MC9328MXL
i.MX Integrated Portable System Processor

Reference Manual

Document Number: MC9328MXLRM
Rev. 5.1
06/2007

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

***For Literature Requests Only*:**
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-521-6274 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. ARM and the ARM POWERED logo are the registered trademarks of ARM Limited. ARM9, ARM920T, ARM9TDMI, ARMv4T, ARM7, ARM7TDMI, Thumb, and StrongARM are trademarks of ARM Limited. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2003, 2004, 2005, 2006, 2007. All rights reserved.

Contents

About This Book

Audience . . . xxv
Organization . . . xxv
Document Revision History . . . xxvii
Suggested Reading . . . xxvii
Conventions . . . xxviii
Definitions, Acronyms, and Abbreviations . . . xxviii

Chapter 1
Introduction

1.1 Block Diagram . . . 1-1
1.2 Features . . . 1-2
1.3 ARM920T Microprocessor Core . . . 1-3
1.4 AHB to IP Bus Interfaces (AIPIs) . . . 1-3
1.5 External Interface Module (EIM) . . . 1-4
1.6 SDRAM Controller (SDRAMC) . . . 1-4
1.7 Clock Generation Module (CGM) and Power Control Module . . . 1-4
1.8 Two Universal Asynchronous Receiver/Transmitters (UART 1, UART 2) . . . 1-5
1.9 Two Serial Peripheral Interfaces (SPI) . . . 1-5
1.10 Two General-Purpose 32-Bit Counters/Timers . . . 1-5
1.11 Watchdog Timer . . . 1-5
1.12 Real-Time Clock/Sampling Timer (RTC) . . . 1-6
1.13 LCD Controller (LCDC) . . . 1-6
1.14 Pulse-Width Modulation (PWM) Module . . . 1-7
1.15 Universal Serial Bus (USB) Device . . . 1-7
1.16 Multimedia Card and Secure Digital (MMC/SD) Host Controller . . . 1-7
1.17 Memory Stick® Host Controller (MSHC) . . . 1-8
1.18 Direct Memory Access Controller (DMAC) . . . 1-8
1.19 Synchronous Serial Interface and Inter-IC Sound (SSI/I^2S) Module . . . 1-8
1.20 Inter-IC (I^2C) Bus Module . . . 1-9
1.21 Video Port . . . 1-9
1.22 General-Purpose I/O (GPIO) Ports . . . 1-9
1.23 Bootstrap Mode . . . 1-9
1.24 Multimedia Accelerator (MMA) . . . 1-10
1.25 Power Management Features . . . 1-10
1.26 Operating Voltage Range . . . 1-10
1.27 Packaging . . . 1-10

Chapter 2
Signal Descriptions and Pin Assignments

2.1 Signal Descriptions . . . 2-1
2.2 I/O Pads Power Supply and Signal Multiplexing Scheme . . . 2-7

Chapter 3
Memory Map

3.1 Memory Space 3-1
3.1.1 Memory Map 3-1
3.1.2 Internal Register Space 3-5
3.1.3 External Memory 3-5
3.1.4 Double Map Image 3-5
3.2 Internal Registers 3-6

Chapter 4
ARM920T Processor

4.1 Introduction. 4-1
4.2 ARM920T Macrocell 4-2
4.2.1 Caches. 4-3
4.2.2 Cache Lock-Down 4-3
4.2.3 Write Buffer 4-3
4.2.4 PATAG RAM. 4-3
4.2.5 MMUs. 4-3
4.2.6 System Controller 4-3
4.2.7 Control Coprocessor (CP15). 4-4
4.3 ARMv4T Architecture 4-4
4.3.1 Registers 4-4
4.3.2 Modes and Exception Handling 4-4
4.3.3 Status Registers. 4-5
4.3.4 Exception Types 4-5
4.3.5 Conditional Execution 4-5
4.4 Four Classes of Instructions 4-5
4.4.1 Data Processing Instructions. 4-5
4.4.2 Load and Store Instructions 4-6
4.4.2.1 Addressing Modes 4-6
4.4.2.2 Block Transfers. 4-6
4.4.3 Branch Instructions. 4-7
4.4.3.1 Branch with Link 4-7
4.4.4 Coprocessor Instructions. 4-7
4.5 The ARM9 Instruction Set 4-7
4.6 The ARM Thumb Instruction Set 4-8
4.6.1 ARM920T Modes and Registers 4-9

Chapter 5
Embedded Trace Macrocell (ETM)

5.1 Introduction to the ETM 5-1
5.2 Programming and Reading ETM Registers 5-1
5.3 Pin Configuration for ETM. 5-2

Chapter 6
Reset Module

6.1 Functional Description of the Reset Module 6-1
6.1.1 Global Reset 6-1
6.1.2 ARM920T Processor Reset 6-2
6.2 Programming Model 6-3
6.2.1 Reset Source Register (RSR) 6-3

Chapter 7
AHB to IP Bus Interface (AIPI)

7.1 Overview 7-1
7.1.1 Features 7-1
7.1.2 General Information 7-1
7.2 Programming Model 7-10
7.2.1 Peripheral Size Registers[1:0] 7-12
7.2.1.1 AIPI1 Peripheral Size Register 0 and AIPI2 Peripheral Size Register 0 7-12
7.2.1.2 AIPI1 Peripheral Size Register 1 and AIPI2 Peripheral Size Register 1 7-13
7.2.2 Peripheral Access Registers 7-14
7.2.3 Peripheral Control Register 7-15
7.2.4 Time-Out Status Register 7-16
7.3 Programming Example 7-17
7.3.1 Data Access to 8-Bit Peripherals 7-17
7.3.2 Data Access to 16-Bit Peripherals 7-18
7.3.3 Data Access to 32-Bit Peripherals 7-19
7.3.4 Special Consideration for Non-Natural Size Access 7-20

Chapter 8
System Control

8.1 Programming Model 8-1
8.1.1 Silicon ID Register 8-2
8.1.2 Function Multiplexing Control Register 8-2
8.1.3 Global Peripheral Control Register 8-4
8.1.4 Global Clock Control Register 8-6
8.2 System Boot Mode Selection 8-7

Chapter 9
Bootstrap Mode Operation

9.1 Operation 9-1
9.1.1 Entering Bootstrap Mode 9-1
9.1.2 Bootstrap Record Format 9-2
9.1.3 Registers Used in Bootloader Program 9-2
9.1.4 Setting Up the RS-232 Terminal 9-3

9.1.5 Changing the Speed of Communication . . . 9-3
9.2 B-Record Example . . . 9-3
9.3 Instruction Buffer Usage . . . 9-3
9.4 Simple Read/Write Examples . . . 9-5
9.5 Bootloader Flowchart . . . 9-7
9.6 Special Notes . . . 9-7

Chapter 10
Interrupt Controller (AITC)

10.1 Introduction . . . 10-1
10.2 Operation . . . 10-2
10.3 AITC Interrupt Controller Signals . . . 10-3
10.4 Programming Model . . . 10-4
10.4.1 Interrupt Control Register . . . 10-7
10.4.2 Normal Interrupt Mask Register . . . 10-9
10.4.3 Interrupt Enable Number Register . . . 10-10
10.4.4 Interrupt Disable Number Register . . . 10-11
10.4.5 Interrupt Enable Register High and Interrupt Enable Register Low . . . 10-12
10.4.5.1 Interrupt Enable Register High . . . 10-12
10.4.5.2 Interrupt Enable Register Low . . . 10-13
10.4.6 Interrupt Type Register High and Interrupt Type Register Low . . . 10-14
10.4.6.1 Interrupt Type Register High . . . 10-14
10.4.6.2 Interrupt Type Register Low . . . 10-15
10.4.7 Normal Interrupt Priority Level Registers . . . 10-15
10.4.7.1 Normal Interrupt Priority Level Register 7 . . . 10-16
10.4.7.2 Normal Interrupt Priority Level Register 6 . . . 10-17
10.4.7.3 Normal Interrupt Priority Level Register 5 . . . 10-18
10.4.7.4 Normal Interrupt Priority Level Register 4 . . . 10-19
10.4.7.5 Normal Interrupt Priority Level Register 3 . . . 10-20
10.4.7.6 Normal Interrupt Priority Level Register 2 . . . 10-21
10.4.7.7 Normal Interrupt Priority Level Register 1 . . . 10-22
10.4.7.8 Normal Interrupt Priority Level Register 0 . . . 10-23
10.4.8 Normal Interrupt Vector and Status Register . . . 10-24
10.4.9 Fast Interrupt Vector and Status Register . . . 10-25
10.4.10 Interrupt Source Register High and Interrupt Source Register Low . . . 10-26
10.4.10.1 Interrupt Source Register High . . . 10-26
10.4.10.2 Interrupt Source Register Low . . . 10-27
10.4.11 Interrupt Force Register High and Interrupt Force Register Low . . . 10-28
10.4.11.1 Interrupt Force Register High . . . 10-28
10.4.11.2 Interrupt Force Register Low . . . 10-29
10.4.12 Normal Interrupt Pending Register High and Normal Interrupt Pending Register Low . 10-30
10.4.12.1 Normal Interrupt Pending Register High . . . 10-30
10.4.12.2 Normal Interrupt Pending Register Low . . . 10-31
10.4.13 Fast Interrupt Pending Register High and Fast Interrupt Pending Register Low . . . 10-32

10.4.13.1 Fast Interrupt Pending Register High . . . 10-32
10.4.13.2 Fast Interrupt Pending Register Low . . . 10-33
10.5 ARM920T Processor Interrupt Controller Operation . . . 10-34
10.5.1 ARM920T Processor Prioritization of Exception Sources . . . 10-34
10.5.2 AITC Prioritization of Interrupt Sources . . . 10-34
10.5.3 Assigning and Enabling Interrupt Sources . . . 10-34
10.5.4 Enabling Interrupts Sources . . . 10-34
10.5.5 Typical Interrupt Entry Sequences . . . 10-35
10.5.6 Writing Reentrant Normal Interrupt Routines . . . 10-36

Chapter 11
External Interface Module (EIM)

11.1 Overview . . . 11-1
11.2 EIM I/O Signals . . . 11-1
11.2.1 Address Bus . . . 11-1
11.2.2 Data Bus . . . 11-1
11.2.3 Read/Write . . . 11-1
11.2.4 Control Signals . . . 11-2
11.2.4.1 $\overline{\text{OE}}$—Output Enable . . . 11-2
11.2.4.2 $\overline{\text{EB}}$ [3:0]—Enable Bytes . . . 11-2
11.2.4.3 $\overline{\text{DTACK}}$—Data Transfer Acknowledge . . . 11-2
11.2.5 Chip Select Outputs . . . 11-2
11.2.5.1 Chip Select 0 ($\overline{\text{CS0}}$) . . . 11-2
11.2.5.2 Chip Select 1–Chip Select 5 ($\overline{\text{CS1}}$–$\overline{\text{CS5}}$) . . . 11-3
11.2.6 Burst Mode Signals . . . 11-3
11.2.6.1 $\overline{\text{BCLK}}$—Burst Clock . . . 11-3
11.2.6.2 $\overline{\text{LBA}}$—Load Burst Address . . . 11-3
11.2.6.3 ECB—End Current Burst . . . 11-3
11.3 Pin Configuration for EIM . . . 11-3
11.4 Typical EIM System Connections . . . 11-5
11.5 EIM Functionality . . . 11-8
11.5.1 Configurable Bus Sizing . . . 11-8
11.5.2 Programmable Output Generation . . . 11-8
11.5.3 Burst Mode Operation . . . 11-8
11.5.4 Burst Clock Divisor . . . 11-8
11.5.5 Burst Clock Start . . . 11-9
11.5.6 Page Mode Emulation . . . 11-9
11.5.7 Error Conditions . . . 11-9
11.6 Programming Model . . . 11-10
11.6.1 Chip Select 0 Control Registers . . . 11-11
11.6.1.1 Chip Select 0 Upper Control Register . . . 11-11
11.6.1.2 Chip Select 0 Lower Control Register . . . 11-12
11.6.2 Chip Select 1–Chip Select 5 Control Registers . . . 11-12
11.6.2.1 Chip Select 1–Chip Select 5 Upper Control Registers . . . 11-12

11.6.2.2 Chip Select 1–Chip Select 5 Lower Control Registers 11-14
11.6.3 EIM Configuration Register 11-20

Chapter 12
Phase-Locked Loop and Clock Controller

12.1 Introduction 12-1
12.2 Clock Sources 12-1
12.2.1 Low Frequency Clock Source 12-1
12.2.2 High Frequency Clock Source 12-1
12.3 DPLL Output Frequency Calculation 12-3
12.3.1 DPLL Phase and Frequency Jitter 12-3
12.4 MC9328MXL Power Management 12-3
12.4.1 PLL Operation at Power-Up 12-3
12.4.2 PLL Operation at Wake-Up 12-4
12.4.3 ARM920T Processor Low-Power Modes 12-4
12.4.4 SDRAM Power Modes 12-4
12.4.5 Power Management in the Clock Controller 12-4
12.5 Programming Model 12-5
12.5.1 Clock Source Control Register 12-5
12.5.2 Peripheral Clock Divider Register 12-7
12.5.3 Programming Digital Phase Locked Loops 12-8
12.5.3.1 MCU PLL Control Register 0 12-9
12.5.3.2 MCU PLL and System Clock Control Register 1 12-10
12.5.4 Generation of 48 MHz Clocks 12-11
12.5.4.1 System PLL Control Register 0 12-12
12.5.4.2 System PLL Control Register 1 12-13

Chapter 13
DMA Controller

13.1 Features 13-1
13.2 Block Diagram 13-2
13.3 Signal Description 13-3
13.3.1 Big Endian and Little Endian 13-4
13.4 Programming Model 13-4
13.4.1 General Registers 13-8
13.4.1.1 DMA Control Register 13-8
13.4.1.2 DMA Interrupt Status Register 13-9
13.4.1.3 DMA Interrupt Mask Register 13-10
13.4.1.4 DMA Burst Time-Out Status Register 13-11
13.4.1.5 DMA Request Time-Out Status Register 13-12
13.4.1.6 DMA Transfer Error Status Register 13-13
13.4.1.7 DMA Buffer Overflow Status Register 13-14
13.4.1.8 DMA Burst Time-Out Control Register 13-15
13.4.2 2D Memory Registers (A and B) 13-16

13.4.2.1 W-Size Registers . . . 13-16
13.4.2.2 X-Size Registers . . . 13-17
13.4.2.3 Y-Size Registers . . . 13-18
13.4.3 Channel Registers . . . 13-18
13.4.3.1 Channel Source Address Register . . . 13-19
13.4.3.2 Destination Address Registers . . . 13-20
13.4.3.3 Channel Count Registers . . . 13-21
13.4.3.4 Channel Control Registers . . . 13-22
13.4.3.5 Channel Request Source Select Registers . . . 13-25
13.4.3.6 Channel Burst Length Registers . . . 13-26
13.4.3.7 Channel Request Time-Out Registers . . . 13-27
13.4.3.8 Channel 0 Bus Utilization Control Register . . . 13-29
13.5 DMA Request Table . . . 13-30

