5-18
SysCfg	P-4
SysCfg	5-34
SysSWLock	5-35
TEOI	5-17
TestCtl	9-57
Timer1Clear,	18-5
Timer1Control,	18-6
Timer1Load,	18-3
Timer1Value,	18-4
Timer2Clear,	18-5
Timer2Control,	18-6
Timer2Load	18-3
Timer2Value	18-4
Timer3Clear	18-5

Copyright 2007 Cirrus Logic

31

Table 31-1. EP93xx Register List

Register Name	Page Number
Timer3Control	18-6
Timer3Load	18-3
Timer3Value	18-4
Timer4ValueHigh	18-7
Timer4ValueLow	18-7
Transmit Descriptor Format - First Word	9-28
Transmit Descriptor Format - Second Word	9-29
Transmit Status	9-32
TRANSPATTRN	8-34
TSDischarge, TSXSample, TSYSample, TSDirect, TSDetect	25-20
TSSetup	25-17
TSSetup2	25-22
TSSWLock	25-21
TSXYMaxMin	25-19
TSXYResult	25-19
TXBufThrshld	9-86
TXCollCnt	9-55
TXCtl	9-44
TXDEnq	9-82
TXDQBAdd	9-79
TXDQBLen	9-80
TXDQCurAdd	9-81
TXDQCurLen	9-80
TXDThrshld	9-90
TXStsQBAdd	9-82
TXStsQBLen	9-83
TXStsQCurAdd	9-84
TXStsQCurLen	9-84
TXStsThrshld	9-88
UART1Ctrl	14-22
UART1Data	14-17
UART1DMACtrl	14-25
UART1Flag	14-22
UART1HDLCAddMask	14-31
UART1HDLCAddMtchVal	14-30

Copyright 2007 Cirrus Logic

31

Table 31-1. EP93xx Register List

Register Name	Page Number
UART1HDLCCtrl	14-27
UART1HDLCRXInfoBuf	14-31
UART1HDLCSts	14-32
UART1IntIDIntClr	14-24
UART1LinCtrlHigh	14-19
UART1LinCtrlLow	14-21
UART1LinCtrlMid	14-20
UART1ModemCtrl	14-25
UART1ModemSts	14-26
UART1RXSts	14-18
UART2Ctrl	15-12
UART2Data	15-7
UART2DMACtrl	15-16
UART2Flag	15-13
UART2IntIDIntClr	15-14
UART2IrLowPwrCntr	15-15
UART2LinCtrlHigh	15-9
UART2LinCtrlLow	15-11
UART2LinCtrlMid	15-10
UART2RXSts	15-8
UART2TMR	15-17
UART3Ctrl	16-8
UART3Data	16-3
UART3DMACtrl	16-11
UART3Flag	16-9
UART3HDLCAddMask	16-16
UART3HDLCAddMtchVal	16-16
UART3HDLCCtrl	16-13
UART3HDLCRXInfoBuf	16-17
UART3HDLCSts	16-18
UART3IntIDIntClr	16-10
UART3LinCtrlHigh	16-5
UART3LinCtrlLow	16-7
UART3LinCtrlMid	16-7
UART3LowPwrCntr	16-11
UART3ModemCtrl	16-12

Copyright 2007 Cirrus Logic

31

Table 31-1. EP93xx Register List

Register Name	Page Number
UART3RXSts	16-4
USBCfgCtrl	11-36
USBHCISts	11-37
VActiveStrtStop	7-39
VBlankStrtStop	7-40
VClkStrtStop	7-41
VICxDefVectAddr	6-15
VICxFIQStatus	6-10
VICxIntEnable	6-11
VICxIntEnClear	6-12
VICxIntSelect	6-11
VICxIRQStatus	6-9
VICxProtection	6-13
VICxRawIntr	6-10
VICxSoftInt	6-12
VICxSoftIntClear	6-13
VICxVectAdd12,	6-16
VICxVectAdd9,	6-16
VICxVectAddr	6-14
VICxVectAddr0	6-15
VICxVectAddr1,	6-15
VICxVectAddr10,	6-16
VICxVectAddr11,	6-16
VICxVectAddr13,	6-16
VICxVectAddr14,	6-16
VICxVectAddr15	6-16
VICxVectAddr2,	6-15
VICxVectAddr3,	6-15
VICxVectAddr4,	6-15
VICxVectAddr5,	6-15
VICxVectAddr6	6-15
VICxVectAddr7,	6-16
VICxVectAddr8,	6-16
VICxVectCntl0,	6-17
VICxVectCntl1,	6-17
VICxVectCntl10,	6-17

Copyright 2007 Cirrus Logic

Table 31-1. EP93xx Register List

Register Name	Page Number
VICxVectCntl11,	6-17
VICxVectCntl12,	6-17
VICxVectCntl13,	6-17
VICxVectCntl14,	6-17
VICxVectCntl15	6-18
VICxVectCntl2,	6-17
VICxVectCntl3,	6-17
VICxVectCntl4,	6-17
VICxVectCntl5,	6-17
VICxVectCntl6,	6-17
VICxVectCntl7,	6-17
VICxVectCntl8,	6-17
VICxVectCntl9,	6-17
VidClkDiv	5-29
VideoAttribs	7-51
VidScrnHPage	7-46
VidScrnPage	7-46
VidSigCtrl	7-78
VidSigRsltVal	7-77
VLineStep	7-48
VLinesTotal	7-38
VSigStrtStop	7-79
VSyncStrtStop	7-38
Watchdog	19-3
WDStatus	19-5

Copyright 2007 Cirrus Logic

Shenzhen Haoxinsheng Technology Co., Ltd.

haoxinshengic.com

DATASHEET

Whatsapp: +8615816789795

Skype: 8615816789795

WeChat: 15816789795

Viber: +8615816789795

E-mail: info@haoxinshengic.com