Chapter 14
Watchdog Timer Module

14.1 General Overview . . . 14-1
14.2 Watchdog Timer Operation . . . 14-1
14.2.1 Timing Specifications . . . 14-1
14.2.2 Watchdog During Reset . . . 14-1
14.2.2.1 Power-On Reset . . . 14-1
14.2.2.2 Software Reset . . . 14-2
14.3 Watchdog After Reset . . . 14-2
14.3.1 Initial Load . . . 14-2
14.3.2 Countdown . . . 14-2
14.3.3 Reload . . . 14-2
14.3.4 Time-Out . . . 14-2
14.3.5 Halting the Counter . . . 14-3
14.4 Watchdog Control . . . 14-3
14.4.1 Interrupt Control . . . 14-3
14.4.2 Reset Sources . . . 14-3
14.5 State Machine . . . 14-4
14.6 Watchdog Timer I/O Signals . . . 14-5
14.7 Programming Model . . . 14-6
14.7.1 Watchdog Control Register . . . 14-6
14.7.2 Watchdog Service Register . . . 14-7
14.7.3 Watchdog Status Register . . . 14-8

Chapter 15
Multimedia Accelerator (MMA)

15.1 Introduction . . . 15-1
15.2 MMA Operation . . . 15-1
15.2.1 Memory Access . . . 15-1
15.2.2 MAC . . . 15-2

15.2.2.1 Basic MAC Operation 15-2
15.2.2.2 Data Access. 15-2
15.2.2.3 Cache 15-3
15.2.3 DCT/iDCT 15-4
15.3 Programming Model 15-5
15.3.1 MMA MAC Control Registers 15-6
15.3.1.1 MMA MAC Module Register. 15-6
15.3.1.2 MMA MAC Control Register. 15-7
15.3.1.3 MMA MAC Multiply Counter Register 15-9
15.3.1.4 MMA MAC Accumulate Counter Register 15-10
15.3.1.5 MMA MAC Interrupt Register 15-11
15.3.1.6 MMA MAC Interrupt Mask Register 15-12
15.3.1.7 MMA MAC FIFO Register 15-13
15.3.1.8 MMA MAC FIFO Status Register 15-13
15.3.1.9 MMA MAC Burst Count Register 15-14
15.3.1.10 MMA MAC Bit Select Register 15-15
15.3.2 MMA MAC XY Count Accumulate Register 15-16
15.3.3 MMA MAC X Register Control Registers 15-17
15.3.3.1 MMA MAC X Base Address Register 15-17
15.3.3.2 MMA MAC X Index Register 15-18
15.3.3.3 MMA MAC X Length Register 15-18
15.3.3.4 MMA MAC X Modify Register 15-19
15.3.3.5 MMA MAC X Increment Register 15-20
15.3.3.6 MMA MAC X Count Register 15-20
15.3.4 MMA MAC Y Register Control Registers 15-22
15.3.4.1 MMA MAC Y Base Address Register 15-22
15.3.4.2 MMA MAC Y Index Register 15-22
15.3.4.3 MMA MAC Y Length Register 15-23
15.3.4.4 MMA MAC Y Modify Register 15-24
15.3.4.5 MMA MAC Y Increment Register 15-24
15.3.4.6 MMA MAC Y Count Register 15-25
15.3.5 MMA DCT/iDCT Registers 15-26
15.3.5.1 DCT/iDCT Control Register. 15-26
15.3.5.2 DCT/iDCT Version Register 15-27
15.3.5.3 DCT/iDCT IRQ Enable Register 15-28
15.3.5.4 DCT/iDCT IRQ Status Register 15-29
15.3.5.5 DCT/iDCT Source Data Address 15-30
15.3.5.6 DCT/iDCT Destination Data Address 15-30
15.3.5.7 DCT/iDCT X-Offset Address. 15-31
15.3.5.8 DCT/iDCT Y-Offset Address. 15-31
15.3.5.9 DCT/iDCT XY Count. 15-32
15.3.5.10 DCT/iDCT Skip Address 15-33
15.3.5.11 DCT/iDCT Data FIFO 15-34

Chapter 16
Serial Peripheral Interface Modules (SPI 1 and SPI 2)

16.1 SPI Block Diagram . . . 16-1
16.2 Operation . . . 16-2
16.2.1 Phase and Polarity Configurations . . . 16-2
16.2.2 Signals . . . 16-3
16.2.3 Pin Configuration for SPI 1 and SPI 2 . . . 16-3
16.3 Programming Model . . . 16-5
16.3.1 Receive (RX) Data Registers . . . 16-6
16.3.2 Transmit (TX) Data Registers . . . 16-7
16.3.3 Control Registers . . . 16-8
16.3.4 Interrupt Control/Status Registers . . . 16-10
16.3.5 Test Registers . . . 16-12
16.3.6 Sample Period Control Registers . . . 16-13
16.3.7 DMA Control Registers . . . 16-14
16.3.8 Soft Reset Registers . . . 16-15

Chapter 17
LCD Controller

17.1 Introduction . . . 17-1
17.2 Features . . . 17-1
17.3 LCDC Operation . . . 17-2
17.3.1 LCD Screen Format . . . 17-2
17.3.2 Panning . . . 17-3
17.3.3 Display Data Mapping . . . 17-3
17.3.4 Black-and-White Operation . . . 17-7
17.3.5 Gray-Scale Operation . . . 17-7
17.3.6 Color Generation . . . 17-8
17.3.7 Frame Rate Modulation Control (FRC) . . . 17-10
17.3.8 Panel Interface Signals and Timing . . . 17-11
17.3.8.1 Pin Configuration for LCDC . . . 17-11
17.3.8.2 Passive Matrix Panel Interface Signals . . . 17-12
17.3.8.3 Passive Panel Interface Timing . . . 17-13
17.3.9 8 bpp Mode Color STN Panel . . . 17-14
17.3.9.1 Active Matrix Panel Interface Signals . . . 17-14
17.3.9.2 Active Panel Interface Timing . . . 17-16
17.4 Programming Model . . . 17-18
17.4.1 Screen Start Address Register . . . 17-20
17.4.2 Size Register . . . 17-21
17.4.3 Virtual Page Width Register . . . 17-22
17.4.4 Panel Configuration Register . . . 17-23
17.4.5 Horizontal Configuration Register . . . 17-25
17.4.6 Vertical Configuration Register . . . 17-26
17.4.7 Panning Offset Register . . . 17-27

17.4.8 LCD Cursor Position Register 17-28
17.4.9 LCD Cursor Width Height and Blink Register 17-30
17.4.10 LCD Color Cursor Mapping Register 17-31
17.4.11 Sharp Configuration 1 Register 17-32
17.4.12 PWM Contrast Control Register 17-35
17.4.13 Refresh Mode Control Register 17-36
17.4.14 DMA Control Register 17-37
17.4.15 Interrupt Configuration Register 17-38
17.4.16 Interrupt Status Register 17-39
17.4.17 Mapping RAM Registers 17-40
17.4.17.1 One Bit/Pixel Monochrome Mode 17-40
17.4.17.2 Four Bits/Pixel Gray-Scale Mode 17-40
17.4.17.3 Four Bits/Pixel Passive Matrix Color Mode 17-41
17.4.17.4 Eight Bits/Pixel Passive Matrix Color Mode 17-41
17.4.17.5 Four Bits/Pixel Active Matrix Color Mode 17-42
17.4.17.6 Eight Bits/Pixel Active Matrix Color Mode 17-42
17.4.17.7 Twelve Bits/Pixel and Sixteen Bits/Pixel Active Matrix Color Mode 17-43

Chapter 18
Multimedia Card/Secure Digital Host Controller Module (MMC/SD)

18.1 Introduction 18-1
18.2 Features List 18-1
18.3 MMC/SD Module Block Diagram 18-2
18.4 MMC/SD Module and Card Information 18-3
18.4.1 MMC and SD Card Pin Assignments and Registers 18-3
18.4.2 Communication 18-4
18.4.3 Signal Description 18-4
18.4.4 Pin Configuration for the MMC/SD Module 18-5
18.5 Functional Description 18-5
18.5.1 DMA Interface 18-6
18.5.1.1 DMA Burst Request 18-6
18.5.1.2 Write-Error Detection 18-8
18.5.2 Memory Controller (Register Handler) 18-8
18.5.2.1 SD I/O—IRQ and ReadWait Service Handling 18-8
18.5.2.2 Card Detection 18-9
18.5.2.3 MMC/SD Module Interrupt Handling 18-9
18.5.3 Logic and Command Interpreters 18-9
18.5.4 System Clock Controller 18-11
18.5.4.1 Card Clock Control 18-12
18.5.5 Transceiver 18-12
18.6 Programming Model 18-13
18.6.1 MMC/SD Clock Control Register 18-14
18.6.2 MMC/SD Status Register 18-16
18.6.3 MMC/SD Clock Rate Register 18-19

18.6.4 MMC/SD Command and Data Control Register 18-20
18.6.5 MMC/SD Response Time-Out Register 18-22
18.6.6 MMC/SD Read Time-Out Register 18-23
18.6.7 MMC/SD Block Length Register 18-24
18.6.8 MMC/SD Number of Blocks Register 18-25
18.6.9 MMC/SD Revision Number Register 18-26
18.6.10 MMC/SD Interrupt Mask Register 18-27
18.6.11 Commands and Arguments 18-29
18.6.11.1 MMC/SD Command Number Register 18-30
18.6.11.2 MMC/SD Higher Argument Register 18-30
18.6.11.3 MMC/SD Lower Argument Register 18-31
18.6.12 MMC/SD Response FIFO Register 18-32
18.6.13 MMC/SD Buffer Access Register 18-33
18.7 Functional Example for the MMC/SD Module 18-33
18.7.1 Basic Operation 18-34
18.7.2 Card Identification State 18-34
18.7.2.1 Card Detect 18-34
18.7.2.2 Reset 18-35
18.7.2.3 Voltage Validation 18-35
18.7.2.4 Card Registry 18-38
18.7.3 Card Access 18-39
18.7.3.1 Block Access: Block Write and Block Read 18-39
18.7.3.1.1 Block Write 18-39
18.7.3.1.2 Block Read 18-42
18.7.3.2 Stream Access—Stream Write and Stream Read (MMC Only) 18-45
18.7.3.2.1 Stream Write 18-45
18.7.3.2.2 Stream Read 18-46
18.7.3.3 Erase—Group Erase (MMC Only) and Sector Erase 18-47
18.7.3.4 Wide Bus Selection or Deselection 18-48
18.7.4 Protection Management 18-48
18.7.4.1 Card Internal Write Protection 18-49
18.7.4.2 Mechanical Write Protect Switch 18-49
18.7.4.3 Password Protect 18-49
18.7.4.3.1 Setting the Password 18-50
18.7.4.3.2 Resetting the Password 18-50
18.7.4.3.3 Locking a Card 18-51
18.7.4.3.4 Unlocking the Card 18-51
18.7.4.3.5 Forcing Erase 18-52
18.7.5 Card Status Register 18-52
18.7.6 SD Status Register 18-54
18.7.7 SD I/O 18-55
18.7.7.1 SD I/O Interrupts 18-55
18.7.7.2 SD I/O Suspend and Resume 18-56
18.7.7.3 SD I/O ReadWait 18-56
18.7.8 Commands and Responses 18-57

18.7.8.1 Application-Specific and General Commands 18-58
18.7.8.2 Command Types 18-58
18.7.8.3 Command Formats 18-59
18.7.8.4 Commands for the MMC/SD Module 18-59
18.7.8.5 Response Formats 18-63
18.7.8.5.1 R1—Normal Response 18-63
18.7.8.5.2 R1b—Normal Response with Busy 18-64
18.7.8.5.3 R2—CID, CSD Register 18-64
18.7.8.5.4 R3—OCR Register 18-64
18.7.8.5.5 R4—Fast I/O for MMC Only 18-65
18.7.8.5.6 R4b—SD I/O Only 18-65
18.7.8.5.7 R5—Interrupt Request (for MMC Only) 18-65
18.7.8.5.8 R6—SD I/O Only 18-65

Chapter 19
Memory Stick Host Controller (MSHC) Module

19.1 Overview 19-1
19.2 Features 19-1
19.3 Block Diagram and Description 19-1
19.4 Memory Stick Interface 19-2
19.4.1 Signal Description 19-3
19.4.2 Pin Configuration for the MSHC Module 19-3
19.5 Memory Stick Host Controller Operation 19-4
19.5.1 Data FIFO Operation 19-4
19.5.2 Bus State Control Operation 19-5
19.5.3 MSHC Module Interrupt Operation 19-5
19.5.3.1 Interrupt Sources 19-5
19.5.3.2 SDIO Interrupt Operation 19-6
19.5.4 Reset Operation 19-7
19.5.5 Power Save Mode Operation 19-8
19.5.5.1 Register Access During Power Save Mode 19-9
19.5.5.2 Register Access while MSHC Module is Disabled 19-9
19.5.6 Auto Command Function 19-9
19.5.7 Serial Clock Divider Operation 19-11
19.5.8 System-Level DMA Transfer Operation 19-11
19.6 Programming Model 19-12
19.7 Memory Stick Command Register 19-13
19.7.1 Memory Stick Control/Status Register 19-14
19.7.2 Memory Stick Transmit FIFO Data Register 19-15
19.7.3 Memory Stick Receive FIFO Data Register 19-16
19.7.4 Memory Stick Interrupt Control/Status Register 19-16
19.7.5 Memory Stick Parallel Port Control/Data Register 19-18
19.7.6 Memory Stick Control 2 Register 19-19
19.7.7 Memory Stick Auto Command Register 19-20

19.7.8 Memory Stick FIFO Access Error Control/Status Register 19-21
19.7.9 Memory Stick Serial Clock Divider Register 19-22
19.7.10 Memory Stick DMA Request Control Register 19-23
19.8 Programmer's Reference 19-24
19.8.1 Memory Stick Serial Interface Overview 19-24
19.8.2 Protocol 19-26
19.8.2.1 Write Packet 19-26
19.8.2.2 Read Packet 19-27
19.8.3 Transfer Protocol Command (TPC) 19-27
19.8.4 Protocol Error 19-28
19.8.4.1 Overview 19-28
19.8.4.2 Two State Access Mode Factor 19-30
19.8.5 Signal Timing 19-31
19.8.5.1 Timing 19-31
19.8.5.2 Bus State Extension 19-31
19.8.5.3 Data Transfer Extension 19-32

Chapter 20
Pulse-Width Modulator (PWM)

20.1 Introduction 20-1
20.2 PWM Signals 20-1
20.2.1 Clock Signals 20-1
20.2.2 Pin Configuration for PWM 20-2
20.3 PWM Operation 20-2
20.3.1 Playback Mode 20-2
20.3.2 Tone Mode 20-3
20.3.3 Digital-to-Analog Converter (D/A) Mode 20-3
20.4 Programming Model 20-3
20.4.1 PWM Control Register 20-3
20.4.1.1 HCTR and BCTR Bit Description 20-6
20.4.2 PWM Sample Register 20-6
20.4.3 PWM Period Register 20-7
20.4.4 PWM Counter Register 20-8

Chapter 21
Real-Time Clock (RTC)

21.1 Operation 21-2
21.1.1 Prescaler and Counter 21-2
21.1.2 Alarm 21-2
21.1.3 Sampling Timer 21-2
21.1.4 Minute Stopwatch 21-3
21.2 Programming Model 21-3
21.2.1 RTC Days Counter Register 21-4
21.2.2 RTC Hours and Minutes Counter Register 21-5

21.2.3 RTC Seconds Counter Register 21-6
21.2.4 RTC Day Alarm Register 21-7
21.2.5 RTC Hours and Minutes Alarm Register 21-8
21.2.6 RTC Seconds Alarm Register 21-9
21.2.7 RTC Control Register 21-10
21.2.8 RTC Interrupt Status Register 21-10
21.2.9 RTC Interrupt Enable Register 21-13
21.2.10 Stopwatch Minutes Register 21-15

Chapter 22
SDRAM Memory Controller

22.1 Features 22-1
22.2 Block Diagram 22-2
22.3 Functional Overview 22-2
22.3.1 SDRAM Command Controller 22-3
22.3.2 Page and Bank Address Comparators 22-3
22.3.3 Row and Column Address Multiplexer 22-3
22.3.4 Data Aligner and Multiplexer 22-3
22.3.5 Configuration Registers 22-3
22.3.6 Refresh Request Counter 22-3
22.3.7 Powerdown Timer 22-4
22.3.8 DMA Operation with the SDRAM Controller 22-4
22.3.9 SDCLK—SDRAM Clock 22-5
22.3.10 SDCKE0, SDCKE1—SDRAM Clock Enables 22-5
22.3.11 CSD0, CSD1—SDRAM Chip-Select 22-5
22.3.12 DQ [31:0]—Data Bus (Internal) 22-5
22.3.13 MA [11:0]—Multiplexed Address Bus 22-6
22.3.14 SDBA [4:0], SDIBA [3:0]—Non-Multiplexed Address Bus 22-6
22.3.15 DQM3, DQM2, DQM1, DQM0—Data Qualifier Mask 22-6
22.3.16 SDWE—Write Enable 22-6
22.3.17 RAS—Row Address Strobe 22-6
22.3.18 CAS—Column Address Strobe 22-6
22.3.19 RESET_SF—Reset or Powerdown 22-7
22.3.20 Pin Configuration for SDRAMC 22-7
22.4 Programming Model 22-8
22.4.1 SDRAM Control Registers 22-9
22.4.2 SDRAM Reset Register 22-16
22.4.3 Miscellaneous Register 22-17
22.5 Operating Modes 22-18
22.5.1 SDRAM and SyncFlash Command Encoding 22-18
22.5.2 Normal Read/Write Mode (SMODE = 000) 22-19
22.5.3 Precharge Command Mode (SMODE = 001) 22-24
22.5.4 Auto-Refresh Mode (SMODE = 010) 22-25
22.5.5 Set Mode Register Mode (SMODE = 011) 22-26

22.5.6 SyncFlash Load Command Mode . . . 22-27
22.5.7 SyncFlash Program Mode . . . 22-28
22.6 General Operation . . . 22-30
22.6.1 Address Multiplexing . . . 22-30
22.6.1.1 Multiplexed Address Bus . . . 22-30
22.6.1.2 Non-Multiplexed Address Bus . . . 22-32
22.6.1.3 Bank Addresses . . . 22-33
22.6.2 Refresh . . . 22-33
22.6.3 Self-Refresh . . . 22-34
22.6.3.1 Self-Refresh During RESET_IN . . . 22-34
22.6.3.2 Self-Refresh During Low-Power Mode . . . 22-35
22.6.3.3 Powerdown Operation During Reset and Low-Power Modes . . . 22-35
22.6.4 Clock Suspend Low-Power Mode . . . 22-37
22.6.4.1 Powerdown (Precharge Powerdown) . . . 22-37
22.6.4.2 Clock Suspend (Active Powerdown) . . . 22-37
22.6.4.3 Refresh During Powerdown or Clock Suspend . . . 22-37
22.7 SDRAM Operation . . . 22-39
22.7.1 SDRAM Selection . . . 22-39
22.7.2 Configuring Controller for SDRAM Memory Array . . . 22-40
22.7.2.1 CAS Latency . . . 22-40
22.7.2.2 Row Precharge Delay . . . 22-40
22.7.2.3 Row-to-Column Delay . . . 22-40
22.7.2.4 Row Cycle Delay . . . 22-40
22.7.2.5 Refresh Rate . . . 22-41
22.7.2.6 Memory Configuration Examples . . . 22-41
22.7.3 SDRAM Reset Initialization . . . 22-59
22.7.4 Mode Register Programming . . . 22-61
22.7.5 Mode Register Programming Examples . . . 22-64
22.7.5.1 Example 1—256 Mbit SDRAM Mode Register . . . 22-64
22.7.5.2 Example 2—64 Mbit SDRAM Mode Register . . . 22-66
22.7.6 SDRAM Memory Refresh . . . 22-69
22.8 SyncFlash Operation . . . 22-69
22.8.1 SyncFlash Reset Initialization . . . 22-69
22.8.2 SyncFlash Mode Register Programming . . . 22-70
22.8.3 Booting From SyncFlash . . . 22-70
22.8.4 SyncFlash Configuration . . . 22-70
22.8.5 SyncFlash Programming . . . 22-73
22.8.6 Clock Suspend Timer Use with SyncFlash . . . 22-74
22.8.7 Powerdown Operation with SyncFlash . . . 22-74
22.9 Deep Powerdown Operation with SyncFlash . . . 22-74

Chapter 23
General-Purpose Timers

23.1 Operation . . . 23-2

23.1.1 Pin Configuration for General-Purpose Timers . . . 23-2
23.2 Programming Model . . . 23-3
23.2.1 Timer Control Registers 1 and 2 . . . 23-3
23.2.2 Timer Prescaler Registers 1 and 2 . . . 23-5
23.2.3 Timer Compare Registers 1 and 2 . . . 23-6
23.2.4 Timer Capture Registers 1 and 2 . . . 23-7
23.2.5 Timer Counter Registers 1 and 2 . . . 23-8
23.2.6 Timer Status Registers 1 and 2 . . . 23-9

Chapter 24
USB Device Port

24.1 Introduction . . . 24-1
24.1.1 Features . . . 24-1
24.2 Module Components . . . 24-3
24.2.1 Universal Serial Bus Device Controller . . . 24-3
24.2.2 Synchronization and Transaction Decode . . . 24-4
24.2.3 Endpoint FIFO Architecture . . . 24-4
24.2.4 Control Logic . . . 24-5
24.2.5 USB Transceiver Interface . . . 24-5
24.2.6 Signal Description . . . 24-5
24.2.7 Pin Configuration for USB . . . 24-6
24.3 Programming Model . . . 24-7
24.3.1 USB Frame Register . . . 24-8
24.3.2 USB Specification Register . . . 24-9
24.3.3 USB Status Register . . . 24-10
24.3.4 USB Control Register . . . 24-11
24.3.5 USB Descriptor RAM Address Register . . . 24-13
24.3.6 USB Descriptor RAM/Endpoint Buffer Data Register . . . 24-14
24.3.7 USB Interrupt Register . . . 24-15
24.3.8 USB Interrupt Mask Register . . . 24-17
24.3.9 USB Enable Register . . . 24-18
24.3.10 Endpoint n Status/Control Registers . . . 24-19
24.3.11 Endpoint n Interrupt Status Registers . . . 24-20
24.3.12 Endpoint n Interrupt Mask Register . . . 24-23
24.3.13 Endpoint n FIFO Data Registers . . . 24-25
24.3.14 Endpoint n FIFO Status Registers . . . 24-26
24.3.15 Endpoint n FIFO Control Registers . . . 24-28
24.3.16 USB Endpoint n Last Read Frame Pointer Registers . . . 24-30
24.3.17 USB Endpoint n Last Write Frame Pointer Registers . . . 24-31
24.3.18 Endpoint n FIFO Alarm Registers . . . 24-32
24.3.19 Endpoint n FIFO Read Pointer Registers . . . 24-33
24.3.20 Endpoint n FIFO Write Pointer Registers . . . 24-34
24.4 Programmer’s Reference . . . 24-34
24.5 Device Initialization . . . 24-35

24.5.1 Configuration Download 24-36
24.5.1.1 USB Endpoint to FIFO Mapping 24-37
24.5.1.2 USB Interrupt 24-37
24.5.1.3 Endpoint Registers 24-37
24.5.1.4 Enable the Device 24-38
24.6 Exception Handling 24-38
24.6.1 Unable to Complete Device Request 24-38
24.6.2 Aborted Device Request 24-38
24.6.3 Unable to Fill or Empty FIFO Due to Temporary Problem 24-39
24.6.4 Catastrophic Error 24-39
24.7 Data Transfer Operations 24-39
24.7.1 USB Packets 24-39
24.7.1.1 Short Packets 24-39
24.7.1.2 Sending Packets 24-40
24.7.1.3 Receiving Packets 24-40
24.7.1.4 Programming the FIFO Controller 24-41
24.7.2 USB Transfers 24-41
24.7.2.1 Data Transfers to the Host 24-41
24.7.2.2 Data Transfers to the Device 24-42
24.7.3 Control Transfers 24-42
24.7.4 Bulk Traffic 24-42
24.7.4.1 Bulk OUT 24-42
24.7.4.2 Bulk IN 24-43
24.7.5 Interrupt Traffic 24-43
24.7.6 Isochronous Operations 24-43
24.7.6.1 Isochronous Transfers in a Nutshell 24-43
24.7.6.2 The SYNCH_FRAME Standard Request 24-44
24.8 Interrupt Services 24-44
24.8.1 USB General Interrupts 24-44
24.8.1.1 MSOF—Missed Start-of-Frame 24-44
24.8.1.2 SOF—Start-of-Frame 24-44
24.8.1.3 RESET_STOP—End of USB Reset Signaling 24-44
24.8.1.4 RESET_START—Start of USB Reset Signaling 24-45
24.8.1.5 WAKEUP—Resume (Wakeup) Signaling Detected 24-45
24.8.1.6 SUSP—USB Suspended 24-45
24.8.1.7 FRAME_MATCH—Match Detected in USB_FRAME Register 24-45
24.8.1.8 CFG_CHG—Host Changed USB Device Configuration 24-45
24.8.2 Endpoint Interrupts 24-45
24.8.2.1 FIFO_FULL 24-45
24.8.2.2 FIFO_EMPTY 24-46
24.8.2.3 FIFO_ERROR 24-46
24.8.2.4 FIFO_HIGH 24-46
24.8.2.5 FIFO_LOW 24-46
24.8.2.6 EOT—End of Transfer 24-46
24.8.2.7 DEVREQ—Device Request 24-46

24.8.2.8 MDEVREQ—Multiple Device Request . . . 24-46
24.8.2.9 EOF—End of Frame . . . 24-47
24.8.3 Interrupts, Missed Interrupts, and the USB . . . 24-47
24.8.3.1 SOF . . . 24-47
24.8.3.2 CFG_CHG . . . 24-47
24.8.3.3 EOT . . . 24-47
24.8.3.4 DEVREQ . . . 24-47
24.9 Reset Operation . . . 24-47
24.9.1 Hard Reset . . . 24-48
24.9.2 USB Software Reset . . . 24-48
24.9.3 UDC Reset . . . 24-48
24.9.4 USB Reset Signaling . . . 24-48

Chapter 25
Universal Asynchronous Receiver/Transmitters (UART) Modules

25.1 Introduction . . . 25-1
25.2 Features . . . 25-1
25.2.1 Module Interface . . . 25-2
25.2.2 UART Pin Configuration . . . 25-3
25.3 Interrupts and DMA Requests . . . 25-4
25.4 General UART Definitions . . . 25-5
25.4.1 $\overline{\text{RTS}}$—UART Request To Send . . . 25-6
25.4.2 $\overline{\text{RTS}}$ Edge Triggered Interrupt . . . 25-6
25.4.3 DTR—Data Terminal Ready . . . 25-6
25.4.4 DTR Edge Triggered Interrupt . . . 25-7
25.4.5 DSR—Data Set Ready . . . 25-7
25.4.6 DCD—Data Carrier Detect . . . 25-7
25.4.7 RI—Ring Indicator . . . 25-7
25.4.8 CTS—Clear To Send . . . 25-8
25.4.9 Programmable CTS Deassertion . . . 25-8
25.4.10 TXD—UART Transmit . . . 25-8
25.4.11 RXD—UART Receive . . . 25-8
25.5 Sub-Block Description . . . 25-9
25.5.1 Transmitter . . . 25-10
25.5.2 Transmitter FIFO Empty Interrupt Suppression . . . 25-11
25.5.3 Receiver . . . 25-12
25.5.4 Idle Line Detect . . . 25-13
25.5.4.1 Idle Condition Detect Configuration . . . 25-13
25.5.5 Receiver Wake . . . 25-14
25.5.6 Receiving a BREAK Condition . . . 25-14
25.5.7 Vote Logic . . . 25-14
25.5.8 Binary Rate Multiplier (BRM) . . . 25-15
25.5.9 Baud Rate Automatic Detection Logic . . . 25-17
25.5.9.1 Baud Rate Automatic Detection Protocol . . . 25-18

25.5.10 Escape Sequence Detection . . . 25-19
25.6 Infrared Interface . . . 25-20
25.7 Programming Model . . . 25-20
25.7.1 UART Receiver Registers . . . 25-23
25.7.2 UART Transmitter Registers . . . 25-25
25.7.3 UART Control Register 1 . . . 25-26
25.7.4 UART Control Register 2 . . . 25-29
25.7.5 UART Control Register 3 . . . 25-32
25.7.5.1 UART1 Control Register 3 . . . 25-32
25.7.5.2 UART2 Control Register 3 . . . 25-34
25.7.6 UART Control Register 4 . . . 25-36
25.7.7 UART FIFO Control Registers . . . 25-38
25.7.8 UART Status Register 1 . . . 25-40
25.7.9 UART Status Register 2 . . . 25-42
25.7.10 UART Escape Character Registers . . . 25-44
25.7.11 UART Escape Timer Registers . . . 25-45
25.7.12 UART BRM Incremental Registers . . . 25-46
25.7.13 UART BRM Modulator Registers . . . 25-47
25.7.14 UART Baud Rate Count Registers . . . 25-48
25.7.15 UART BRM Incremental Preset Registers 1–4 . . . 25-49
25.7.16 UART BRM Modulator Preset Registers 1–4 . . . 25-50
25.7.17 UART Test Register 1 . . . 25-51
25.8 UART Operation in Low-Power System States . . . 25-52

Chapter 26
I^{2}C Module

26.1 Overview . . . 26-1
26.2 Interface Features . . . 26-1
26.3 I^{2}C System Configuration . . . 26-2
26.4 I^{2}C Protocol . . . 26-3
26.4.1 Clock Synchronization . . . 26-4
26.4.2 Arbitration Procedure . . . 26-5
26.4.3 Handshaking . . . 26-5
26.4.4 Clock Stretching . . . 26-5
26.5 Pin Configuration for I^{2}C . . . 26-5
26.6 Programming Model . . . 26-6
26.6.1 I^{2}C Address Register . . . 26-7
26.6.2 I^{2}C Frequency Divider Register . . . 26-8
26.6.3 I^{2}C Control Register . . . 26-10
26.6.4 I^{2}C Status Register . . . 26-11
26.6.5 I^{2}C Data I/O Register . . . 26-13
26.7 I^{2}C Programming Examples . . . 26-13
26.7.1 Initialization Sequence . . . 26-14
26.7.2 Generation of START . . . 26-14

26.7.3 Post-Transfer Software Response . . . 26-14
26.7.4 Generation of STOP . . . 26-15
26.7.5 Generation of Repeated START . . . 26-15
26.7.6 Slave Mode . . . 26-15
26.7.7 Arbitration Lost . . . 26-15

Chapter 27
Synchronous Serial Interface (SSI)

27.1 Introduction . . . 27-1
27.2 SSI Architecture . . . 27-1
27.2.1 SSI Clocking . . . 27-4
27.2.1.1 Normal Operating Mode . . . 27-4
27.2.1.2 Master / Synchronous Mode . . . 27-4
27.2.2 SSI Clock and Frame Sync Generation . . . 27-4
27.2.3 Pin Configuration for SSI . . . 27-5
27.2.3.1 Pin Configuration Example Software . . . 27-7
27.3 Programming Model . . . 27-7
27.3.1 SSI Transmit Data Register . . . 27-8
27.3.2 SSI Transmit FIFO Register . . . 27-9
27.3.3 SSITransmit Shift Register . . . 27-9
27.3.4 SSI Receive Data Register . . . 27-12
27.3.5 SSI Receive FIFO Register . . . 27-12
27.3.6 SSI Receive Shift Register . . . 27-13
27.3.7 SSI Control/Status Register . . . 27-15
27.3.7.1 I2S Mode Selection . . . 27-19
27.3.8 SSI Transmit Configuration Register . . . 27-21
27.3.9 SSI Receive Configuration Register . . . 27-23
27.3.10 SSI Transmit Clock Control Register and SSI Receive Clock Control Register . . . 27-27
27.3.10.1 Calculating the SSI Bit Clock from the Input Clock Value . . . 27-28
27.3.11 SSI Time Slot Register . . . 27-30
27.3.12 SSI FIFO Control/Status Register . . . 27-31
27.3.13 SSI Option Register . . . 27-34
27.4 SSI Data and Control Pins . . . 27-35
27.4.1 SSI_TXDAT, Serial Transmit Data . . . 27-35
27.4.2 SSI_RXDAT, Serial Receive Data . . . 27-35
27.4.3 SSI_TXCLK, Serial Transmit Clock . . . 27-35
27.4.4 SSI_RXCLK, Serial Receive Clock . . . 27-35
27.4.5 SSI_TXFS, Serial Transmit Frame Sync . . . 27-36
27.4.6 SSI_RXFS, Serial Receive Frame Sync . . . 27-36
27.5 SSI Operating Modes . . . 27-38
27.5.1 Normal Mode . . . 27-39
27.5.1.1 Normal Mode Transmit . . . 27-39
27.5.1.2 Normal Mode Receive . . . 27-39
27.5.2 Network Mode . . . 27-41

27.5.2.1 Network Mode Transmit . . . 27-41
27.5.2.2 Network Mode Receive . . . 27-42
27.6 Gated Clock Mode . . . 27-43
27.7 External Frame and Clock Operation . . . 27-44
27.8 SSI Reset and Initialization Procedure . . . 27-44

Chapter 28
CMOS Sensor Interface Module

28.1 Introduction . . . 28-1
28.2 CSI Module Architecture . . . 28-1
28.3 CSI Module Interface Signal Description . . . 28-2
28.3.1 Pin Configuration for CSI . . . 28-3
28.4 CSI Module Operation . . . 28-3
28.4.1 Data FIFO Operation . . . 28-3
28.4.2 CSI Interrupt Operation . . . 28-4
28.4.3 Register Access While CSI Module is Disabled . . . 28-4
28.5 Programming Model . . . 28-4
28.5.1 CSI Control Register 1 . . . 28-5
28.5.2 CSI Control Register 2 . . . 28-8
28.5.3 CSI Status Register 1 . . . 28-10
28.5.4 CSI Statistic FIFO Register 1 . . . 28-11
28.5.5 CSI RxFIFO Register 1 . . . 28-12
28.6 Statistic Data Generation . . . 28-13
28.6.1 Statistic Block Diagram and Description . . . 28-13
28.6.2 Auto Exposure and Auto White Balance . . . 28-13
28.6.3 Auto Focus . . . 28-15
28.6.4 Packing of Statistic Data . . . 28-15
28.6.5 Sensor Interface Signals . . . 28-16
28.6.6 Statistic Control Signals . . . 28-16
28.6.6.1 Start of Frame . . . 28-16
28.6.6.2 Auto Focus Spread . . . 28-16
28.6.7 Statistic Output and DMA Signals . . . 28-16
28.6.7.1 Statistic Data Out . . . 28-16
28.6.7.2 Statistic FIFO Full . . . 28-16
28.6.7.3 Statistic Data Request . . . 28-17

Chapter 29
GPIO Module and I/O Multiplexer (IOMUX)

29.1 General Description . . . 29-1
29.2 GPIO Module Overview . . . 29-2
29.2.1 GPIO Module Features . . . 29-2
29.2.2 Interrupts . . . 29-3
29.2.3 GPIO Signal Description . . . 29-3
29.3 GPIO Module Block Diagram . . . 29-4

29.4 Pin Configuration for GPIO 29-4
29.5 Programming Model 29-7
29.5.1 Data Direction Registers 29-9
29.5.2 Output Configuration Registers 29-10
29.5.2.1 Output Configuration Register 1 29-10
29.5.2.2 Output Configuration Register 2 29-11
29.5.3 Input Configuration Registers 29-12
29.5.3.1 Input Configuration Register A1 29-12
29.5.3.2 Input Configuration Register A2 29-13
29.5.3.3 Input Configuration Register B1 29-14
29.5.3.4 Input Configuration Register B2 29-15
29.5.4 Data Registers 29-16
29.5.5 GPIO In Use Registers 29-17
29.5.6 Sample Status Registers 29-18
29.5.7 Interrupt Configuration Registers 29-19
29.5.7.1 Interrupt Configuration Register 1 29-19
29.5.7.2 Interrupt Configuration Register 2 29-20
29.5.8 Interrupt Mask Registers 29-21
29.5.9 Interrupt Status Registers 29-22
29.5.10 General Purpose Registers 29-23
29.5.11 Software Reset Registers 29-24
29.5.12 Pull_Up Enable Registers 29-25

About This Book

This reference manual describes the features and operation of the MC9328MXL (i.MXL) microprocessor, It provides the details of how to initialize, configure, and program the MC9328MXL. The manual presumes basic knowledge of ARM920T™ architecture.

Audience

The MC9328MXL Reference Manual is intended to provide a design engineer with the necessary data to successfully integrate the MC9328MXL into a wide variety of applications. It is assumed that the reader has a good working knowledge of the ARM920T processor. For programming information about the ARM920T processor, see the documents listed in the Suggested Reading section of this preface.

Organization

The MC9328MXL Reference Manual is organized into 29 chapters that cover the operation and programming of the i.MXL device. Summaries of the chapters follow.

Chapter 1	**Introduction:** This chapter contains a device feature list, overview of system modules, and system block diagrams.
Chapter 2	**Signal Descriptions and Pin Assignments:** This chapter's content has been moved to the MC9328MXL Data Sheet.
Chapter 3	**Memory Map:** This chapter summarizes the memory organization, programming information and a listing of all of the registers in the MC9328MXL.
Chapter 4	**ARM920T Processor:** This chapter provides a high-level overview of the ARM920T processor including the ARM9 Thumb® instruction set.
Chapter 5	**Embedded Trace Macrocell (ETM):** This chapter provides a summary of the operation and features of the ARM Embedded Trace Macrocell™.
Chapter 6	**Reset Module:** The reset module processes of all of the system reset signals required by the MC9328MXL. This chapter gives a detailed description of the reset module and associated timing and signals.
Chapter 7	**AHB to IP Bus Interface (AIPI)**: This chapter provides an overview of the R-AHB to IP bus interface. The AIPI module in the MC9328MXL acts as an interface between the R-AHB (Reduced ARM Advanced High-performance Bus) and lower bandwidth peripherals.
Chapter 8	**System Control:** This chapter describes the operation of and programming models for the system multiplex control, peripheral control, ID register, and I/O drive control registers.
Chapter 9	**Bootstrap Mode Operation:** The operation of bootstrap models is described in detail in this chapter. This chapter describes programming information necessary to allow a system to initialize a target system and download a program or data to the target system's RAM using the UART controller.

Chapter 10	**Interrupt Controller (AITC):** This chapter provides a description and operational considerations for interrupt controller operation to perform interrupt masking, priority support, and hardware acceleration of normal interrupts.
Chapter 11	**External Interface Module (EIM):** This chapter describes the external interface module and shows how the module handles the interface to devices external to the MC9328MXL, including generation of chip selects for external peripherals and memory.
Chapter 12	**Phase-Locked Loop and Clock Controller:** This chapter provides detailed information about the operation and programming of the clock generation module as well as the recommended circuit schematics for external clock circuits. It also describes and provides programming information about the operation of the power control module and the system power states.
Chapter 13	**DMA Controller (DMAC):** This chapter describes the operation of the direct memory access controller contained in the MC9328MXL. The DMA controller provides two memory channels and four I/O channels to support a wide variety of DMA operations.
Chapter 14	**Watchdog Timer Module:** The operation of the watchdog timer module is described in this chapter. It includes information of how the watchdog timer protects against system failures by providing a method of escaping from unexpected events or programming errors.
Chapter 15	**Multimedia Accelerator (MMA):** This chapter describes the operation of the MMA which is used in conjunction with the ARM920T processor to perform the iterative operations of a digital signal processor for applications such as MPEG or MP3 encoding/decoding and speech compression/decompression.
Chapter 16	**Serial Peripheral Interface Modules (SPI 1 and SPI 2)**: The programming and operation of the two identical serial peripheral interface modules (SPI 1 and SPI 2) is described in this chapter.
Chapter 17	**LCD Controller (LCDC):** This chapter describes the operation and programming of the liquid crystal display controller, which provides display data for external LCD drivers or for an LCD panel.
Chapter 18	**Multimedia Card/Secure Digital Host Controller (MMC/SD):** This chapter describes the Multimedia Card (MMC) host controller which controls Flash-based mass storage products. This chapter also describes the Secure Digital feature of the MMC, its operation and programming information.
Chapter 19	**Memory Stick Host Controller (MSHC):** This chapter describes how data is transferred to a Memory Stick device. It also discusses how to configure and program the Memory Stick Host Controller.
Chapter 20	**Pulse-Width Modulator (PWM):** This chapter describes the operation and configuration of the pulse-width modulator. Programming information is also provided.
Chapter 21	**Real-Time Clock (RTC):** This chapter describes the operation of the real-time clock module, which is composed of a prescaler, time-of-day (TOD) clock, TOD alarm, programmable real-time interrupt, watchdog timer, and minute stopwatch as well as control registers and bus interface hardware.
Chapter 22	**SDRAM Memory Controller (SDRAMC):** The operation and programming of the SDRAM controller is described in this chapter. This module provides a glueless interface to 16-bit or 32-bit synchronous DRAM.

Chapter 23	**General-Purpose Timers:** This chapter describes the two 16-bit timers that can be used as both watchdogs and alarms.
Chapter 24	**Universal Asynchronous Receiver/Transmitters (UART):** This chapter describes the capabilities and operation of the three UARTs. It also discusses how to configure and program the UART modules.
Chapter 25	**USB Device Port:** This chapter provides configuration, interface description and detailed programming information for designers to achieve the optimum performance from the USB device.
Chapter 26	**I^2C Module:** This chapter describes the I^2C module of the MC9328MXL including I^2C protocol, clock synchronization, and the registers in the I^2C programming mode.
Chapter 27	**Synchronous Serial Interface (SSI):** This chapter presents the two Synchronous Serial Interface modules and discusses the architecture, programming model, operating modes, and initialization of the SSI.
Chapter 28	**CMOS Sensor Interface (CSI):** The CSI module is a logic interface that enables the i.MX to connect directly to external CMOS image sensors. This chapter describes the CSI module, and discusses the architecture, the programming model, and the software initialization sequence.
Chapter 29	**GPIO and I/O Multiplexer (IOMUX):** This chapter covers all GPIO lines found in the MC9328MXL. Because each pin is individually configurable, a detailed description of the operation is provided.

Document Revision History

Table 0-1 includes technical content changes made for this revision.

Table 0-1. Revision History

Location	Description of Change
Chapter 13, “DMA Controller,” Table 13-23 on page 13-30	DMA Request table: DMA_REQ [12] change from “Reserved” to “Ext_DMA”.
Chapter 27, “Synchronous Serial Interface (SSI),” Section 27.7, “External Frame and Clock Operation.”	Clarification to first paragraph.
Chapter 16, “Serial Peripheral Interface Modules (SPI 1 and SPI 2),” Section 16.3.3, “Control Registers.”	BIT_COUNT field description change.

Suggested Reading

The following documents are required for a complete description of the MC9328MXL and are necessary to design properly with the device. Especially for those not familiar with the ARM920T processor or previous DragonBall products, the following documents will be helpful when used in conjunction with this manual.

ARM Architecture Reference Manual (ARM Ltd., order number ARM DDI 0100)

ARM9DT1 Data Sheet Manual (ARM Ltd., order number ARM DDI 0029)

ARM Technical Refines Manual (ARM Ltd., order number ARM DDI 0151C)

EMT9 Technical Reference Manual (ARM Ltd., order number DDI O157E)

MC9328MXL Product Brief (order number MC9328MXLP/D)

MC9328MXL Data Sheet (order number MC9328MXL/D)

The manuals may be found at the Motorola Semiconductors World Wide Web site at http://www.motorola.com/semiconductors. These documents may be downloaded directly from the World Wide Web site, or printed versions may be ordered. The World Wide Web site also may have useful application notes.

Conventions

This reference manual uses the following conventions:

- $\overline{\text{OVERBAR}}$ is used to indicate a signal that is active when pulled low: for example, $\overline{\text{RESET}}$.
- *Logic level one* is a voltage that corresponds to Boolean true (1) state.
- *Logic level zero* is a voltage that corresponds to Boolean false (0) state.
- To *set* a bit or bits means to establish logic level one.
- To *clear* a bit or bits means to establish logic level zero.
- A *signal* is an electronic construct whose state conveys or changes in state convey information.
- A *pin* is an external physical connection. The same pin can be used to connect a number of signals.
- *Asserted* means that a discrete signal is in active logic state.
  - *Active low* signals change from logic level one to logic level zero.
  - *Active high* signals change from logic level zero to logic level one.
- *Negated* means that an asserted discrete signal changes logic state.
  - *Active low* signals change from logic level zero to logic level one.
  - *Active high* signals change from logic level one to logic level zero.
- LSB means *least significant bit* or *bits*, and MSB means *most significant bit* or *bits*. References to low and high bytes or words are spelled out.
- Numbers preceded by a percent sign (%) are binary. Numbers preceded by a dollar sign ($) or *0x* are hexadecimal.

Definitions, Acronyms, and Abbreviations

The following list defines acronyms and abbreviations used in this document.

ADC	analog-to-digital converter
AFE	analog front end
API	application programming interface
BCD	binary coded decimal
BER	bit error ratio
CGM	clock generation module
CMOS	complimentary metal-oxide semiconductor
CRC	cyclic redundancy check
CSIC	complex instruction set computer
DAC	digital-to-analog converter

DDR RAM	double data rate RAM
DMA	direct memory access
DRAM	dynamic random access memory
FEC	forward error correction
FIFO	first in first out
GPIO	general purpose input/output
I/O	Input/Output
ICE	in-circuit emulation
IrDa	infrared
JTAG	joint test action group
MAP	mold array process
MAPBGA	mold array process ball grid array
MIPS	million instructions per second
MMC	multimedia card
PLL	phase locked loop
PWM	pulse-width modulator
RTC	real-time clock
SIM	system integration module
SD	secure digital
SDRAM	synchronous dynamic random access memory
SPI	serial peripheral interface
SRAM	static random access memory
TQFP	thin quad flat pack
UART	universal asynchronous receiver/transmitter
USB	universal serial bus
XTAL	crystal

Chapter 1
Introduction

The i.MX (Media Extensions) series provides a leap in performance with an ARM9™ microprocessor core and highly integrated system functions. i.MX products specifically address the requirements of the personal portable product market by providing intelligent integrated peripherals, an advanced processor core, and power management capabilities.

The MC9328MXL is equipped with an optimized feature set to target low-cost solutions in the portable handheld market. Its internal bus architecture and fast system speed are the same as MC9328MX1. The ARM920TDMI core speed is programmable from 0 to 200 MHz, while system speed is programmable from 0 to 96 MHz.

The MC9328MXL provides the following benefits:

- Features a high level of on-chip integration
- Provides uncompromising performance in a very low-power system design
- Optimized for multimedia applications
- Connectivity features include SPI, UART, USB, and SSI/I^2S
- Supports a wide variety of applications including the most popular PDA designs, smart phones, and next-generation wireless communicators

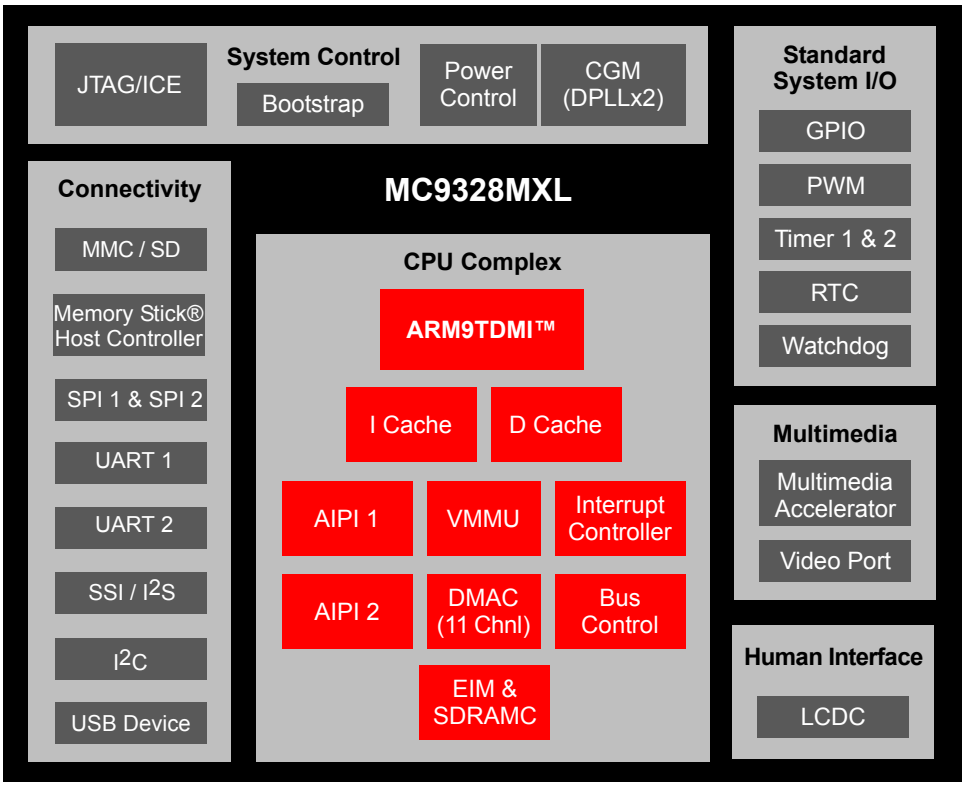


Figure 1-1. MC9328MXL Functional Block Diagram

To support a wide variety of applications, the MC9328MXL boasts a robust array of features, including the following:

- ARM920T Microprocessor Core
- AHB to IP Bus Interfaces (AIPIs)
- External Interface Module (EIM)
- SDRAM Controller (SDRAMC)
- DPLL Clock and Power Control Module
- Two Universal Asynchronous Receiver/Transmitters (UART 1 and UART 2)
- Two Serial Peripheral Interfaces (SPI1 and SPI2)
- Two General-Purpose 32-bit Counters/Timers
- Watchdog Timer
- Real-Time Clock/Sampling Timer (RTC)
- LCD Controller (LCDC)
- Pulse-Width Modulation (PWM) Module
- Universal Serial Bus (USB) Device
- Multimedia Card and Secure Digital (MMC/SD) Host Controller
- Memory Stick® Host Controller (MSHC)
- Direct Memory Access Controller (DMAC)

- Synchronous Serial Interface (SSI1) and Inter-IC Sound (I^2S) Module
- Inter-IC (I^2C) Bus Module
- Video Port
- General-Purpose I/O (GPIO) Ports
- Bootstrap Mode
- Multimedia Accelerator (MMA)
- 256-pin and 225-pin MAPBGA Package

The following sections detail the features of the MC9328MXL's functional blocks.

1.1 ARM920T Microprocessor Core

The MC9328MXL uses the ARM920T microprocessor core which has the following features:

- 200 MHz maximum processing speed
- 16 Kbyte instruction cache and 16 Kbyte data cache
- ARM9 high performance 32-bit RISC engine
- Thumb® 16-bit compressed instruction set for a leading level of code density
- EmbeddedICE™ JTAG software debug
- 100-percent user code binary compatibility with ARM7TDMI® processors
- ARM9TDMI® core, including integrated caches, write buffers, and bus interface units, provides CPU-cache transparency
- Advanced Microcontroller Bus Architecture (AMBA™) system-on-chip multi-master bus interface
- Flexible CPU and bus clocking relationships including asynchronous, synchronous, and single-clock configurations
- Cache locking to support mixed loads of real-time and user applications
- Virtual Memory Management Unit (VMMU)

1.2 AHB to IP Bus Interfaces (AIPIs)

The MC9328MXL AIPIs provide a communication interface between the high-speed AHB bus and a lower-speed IP bus for slow slave peripherals.

1.3 External Interface Module (EIM)

The MC9328MXL EIM features:

- Up to six chip selects for external devices, each with 16 Mbyte of address space (chip selects for ROM support a maximum of 32 Mbyte of address space)
- Programmable protection, port size, and wait states for each chip select
- Internal/external boot ROM selection
- Selectable bus watchdog counter

- Burst support for external AMD™ or Intel® flash with 32-bit data path
- Interrupt controller to handle a maximum of 63 interrupt sources
- Vectored interrupt capability with prioritization for 16 sources
- Supports $\overline{\text{DTACK}}$ function in the $\overline{\text{CS5}}$

1.4 SDRAM Controller (SDRAMC)

The MC9328MXL SDRAMC features:

- Supports 4 banks of 64-, 128-, or 256-Mbit synchronous DRAMs
- Includes 2 independent chip-selects
  - Up to 64 Mbyte per chip-select
  - Up to four banks simultaneously active per chip-select
  - JEDEC standard pinout and operation
- Supports burst reads of word (32-bit) data types
- PC100 compliant interface
  - 100 MHz system clock achievable with “-8” option PC100 compliant memories
  - single and fixed-length (8-word) word access
  - Typical access time of 8-1-1-1 at 100 MHz
- Software configurable bus width, row and column sizes, and delays for differing system requirements
- Built in auto-refresh timer and state machine
- Hardware supported self-refresh entry and exit which keeps data valid during system reset and low-power modes
- Auto-powerdown (clock suspend) timer

1.5 Clock Generation Module (CGM) and Power Control Module

The MC9328MXL CGM and Power Control Module features:

- Digital phase-locked loops (PLLs) and clock controller for all internal clocks generation
- MCUPLL generates FCLK to the CPU from either a 32 kHz or 32.768 kHz
- System PLL generates the system clock and the 48 MHz clock for the USB from a 16 MHz or either a 32 kHz or 32.768 kHz
- Support for three power modes for different power consumption needs: run, doze, and stop

1.6 Two Universal Asynchronous Receiver/Transmitters (UART 1 and UART 2)

The MC9328MXL UARTs features:

- Support for serial data transmit/receive operation: 7 or 8 data bits, 1 or 2 stop bits, and programmable parity (even, odd, or none)

- Programmable baud rates up to 1.00 MHz
- 32-byte FIFO on Tx and 32 half-word FIFO on Rx that support autobaud
- IrDA 1.0 support

1.7 Two Serial Peripheral Interfaces (SPI)

The MC9328MXL SPIs feature:

- SPI 1 is master/slave configurable, SPI 2 is master only
- Up to 16-bit programmable data transfer
- 8 × 16 FIFO for both Tx and Rx data

1.8 Two General-Purpose 32-Bit Counters/Timers

The MC9328MXL General-Purpose Counters/Timers features:

- Automatic interrupt generation
- Programmable timer input/output pins
- Input capture capability with programmable trigger edge
- Output compare with programmable mode

1.9 Watchdog Timer

The MC9328MXL Watchdog Timer features:

- Programmable time out of 0.5 s to 64 s
- Resolution of 0.5 s

1.10 Real-Time Clock/Sampling Timer (RTC)

The MC9328MXL RTC features:

- 32.768 kHz or 32 kHz
- Full clock features: seconds, minutes, hours, and days
- Capable of counting up to 512 days
- Minute countdown timer with interrupt
- Programmable daily alarm with interrupt
- Sampling timer with interrupt
- Once-per-second, once-per-minute, once-per-hour, and once-per-day interrupts
- Interrupt generation for digitizer sampling or keyboard debouncing

1.11 LCD Controller (LCDC)

The MC9328MXL LCDC features:

- Software programmable screen size (a maximum of 640 × 512 pixels) to support single (non-split) monochrome, color STN panels, and color TFT panels
- Support for 4 bpp (bits per pixel), 8 bpp, and 12 bpp for passive color panels
- Support for 4 bpp, 8 bpp, 12 bpp, and 16 bpp for TFT panels
  - Up to 256 colors out of a palette of 4096 for 8 bpp
  - True 64K color for 16 bpp
- In color STN mode, the maximum bit depth is 12 bpp
- In BW mode, the maximum bit depth is 4 bpp
- Up to 16 grey levels out of 16 palettes
- Capable of directly driving popular LCD drivers from manufacturers including Motorola, Sharp, Hitachi, and Toshiba
- Support for data bus width for 12- or 16-bit TFT panels
- Panel interface of 8-, 4-, and 2-bits, and a 1-bit wide LCD panel data bus for monochrome panels
- Direct interface to Sharp® 320 × 240 HR-TFT panel
- Support for logical operation between color hardware cursor and background
- Uses system memory as display memory
- LCD contrast control using 8-bit PWM
- Support for self-refresh LCD modules
- Hardware panning (soft horizontal scrolling)

1.12 Pulse-Width Modulation (PWM) Module

The MC9328MXL PWM Module features:

- 4 × 16 FIFO to minimize interrupt overhead
- 16-bit resolution
- Sound and melody generation

1.13 Universal Serial Bus (USB) Device

The MC9328MXL USB Device features:

- Compliant with *Universal Serial Bus Specification, revision 1.1*
- Up to six logical endpoints—see Table 1-1 on page 1-7
- Support for isochronous communications pipes
  - Frame match interrupt feature notifies the user when a specific USB frame occurs
  - For DMA access, the maximum packet size for the isochronous endpoint is restricted by the FIFO size of the endpoint
  - For programmed I/O, isochronous data packets range from 0 bytes to 1023 bytes

- Support for control, bulk, and interrupt pipes
  - Packet sizes are limited to 8, 16, 32, or 64 bytes
  - Maximum packet size depends on the FIFO size of the endpoint
- Support (via a register bit) for a remote wake-up feature
- Full-speed (12 MHz) operation
- Operation can be programmed for both bus-powered and self-powered mode

Table 1-1. Endpoint Configurations

Endpoint	Direction	Physical FIFO Size (Bytes)	Endpoint Configuration	Maximum Packet Size (Bytes)
0	IN and OUT	32	Control	32
1–5	IN or OUT	32 or 64[1]	Control, interrupt, bulk, or isochronous	User configurable: 8, 16, 32, or 64 (depending on FIFO size)

1.FIFO1 and FIFO2 are 64 bytes each; FIFO3, FIFO4, and FIFO5 are 32 bytes each.

1.14 Multimedia Card and Secure Digital (MMC/SD) Host Controller

The MMC/SD Host Controller features:

- Compatible with the *MultiMediaCard System Specification* (SPI mode excluded), version 3.1
- Compatible to 1/4 bit with the *SD Memory Card Specification* (SPI mode excluded), version 1.0 and *SD I/O Specification* (SPI mode excluded), version 1.0 with 1 or 4 channel(s)
- Up to ten MMC cards and one SD are supported by standard (maximum data rate with a maximum of ten cards)
- Support for hot swappable operation
- Support for data rates from 20 Mbps to 80 Mbps

1.15 Memory Stick® Host Controller (MSHC)

The MSHC features:

- Integrated 8-byte (4-word) FIFO buffer for transmit and receive
- Integrated CRC circuit
- Support for internal or external serial clock source
- Integrated Serial Clock Divider
- DMA support; DMA request condition is selectable based on FIFO status
- Automatic command execution when an interrupt from the Memory Stick is detected (can be toggled on/off)
- RDY time-out period set by the number of serial clock cycles
- Interrupt output to the ARM920T core when a time-out occurs
- Two integrated general-purpose input pins for detecting Memory Stick insertion/extraction
- 16-bit host bus access (byte access not supported)

1.16 Direct Memory Access Controller (DMAC)

The MC9328MXL DMAC features:

- 11 channels to support linear memory, 2D memory, FIFO, and End-of-Burst Enable FIFO for both source and destination
- Support for 8-, 16-, or 32-bit FIFO port size and memory port size data transfer
- Support for big-endian and little-endian
- Configurable DMA burst length for each channel up to 16 words, 32 half-words, or 64 bytes
- Bus utilization control for a channel that is not triggered by DMA requests
- Bulk data transfer complete or transfer error interrupts provided to interrupt handler (and then to the core)
- DMA burst time-out error terminates the DMA cycle when the burst cannot be completed within a programmed timing period
- Acknowledge signal provided to peripheral after DMA burst is complete

1.17 Synchronous Serial Interface and Inter-IC Sound (SSI/I^2S) Module

The MC9328MXL SSI/I^2S Module features:

- Supports generic SSI interface for external audio chip or interprocessor communication
- Supports Philips standard Inter-IC Sound (I^2S) bus for external digital audio chip interface

1.18 Inter-IC (I^2C) Bus Module

The MC9328MXL I^2C Bus Module features:

- Support for Philips I^2C-bus standard for external digital control
- Support for 3.3 V tolerant devices
- Multiple-master operation
- Software-programmable for 1 of 64 different serial clock frequencies
- Software-selectable acknowledge bit
- Interrupt-driven, byte-by-byte data transfer
- Arbitration-lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation and detection
- Repeated START signal generation
- Acknowledge bit generation and detection
- Bus-busy detection

1.19 Video Port

The video port supports external CMOS sensor video data input.

1.20 General-Purpose I/O (GPIO) Ports

The MC9328MXL GPIO ports feature:

- Interrupt capability
- 97 total I/O pins multiplexed with most dedicated functions for pin efficiency

1.21 Bootstrap Mode

The MC9328MXL Bootstrap Mode features:

- Allows user to initialize system and download program or data to system memory through UART
- Accepts execution command to run program stored in system memory
- Supports memory/register read/write operation of selectable data size of byte, half-word, or word
- Provides a 32-byte instruction buffer for ARM920T core vector table storage, instruction storage and execution

1.22 Multimedia Accelerator (MMA)

The Multimedia Accelerator features:

- MAC for FIR and FFT operation—MP3 applications save 10% to 15% CPU MIPS
- DCT/iDCT hardware accelerator—MPEG4 decode applications save approximately 10% CPU MIPS

1.23 Power Management Features

The MC9328MXL provides the following power management features:

- Programmable clock synthesizer using either a 32 kHz or 32.768 kHz crystal for full frequency control
- Low-power stop capabilities
- Modules that can be individually shut down
- Lowest power mode control

1.24 Operating Voltage Range

The MC9328MXL operating voltages are as follows:

- I/O voltage—1.7 V to 2.0 V or 2.7 V to 3.3 V
- Internal logic voltage—150 MHz: 1.7 V to 1.9 V; 200 MHz: 1.8 to 2.0V

1.25 Packaging

The MC9328MXL features two packages:

- 256-pin MAPBGA 14 mm × 14 mm × 1.3 mm package, with 0.8 mm ball pitch
- 225-contact PBGA 13 mm × 13 mm package, with 0.8 mm ball pitch

Chapter 2
Signal Descriptions and Pin Assignments

2.1 Signal and Pin Information

For information about the MC9328MXL signals and their pin assignments refer to the MC9328MXL data sheet (document order number: MC9328MXL).

Chapter 3
Memory Map

This chapter describes the memory maps and the chip configuration registers of the MC9328MXL.

3.1 Memory Space

The ARM920T microprocessor implements a virtual addressing mechanism. Refer to the ARM920T Memory Management Unit in the ARM9 technical reference manual for more information on this topic.

The ARM920T processor physical memory map can be divided according to the addresses shown in Figure 3-1 on page 3-2.

3.1.1 Memory Map

The base address referred to in each peripheral register address is the address from this table. The exact address description of each of the peripherals is described in each peripheral section.

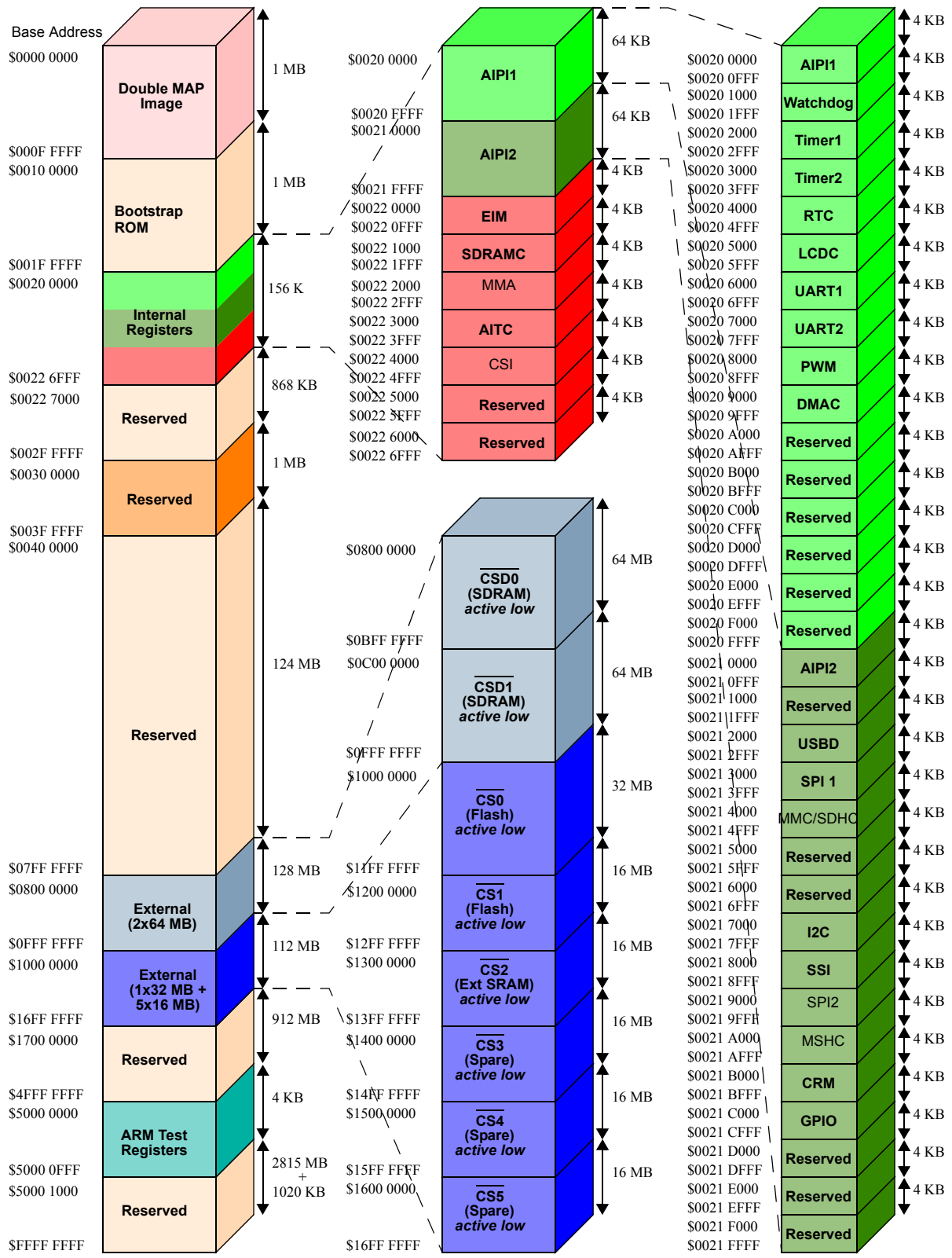


Figure 3-1. MC9328MXL MCU Physical Memory Map (4 Gbyte)

Table 3-1. MCU Memory Space (Physical Addresses)

Address	Description	Size
$0000 0000 - $000F FFFF	Double Map Image	1 Mbyte
$0010 0000 - $001F FFFF	Bootstrap ROM	1 Mbyte
$0020 0000 - $0020 0FFF	AIPI1	4 kbyte
$0020 1000 - $0020 1FFF	WatchDog	4 kbyte
$0020 2000 - $0020 2FFF	TIMER1	4 kbyte
$0020 3000 - $0020 3FFF	TIMER2	4 kbyte
$0020 4000 - $0020 4FFF	RTC	4 kbyte
$0020 5000 - $0020 5FFF	LCDC	4 kbyte
$0020 6000 - $0020 6FFF	UART1	4 kbyte
$0020 7000 - $0020 7FFF	UART2	4 kbyte
$0020 8000 - $0020 8FFF	PWM	4 kbyte
$0020 9000 - $0020 9FFF	DMAC	4 kbyte
$0020 A000 - $0020 AFFF	Reserved	4 kbyte
$0020 B000 - $0020 BFFF	Reserved	4 kbyte
$0020 C000 - $0020 CFFF	Reserved	4 kbyte
$0020 D000 - $0020 DFFF	Reserved	4 kbyte
$0020 E000 - $0020 EFFF	Reserved	4 kbyte
$0020 F000 - $0020 FFFF	Reserved	4 kbyte
$0021 0000 - $0021 0FFF	AIPI2	4 kbyte
$0021 1000 - $0021 1FFF	Reserved	4 kbyte
$0021 2000 - $0021 2FFF	USBD	4 kbyte
$0021 3000 - $0021 3FFF	SPI 1	4 kbyte
$0021 4000 - $0021 4FFF	MMC/SDHC	4 kbyte
$0021 5000 - $0021 5FFF	Reserved	4 kbyte
$0021 6000 - $0021 6FFF	Reserved	4 kbyte
$0021 7000 - $0021 7FFF	I2C	4 kbyte
$0021 8000 - $0021 8FFF	SSI	4 kbyte
$0021 9000 - $0021 9FFF	SPI 2	4 kbyte

Table 3-1. MCU Memory Space (Physical Addresses) (continued)

Address	Description	Size
$0021 A000 - $0021 AFFF	MSHC	4 kbyte
$0021 B000 - $0021 BFFF	CRM	4 kbyte
$0021 C000 - $0021 CFFF	GPIO	4 kbyte
$0021 D000 - $0021 DFFF	Reserved	4 kbyte
$0021 E000 - $0021 EFFF	Reserved	4 kbyte
$0021 F000 - $0021 FFFF	Reserved	4 kbyte
$0022 0000 - $0022 0FFF	EIM	4 kbyte
$0022 1000 - $0022 1FFF	SDRAMC	4 kbyte
$0022 2000 - $0022 2FFF	MMA	4 kbyte
$0022 3000 - $0022 3FFF	AITC	4 kbyte
$0022 4000 - $0022 4FFF	CSI	4 kbyte
$0022 5000 - $0022 5FFF	Reserved	4 kbyte
$0022 6000 - $0022 6FFF	Reserved	4 kbyte
$0022 7000 - $002F FFFF	Reserved	868 kbyte
$0030 0000 - $003F FFFF	Reserved	1 Mbyte
$0040 0000 - $07FF FFFF	Reserved	124 Mbyte
$0800 0000 - $0BFF FFFF	External memory ($\overline{CSD0}$)	64 Mbyte
$0C00 0000 - $0FFF FFFF	External memory ($\overline{CSD1}$)	64 Mbyte
$1000 0000 - $11FF FFFF	External memory ($\overline{CS0}$)	32 Mbyte
$1200 0000 - $12FF FFFF	External memory ($\overline{CS1}$)	16 Mbyte
$1300 0000 - $13FF FFFF	External Memory ($\overline{CS2}$)	16 Mbyte
$1400 0000 - $14FF FFFF	External Memory ($\overline{CS3}$)	16 Mbyte
$1500 0000 - $15FF FFFF	External Memory ($\overline{CS4}$)	16 Mbyte
$1600 0000 - $16FF FFFF	External Memory ($\overline{CS5}$)	16 Mbyte
$1700 0000 - $4FFF FFFF	Reserved	912 Mbyte
$5000 0000 - $5000 0FFF	ARM920T Test Registers	4 kbyte
$5000 1000 - $FFFF FFFF	Reserved	2815 Mbyte + 1020 kbyte

3.1.2 Internal Register Space

Internal registers are located from 0x00200000 to 0x00224FFF. Some of the MC9328MXL peripherals are each allocated 4 Kbyte starting at address $00200000 and they are connected to the AIPI1 (AHB IP Interface). Any ARM920T core write access to these modules will experience two wait states—that is, any write access will be a three cycle long access, and any ARM920T core read access from these modules will have one wait state—that is, any read access will be two cycle long access. The other MC9328MXL peripherals are each allocated 4 Kbyte starting at address $00210000 and they are connected to the AIPI2. Any ARM920T core write access to these modules will have two wait states—that is, any write access will be a three cycle long access, and any ARM920T core read access from these modules will have one wait state—any read access will be two cycle long access.

4 kbyte address space beginning at 0x00220000 to 0x00220FFF is assigned for EIM internal registers.

4 kbyte address space beginning at 0x00221000 to 0x00221FFF is assigned for SDRAMC internal registers.

4 kbyte address space beginning at 0x00222000 to 0x00222FFF is assigned for MMA internal registers.

4 kbyte address space beginning at 0x00223000 to 0x00223FFF is assigned for AITC internal registers.

4 kbyte address space beginning at 0x00224000 to 0x00224FFF is assigned for CSI internal registers.

Within each 4 kbyte peripheral space, any number of architected registers may be defined (as outlined in the chapter for each peripheral), and software must explicitly address them making no assumptions regarding multiple mapping.

3.1.3 External Memory

There are 240 Mbytes of the memory map allocated for external chip access, beginning at address $08000000. There are 8 external chip selects which are allocated 64 Mbyte each for CSD1–CSD0, 16 Mbyte each for CS5Q–CS1, and 32 Mbyte for CS0.

3.1.4 Double Map Image

The first 1 Mbyte system address space (starting at address $0) is defined as double map image space. This address space is mapped to the first 1 Mbyte of boot ROM upon power up. In MC9328MXL the boot ROM can be either SyncFlash, CS0, or Bootstrap ROM. After system power up, reading or writing to the double map space ($0000,0000 to $000F,FFFF) is the same as reading or writing to the first 1 Mbyte of the selected boot ROM which is controlled by the configuration of BOOT [3:0] input pins.

3.2 Internal Registers

The internal registers in the MC9328MXL are listed in Table 3-2.

Table 3-2. Internal Registers Sorted by Address

Module Name	Address	Name	Description
AIPI1	0x00200000	PSR0_1	AIPI1 Peripheral Size Register 0
AIPI1	0x00200004	PSR1_1	AIPI1 Peripheral Size Register 1
AIPI1	0x00200008	PAR_1	AIPI1 Peripheral Access Register
AIPI1	0x0020000C	PCR_1	AIPI1 Peripheral Control Register
AIPI1	0x00200010	TSR_1	AIPI1 Time-Out Status Register
Watchdog	0x00201000	WCR	Watchdog Control Register
Watchdog	0x00201004	WSR	Watchdog Service Register
Watchdog	0x00201008	WSTR	Watchdog Status Registert
Timer 1	0x00202000	TCTL1	Timer 1 Control Register
Timer 1	0x00202004	TPRER1	Timer 1 Prescaler Register
Timer 1	0x00202008	TCMP1	Timer 1 Compare Register
Timer 1	0x0020200C	TCR1	Timer 1 Capture Register
Timer 1	0x00202010	TCN1	Timer 1 Counter Register
Timer 1	0x00202014	TSTAT1	Timer 1 Status Register
Timer 2	0x00203000	TCTL2	Timer 2 Control Register
Timer 2	0x00203004	TPRER2	Timer 2 Prescaler Register
Timer 2	0x00203008	TCMP2	Timer 2 Compare Register
Timer 2	0x0020300C	TCR2	Timer 2 Capture Register
Timer 2	0x00203010	TCN2	Timer 2 Counter Register
Timer 2	0x00203014	TSTAT2	Timer 2 Status Register
RTC	0x00204000	HOURMIN	RTC Hours and Minutes Counter Register
RTC	0x00204004	SECONDS	RTC Seconds Counter Register
RTC	0x00204008	ALRM_HM	RTC Hours and Minutes Alarm Register
RTC	0x0020400C	ALRM_SEC	RTC Seconds Alarm Register
RTC	0x00204010	RCCTL	RTC Control Register

Table 3-2. Internal Registers Sorted by Address (continued)

Module Name	Address	Name	Description
RTC	0x00204014	RTCISR	RTC Interrupt Status Register
RTC	0x00204018	RTCIENR	RTC Interrupt Enable Register
RTC	0x0020401C	STPWCH	Stopwatch Minutes Register
RTC	0x00204020	DAYR	RTC Days Counter Register
RTC	0x00204024	DAYALARM	RTC Day Alarm Register
LCDC	0x00205000	SSA	Screen Start Address Register
LCDC	0x00205004	SIZE	Size Register
LCDC	0x00205008	VPW	Virtual Page Width Register
LCDC	0x0020500C	CPOS	LCD Cursor Position Register
LCDC	0x00205010	LCWHB	LCD Cursor Width Height and Blink Register
LCDC	0x00205014	LCHCC	LCD Color Cursor Mapping Register
LCDC	0x00205018	PCR	Panel Configuration Register
LCDC	0x0020501C	HCR	Horizontal Configuration Register
LCDC	0x00205020	VCR	Vertical Configuration Register
LCDC	0x00205024	POS	Panning Offset Register
LCDC	0x00205028	LGPMR	LCD Gray Palette Mapping Register
LCDC	0x0020502C	PWMR	PWM Contrast Control Register
LCDC	0x00205030	DMACR	DMA Control Register
LCDC	0x00205034	RMCR	Refresh Mode Control Register
LCDC	0x00205038	LCDICR	Interrupt Configuration Register
LCDC	0x00205040	LCDISR	Interrupt Status Register
UART 1	0x00206000	URX0D_1	UART1 Receiver Register 0
UART 1	0x00206004	URX1D_1	UART1 Receiver Register 1
UART 1	0x00206008	URX2D_1	UART1 Receiver Register 2
UART 1	0x0020600C	URX3D_1	UART1 Receiver Register 3
UART 1	0x00206010	URX4D_1	UART1 Receiver Register 4
UART 1	0x00206014	URX5D_1	UART1 Receiver Register 5
UART 1	0x00206018	URX6D_1	UART1 Receiver Register 6

Table 3-2. Internal Registers Sorted by Address (continued)

Module Name	Address	Name	Description
UART 1	0x0020601C	URX7D_1	UART1 Receiver Register 7
UART 1	0x00206020	URX8D_1	UART1 Receiver Register 8
UART 1	0x00206024	URX9D_1	UART1 Receiver Register 9
UART 1	0x00206028	URX10D_1	UART1 Receiver Register 10
UART 1	0x0020602C	URX11D_1	UART1 Receiver Register 11
UART 1	0x00206030	URX12D_1	UART1 Receiver Register 12
UART 1	0x00206034	URX13D_1	UART1 Receiver Register 13
UART 1	0x00206038	URX14D_1	UART1 Receiver Register 14
UART 1	0x0020603C	URX15D_1	UART1 Receiver Register 15
UART 1	0x00206040	UTX0D_1	UART1 Transmitter Register 0
UART 1	0x00206044	UTX1D_1	UART1 Transmitter Register 1
UART 1	0x00206048	UTX2D_1	UART1 Transmitter Register 2
UART 1	0x0020604C	UTX3D_1	UART1 Transmitter Register 3
UART 1	0x00206050	UTX4D_1	UART1 Transmitter Register 4
UART 1	0x00206054	UTX5D_1	UART1 Transmitter Register 5
UART 1	0x00206058	UTX6D_1	UART1 Transmitter Register 6
UART 1	0x0020605C	UTX7D_1	UART1 Transmitter Register 7
UART 1	0x00206060	UTX8D_1	UART1 Transmitter Register 8
UART 1	0x00206064	UTX9D_1	UART1 Transmitter Register 9
UART 1	0x00206068	UTX10D_1	UART1 Transmitter Register 10
UART 1	0x0020606C	UTX11D_1	UART1 Transmitter Register 11
UART 1	0x00206070	UTX12D_1	UART1 Transmitter Register 12
UART 1	0x00206074	UTX13D_1	UART1 Transmitter Register 13
UART 1	0x00206078	UTX14D_1	UART1 Transmitter Register 14
UART 1	0x0020607C	UTX15D_1	UART1 Transmitter Register 15
UART 1	0x00206080	UCR1_1	UART1 Control Register 1
UART 1	0x00206084	UCR2_1	UART1 Control Register 2
UART 1	0x00206088	UCR3_1	UART1 Control Register 3

Table 3-2. Internal Registers Sorted by Address (continued)

Module Name	Address	Name	Description
UART 1	0x0020608C	UCR4_1	UART1 Control Register 4
UART 1	0x00206090	UFCR_1	UART1 FIFO Control Register
UART 1	0x00206094	USR1_1	UART1 Status Register 1
UART 1	0x00206098	USR2_1	UART1 Status Register 2
UART 1	0x0020609C	UESC_1	UART1 Escape Character Register
UART 1	0x002060A0	UTIM_1	UART1 Escape Timer Register
UART 1	0x002060A4	UBIR_1	UART1 BRM Incremental Register
UART 1	0x002060A8	UBMR_1	UART1 BRM Modulator Register
UART 1	0x002060AC	UBRC_1	UART1 Baud Rate Count Register
UART 1	0x002060B0	BIPR1_1	UART1 BRM Incremental Preset Register 1
UART 1	0x002060B4	BIPR2_1	UART1 BRM Incremental Preset Register 2
UART 1	0x002060B8	BIPR3_1	UART1 BRM Incremental Preset Register 3
UART 1	0x002060BC	BIPR4_1	UART1 BRM Incremental Preset Register 4
UART 1	0x002060C0	BMPR1_1	UART1 BRM Modulator Preset Register 1
UART 1	0x002060C4	BMPR2_1	UART1 BRM Modulator Preset Register 2
UART 1	0x002060C8	BMPR3_1	UART1 BRM Modulator Preset Register 3
UART 1	0x002060CC	BMPR4_1	UART1 BRM Modulator Preset Register 4
UART 1	0x002060D0	UTS_1	UART1 Test Register 1
UART 2	0x00207000	URX0D_2	UART2 Receiver Register 0
UART 2	0x00207004	URX1D_2	UART2 Receiver Register 1
UART 2	0x00207008	URX2D_2	UART2 Receiver Register 2
UART 2	0x0020700C	URX3D_2	UART2 Receiver Register 3
UART 2	0x00207010	URX4D_2	UART2 Receiver Register 4
UART 2	0x00207014	URX5D_2	UART2 Receiver Register 5
UART 2	0x00207018	URX6D_2	UART2 Receiver Register 6
UART 2	0x0020701C	URX7D_2	UART2 Receiver Register 7
UART 2	0x00207020	URX8D_2	UART2 Receiver Register 8
UART 2	0x00207024	URX9D_2	UART2 Receiver Register 9

Table 3-2. Internal Registers Sorted by Address (continued)

Module Name	Address	Name	Description
UART 2	0x00207028	URX10D_2	UART2 Receiver Register 10
UART 2	0x0020702C	URX11D_2	UART2 Receiver Register 11
UART 2	0x00207030	URX12D_2	UART2 Receiver Register 12
UART 2	0x00207034	URX13D_2	UART2 Receiver Register 13
UART 2	0x00207038	URX14D_2	UART2 Receiver Register 14
UART 2	0x0020703C	URX15D_2	UART2 Receiver Register 15
UART 2	0X00207040	UTX0D_2	UART2 Transmitter Register 0
UART 2	0X00207044	UTX1D_2	UART2 Transmitter Register 1
UART 2	0X00207048	UTX2D_2	UART2 Transmitter Register 2
UART 2	0X0020704C	UTX3D_2	UART2 Transmitter Register 3
UART 2	0X00207050	UTX4D_2	UART2 Transmitter Register 4
UART 2	0X00207054	UTX5D_2	UART2 Transmitter Register 5
UART 2	0X00207058	UTX6D_2	UART2 Transmitter Register 6
UART 2	0X0020705C	UTX7D_2	UART2 Transmitter Register 7
UART 2	0X00207060	UTX8D_2	UART2 Transmitter Register 8
UART 2	0X00207064	UTX9D_2	UART2 Transmitter Register 9
UART 2	0X00207068	UTX10D_2	UART2 Transmitter Register 10
UART 2	0X0020706C	UTX11D_2	UART2 Transmitter Register 11
UART 2	0X00207070	UTX12D_2	UART2 Transmitter Register 12
UART 2	0X00207074	UTX13D_2	UART2 Transmitter Register 13
UART 2	0X00207078	UTX14D_2	UART2 Transmitter Register 14
UART 2	0X0020707C	UTX15D_2	UART2 Transmitter Register 15
UART 2	0x00207080	UCR1_2	UART2 Control Register 1
UART 2	0x00207084	UCR2_2	UART2 Control Register 2
UART 2	0x00207088	UCR3_2	UART2 Control Register 3
UART 2	0x0020708C	UCR4_2	UART2 Control Register 4
UART 2	0x00207090	UFCR_2	UART2 FIFO Control Register
UART 2	0x00207094	USR1_2	UART2 Status Register 1

Table 3-2. Internal Registers Sorted by Address (continued)

Module Name	Address	Name	Description
UART 2	0x00207098	USR2_2	UART2 Status Register 2
UART 2	0x0020709C	UESC_2	UART2 Escape Character Register
UART 2	0x002070A0	UTIM_2	UART2 Escape Timer Register
UART 2	0x002070A4	UBIR_2	UART2 BRM Incremental Register
UART 2	0x002070A8	UBMR_2	UART2 BRM Modulator Register
UART 2	0x002070AC	UBRC_2	UART2 Baud Rate Count Register
UART 2	0x002070B0	BIPR1_2	UART2 BRM Incremental Preset Register 1
UART 2	0x002070B4	BIPR2_2	UART2 BRM Incremental Preset Register 2
UART 2	0x002070B8	BIPR3_2	UART2 BRM Incremental Preset Register 3
UART 2	0x002070BC	BIPR4_2	UART2 BRM Incremental Preset Register 4
UART 2	0x002070C0	BMPR1_2	UART2 BRM Modulator Preset Register 1
UART 2	0x002070C4	BMPR2_2	UART2 BRM Modulator Preset Register 2
UART 2	0x002070C8	BMPR3_2	UART2 BRM Modulator Preset Register 3
UART 2	0x002070CC	BMPR4_2	UART2 BRM Modulator Preset Register 4
UART 2	0x002070D0	UTS_2	UART2 Test Register 1
PWM	0x00208000	PWMC	PWM Control Register
PWM	0x00208004	PWMS	PWM Sample Register
PWM	0x00208008	PWMP	PWM Period Register
PWM	0x0020800C	PWMCNT	PWM Counter Register
DMAC	0x00209000	DCR	DMA Control Register
DMAC	0x00209004	DISR	DMA Interrupt Status Register
DMAC	0x00209008	DIMR	DMA Interrupt Mask Register
DMAC	0x0020900C	DBTOSR	DMA Burst Time-Out Status Register
DMAC	0x00209010	DRTOSR	DMA Request Time-Out Status Register
DMAC	0x00209014	DSESR	DMA Transfer Error Status Register
DMAC	0x00209018	DBOSR	DMA Buffer Overflow Status Register
DMAC	0x0020901C	DBTOCR	DMA Burst Time-Out Control Register
DMAC	0x00209040	WSRA	W-Size Register A

Table 3-2. Internal Registers Sorted by Address (continued)

Module Name	Address	Name	Description
DMAC	0x00209044	XSRA	X-Size Register A
DMAC	0x00209048	YSRA	Y-Size Register A
DMAC	0x0020904C	WSRB	W-Size Register B
DMAC	0x00209050	XSRB	X-Size Register B
DMAC	0x00209054	YSRB	Y-Size Register B
DMAC	0x00209080	SAR0	Channel 0 Source Address Register
DMAC	0x00209084	DAR0	Channel 0 Destination Address Register
DMAC	0x00209088	CNTR0	Channel 0 Count Register
DMAC	0x0020908C	CCR0	Channel 0 Control Register
DMAC	0x00209090	RSSR0	Channel 0 Request Source Select Register
DMAC	0x00209094	BLR0	Channel 0 Burst Length Register
DMAC	0x00209098	RTOR0 BUCR0	Channel 0 Request Time-Out Register Channel 0 Bus Utilization Control Register
DMAC	0x002090C0	SAR1	Channel 1 Source Address Register
DMAC	0x002090C4	DAR1	Channel 1 Destination Address Register
DMAC	0x002090C8	CNTR1	Channel 1 Count Register
DMAC	0x002090CC	CCR1	Channel 1 Control Register
DMAC	0x002090D0	RSSR1	Channel 1 Request Source Select Register
DMAC	0x002090D4	BLR1	Channel 1 Burst Length Register
DMAC	0x002090D8	RTOR1 BUCR1	Channel 1 Request Time-Out Register Channel 1 Bus Utilization Control Register
DMAC	0x00209100	SAR2	Channel 2 Source Address Register
DMAC	0x00209104	DAR2	Channel 2 Destination Address Register
DMAC	0x00209108	CNTR2	Channel 2 Count Register
DMAC	0x0020910C	CCR2	Channel 2 Control Register
DMAC	0x00209110	RSSR2	Channel 2 Request Source Select Register
DMAC	0x00209114	BLR2	Channel 2 Burst Length Register
DMAC	0x00209118	RTOR2 BUCR2	Channel 2 Request Time-Out Register Channel 2 Bus Utilization Control Register

Table 3-2. Internal Registers Sorted by Address (continued)

Module Name	Address	Name	Description
DMAC	0x00209140	SAR3	Channel 3 Source Address Register
DMAC	0x00209144	DAR3	Channel 3 Destination Address Register
DMAC	0x00209148	CNTR3	Channel 3 Count Register
DMAC	0x0020914C	CCR3	Channel 3 Control Register
DMAC	0x00209150	RSSR3	Channel 3 Request Source Select Register
DMAC	0x00209154	BLR3	Channel 3 Burst Length Register
DMAC	0x00209158	RTOR3 BUCR3	Channel 3 Request Time-Out Register Channel 3 Bus Utilization Control Register
DMAC	0x00209180	SAR4	Channel 4 Source Address Register
DMAC	0x00209184	DAR4	Channel 4 Destination Address Register
DMAC	0x00209188	CNTR4	Channel 4 Count Register
DMAC	0x0020918C	CCR4	Channel 4 Control Register
DMAC	0x00209190	RSSR4	Channel 4 Request Source Select Register
DMAC	0x00209194	BLR4	Channel 4 Burst Length Register
DMAC	0x00209198	RTOR4 BUCR4	Channel 4 Request Time-Out Register Channel 4 Bus Utilization Control Register
DMAC	0x002091C0	SAR5	Channel 5 Source Address Register
DMAC	0x002091C4	DAR5	Channel 5 Destination Address Register
DMAC	0x002091C8	CNTR5	Channel 5 Count Register
DMAC	0x002091CC	CCR5	Channel 5 Control Register
DMAC	0x002091D0	RSSR5	Channel 5 Request Source Select Register
DMAC	0x002091D4	BLR5	Channel 5 Burst Length Register
DMAC	0x002091D8	RTOR5 BUCR5	Channel 5 Request Time-Out Register Channel 5 Bus Utilization Control Register
DMAC	0x00209200	SAR6	Channel 6 Source Address Register
DMAC	0x00209204	DAR6	Channel 6 Destination Address Register
DMAC	0x00209208	CNTR6	Channel 6 Count Register
DMAC	0x0020920C	CCR6	Channel 6 Control Register
DMAC	0x00209210	RSSR6	Channel 6 Request Source Select Register

Table 3-2. Internal Registers Sorted by Address (continued)

Module Name	Address	Name	Description
DMAC	0x00209214	BLR6	Channel 6 Burst Length Register
DMAC	0x00209218	RTOR6 BUCR6	Channel 6 Request Time-Out Register Channel 6 Bus Utilization Control Register
DMAC	0x00209240	SAR7	Channel 7 Source Address Register
DMAC	0x00209244	DAR7	Channel 7 Destination Address Register
DMAC	0x00209248	CNTR7	Channel 7 Count Register
DMAC	0x0020924C	CCR7	Channel 7 Control Register
DMAC	0x00209250	RSSR7	Channel 7 Request Source Select Register
DMAC	0x00209254	BLR7	Channel 7 Burst Length Register
DMAC	0x00209258	RTOR7 BUCR7	Channel 7 Request Time-Out Register Channel 7 Bus Utilization Control Register
DMAC	0x00209280	SAR8	Channel 8 Source Address Register
DMAC	0x00209284	DAR8	Channel 8 Destination Address Register
DMAC	0x00209288	CNTR8	Channel 8 Count Register
DMAC	0x0020928C	CCR8	Channel 8 Control Register
DMAC	0x00209290	RSSR8	Channel 8 Request Source Select Register
DMAC	0x00209294	BLR8	Channel 8 Burst Length Register
DMAC	0x00209298	RTOR8 BUCR8	Channel 8 Request Time-Out Register Channel 8 Bus Utilization Control Register
DMAC	0x002092C0	SAR9	Channel 9 Source Address Register
DMAC	0x002092C4	DAR9	Channel 9 Destination Address Register
DMAC	0x002092C8	CNTR9	Channel 9 Count Register
DMAC	0x002092CC	CCR9	Channel 9 Control Register
DMAC	0x002092D0	RSSR9	Channel 9 Request Source Select Register
DMAC	0x002092D4	BLR9	Channel 9 Burst Length Register
DMAC	0x002092D8	RTOR9 BUCR9	Channel 9 Request Time-Out Register Channel 9 Bus Utilization Control Register
DMAC	0x00209300	SAR10	Channel 10 Source Address Register
DMAC	0x00209304	DAR10	Channel 10 Destination Address Register
DMAC	0x00209308	CNTR10	Channel 10 Count Register

Table 3-2. Internal Registers Sorted by Address (continued)

Module Name	Address	Name	Description
AIPI2	0x00210000	PSR0_2	AIPI2 Peripheral Size Register 0
AIPI2	0x00210004	PSR1_2	AIPI2 Peripheral Size Register 1
AIPI2	0x00210008	PAR_2	AIPI2 Peripheral Access Register
AIPI2	0x0021000C	PCR_2	AIPI2 Peripheral Control Register
AIPI2	0x00210010	TSR_2	AIPI2Time-Out Status Register
USBD	0x00212000	USB_FRAME	USB Frame Number and Match Register
USBD	0x00212004	USB_SPEC	USB Specification and Release Number Register
USBD	0x00212008	USB_STAT	USB Status Register
USBD	0x0021200C	USB_CTRL	USB Control Register
USBD	0x00212010	USB_DADR	USB Descriptor RAM Address Register
USBD	0x00212014	USB_DDAT	USB Descriptor RAM/Endpoint Buffer Data Register
USBD	0x00212018	USB_INTR	USB Interrupt Status Register
USBD	0x0021201C	USB_MASK	USB Interrupt Mask Register
USBD	0x00212020	USB_MCTL	USB FIFO Memory Control Register
USBD	0x00212024	USB_ENAB	USB Enable Register
USBD	0x00212030	USB_EP0_STAT	Endpoint 0 Status/Control Register
USBD	0x00212034	USB_EP0_INTR	Endpoint 0 Interrupt Status Register
USBD	0x00212038	USB_EP0_MASK	Endpoint 0 Interrupt Mask Register
USBD	0x0021203C	USB_EP0_FDAT	Endpoint 0 FIFO Data Register
USBD	0x00212040	USB_EP0_FSTAT	Endpoint 0 FIFO Status Register
USBD	0x00212044	USB_EP0_FCTRL	Endpoint 0 FIFO Control Register
USBD	0x00212048	USB_EP0_LRFP	Endpoint 0 Last Read Frame Pointer Register
USBD	0x0021204C	USB_EP0_LWFP	Endpoint 0 Last Write Frame Pointer Register
USBD	0x00212050	USB_EP0_FALRM	Endpoint 0 FIFO Alarm Register
USBD	0x00212054	USB_EP0_FRDP	Endpoint 0 FIFO Read Pointer Register
USBD	0x00212058	USB_EP0_FWRP	Endpoint 0 FIFO Write Pointer Register
USBD	0x00212060	USB_EP1_STAT	Endpoint 1 Status/Control Register
USBD	0x00212064	USB_EP1_INTR	Endpoint 1 Interrupt Status Register

Table 3-2. Internal Registers Sorted by Address (continued)

Module Name	Address	Name	Description
USBD	0x00212068	USB_EP1_MASK	Endpoint 1 Interrupt Mask Register
USBD	0x0021206C	USB_EP1_FDAT	Endpoint 1 FIFO Data Register
USBD	0x00212070	USB_EP1_FSTAT	Endpoint 1 FIFO Status Register
USBD	0x00212074	USB_EP1_FCTRL	Endpoint 1 FIFO Control Register
USBD	0x00212078	USB_EP1_LRFP	Endpoint 1 Last Read Frame Pointer Register
USBD	0x0021207C	USB_EP1_LWFP	Endpoint 1 Last Write Frame Pointer Register
USBD	0x00212080	USB_EP1_FALRM	Endpoint 1 FIFO Alarm Register
USBD	0x00212084	USB_EP1_FRDP	Endpoint 1 FIFO Read Pointer Register
USBD	0x00212088	USB_EP1_FWRP	Endpoint 1 FIFO Write Pointer Register
USBD	0x00212090	USB_EP2_STAT	Endpoint 2 Status/Control Register
USBD	0x00212094	USB_EP2_INTR	Endpoint 2 Interrupt Status Register
USBD	0x00212098	USB_EP2_MASK	Endpoint 2 Interrupt Mask Register
USBD	0x0021209C	USB_EP2_FDAT	Endpoint 2 FIFO Data Register
USBD	0x002120A0	USB_EP2_FSTAT	Endpoint 2 FIFO Status Register
USBD	0x002120A4	USB_EP2_FCTRL	Endpoint 2 FIFO Control Register
USBD	0x002120A8	USB_EP2_LRFP	Endpoint 2 Last Read Frame Pointer Register
USBD	0x002120AC	USB_EP2_LWFP	Endpoint 2 Last Write Frame Pointer Register
USBD	0x002120B0	USB_EP2_FALRM	Endpoint 2 FIFO Alarm Register
USBD	0x002120B4	USB_EP2_FRDP	Endpoint 2 FIFO Read Pointer Register
USBD	0x002120B8	USB_EP2_FWRP	Endpoint 2 FIFO Write Pointer Register
USBD	0x002120C0	USB_EP3_STAT	Endpoint 3 Status/Control Register
USBD	0x002120C4	USB_EP3_INTR	Endpoint 3 Interrupt Status Register
USBD	0x002120C8	USB_EP3_MASK	Endpoint 3 Interrupt Mask Register
USBD	0x002120CC	USB_EP3_FDAT	Endpoint 3 FIFO Data Register
USBD	0x002120D0	USB_EP3_FSTAT	Endpoint 3 FIFO Status Register
USBD	0x002120D4	USB_EP3_FCTRL	Endpoint 3 FIFO Control Register
USBD	0x002120D8	USB_EP3_LRFP	Endpoint 3 Last Read Frame Pointer Register
USBD	0x002120DC	USB_EP3_LWFP	Endpoint 3 Last Write Frame Pointer Register

Table 3-2. Internal Registers Sorted by Address (continued)

Module Name	Address	Name	Description
USBD	0x002120E0	USB_EP3_FALRM	Endpoint 3 FIFO Alarm Register
USBD	0x002120E4	USB_EP3_FRDP	Endpoint 3 FIFO Read Pointer Register
USBD	0x002120E8	USB_EP3_FWRP	Endpoint 3 FIFO Write Pointer Register
USBD	0x002120F0	USB_EP4_STAT	Endpoint 4 Status/Control Register
USBD	0x002120F4	USB_EP4_INTR	Endpoint 4 Interrupt Status Register
USBD	0x002120F8	USB_EP4_MASK	Endpoint 4 Interrupt Mask Register
USBD	0x002120FC	USB_EP4_FDAT	Endpoint 4 FIFO Data Register
USBD	0x00212100	USB_EP4_FSTAT	Endpoint 4 FIFO Status Register
USBD	0x00212104	USB_EP4_FCTRL	Endpoint 4 FIFO Control Register
USBD	0x00212108	USB_EP4_LRFP	Endpoint 4 Last Read Frame Pointer Register
USBD	0x0021210C	USB_EP4_LWFP	Endpoint 4 Last Write Frame Pointer Register
USBD	0x00212110	USB_EP4_FALRM	Endpoint 4 FIFO Alarm Register
USBD	0x00212114	USB_EP4_FRDP	Endpoint 4 FIFO Read Pointer Register
USBD	0x00212118	USB_EP4_FWRP	Endpoint 4 FIFO Write Pointer Register
USBD	0x00212120	USB_EP5_STAT	Endpoint 5 Status/Control Register
USBD	0x00212124	USB_EP5_INTR	Endpoint 5 Interrupt Status Register
USBD	0x00212128	USB_EP5_MASK	Endpoint 5 Interrupt Mask Register
USBD	0x0021212C	USB_EP5_FDAT	Endpoint 5 FIFO Data Register
USBD	0x00212130	USB_EP5_FSTAT	Endpoint 5 FIFO Status Register
USBD	0x00212134	USB_EP5_FCTRL	Endpoint 5 FIFO Control Register
USBD	0x00212138	USB_EP5_LRFP	Endpoint 5 Last Read Frame Pointer Register
USBD	0x0021213C	USB_EP5_LWFP	Endpoint 5 Last Write Frame Pointer Register
USBD	0x00212140	USB_EP5_FALRM	Endpoint 5 FIFO Alarm Register
USBD	0x00212144	USB_EP5_FRDP	Endpoint 5 FIFO Read Pointer Register
USBD	0x00212148	USB_EP5_FWRP	Endpoint 5 FIFO Write Pointer Register
SPI 1	0x00213000	RXDATAREG1	SPI 1 Rx Data Register
SPI 1	0x00213004	TXDATAREG1	SPI 1 Tx Data Register
SPI 1	0x00213008	CONTROLREG1	SPI 1 Control Register

Table 3-2. Internal Registers Sorted by Address (continued)

Module Name	Address	Name	Description
SPI 1	0x0021300C	INTREG1	SPI 1 Interrupt Control/Status Register
SPI 1	0x00213010	TESTREG1	SPI 1 Test Register
SPI 1	0x00213014	PERIODREG1	SPI 1 Sample Period Control Register
SPI 1	0x00213018	DMAREG1	SPI 1 DMA Control Register
SPI 1	0x0021301C	RESETREG1	SPI 1 Soft Reset Register
MMC/SDHC	0x00214000	STR_STP_CLK	MMC/SD Clock Control Register
MMC/SDHC	0x00214004	STATUS	MMC/SD Status Register
MMC/SDHC	0x00214008	CLK_RATE	MMC/SD Clock Rate Register
MMC/SDHC	0x0021400C	CMD_DAT_CONT	MMC/SD Command and Data Control Register
MMC/SDHC	0x00214010	RES_TO	MMC/SD Response Time Out Register
MMC/SDHC	0x00214014	READ_TO	MMC/SD Read Time Out Register
MMC/SDHC	0x00214018	BLK_LEN	MMC/SD Block Length Register
MMC/SDHC	0x0021401C	NOB	MMC/SD Number of Blocks Register
MMC/SDHC	0x00214020	REV_NO	MMC/SD Revision Number Register
MMC/SDHC	0x00214024	INT_MASK	MMC/SD Interrupt Mask Register
MMC/SDHC	0x00214028	CMD	MMC/SD Command Number Register
MMC/SDHC	0x0021402C	ARGH	MMC/SD Higher Argument Register
MMC/SDHC	0x00214030	ARGL	MMC/SD Lower Argument Register
MMC/SDHC	0x00214034	RES_FIFO	MMC/SD Response FIFO Register
MMC/SDHC	0x00214038	BUFFER_ACCESS	MMC/SD Buffer Access Register
I^2C	0x00217000	IADR	I^2C Address Register
I^2C	0x00217004	IFDR	I^2C Frequency Divider Register
I^2C	0x00217008	I2CR	I^2C Control Register
I^2C	0x0021700C	I2CSR	I^2C Status Register
I^2C	0x00217010	I2DR	I^2C Data I/O Register
SSI	0x00218000	STX	SSI Transmit Data Register
SSI	0x00218004	SRX	SSI Receive Data Register

Table 3-2. Internal Registers Sorted by Address (continued)

Module Name	Address	Name	Description
SSI	0x00218008	SCSR	SSI Control/Status Register
SSI	0x0021800C	STCR	SSI Transmit Configuration Register
SSI	0x00218010	SRCR	SSI Receive Configuration Register
SSI	0x00218014	STCCR	SSI Transmit Clock Control Register
SSI	0x00218018	SRCCR	SSI Receive Clock Control Register
SSI	0x0021801C	STSR	SSI Time Slot Register
SSI	0x00218020	SFCSR	SSI FIFO Control/Status Register
SSI	0x00218028	SOR	SSI Option Register
SPI 2	0x00219000	RXDATAREG2	SPI 2 Rx Data Register
SPI 2	0x00219004	TXDATAREG2	SPI 2 Tx Data Register
SPI 2	0x00219008	CONTROLREG2	SPI 2 Control Register
SPI 2	0x0021900C	INTREG2	SPI 2 Interrupt Control/Status Register
SPI 2	0x00219010	TESTREG2	SPI 2 Test Register
SPI 2	0x00219014	PERIODREG2	SPI 2 Sample Period Control Register
SPI 2	0x00219018	DMAREG2	SPI 2 DMA Control Register
SPI 2	0x0021901C	RESETREG2	SPI 2 Soft Reset Register
MSHC	0x0021A000	MSCMD	Memory Stick Command Register
MSHC	0x0021A002	MSCS	Memory Stick Control/Status Register
MSHC	0x0021A004	MSTDATA	Memory Stick Transmit FIFO Data Register
MSHC	0x0021A004	MSRDATA	Memory Stick Receive FIFO Data Register
MSHC	0x0021A006	MSICS	Memory Stick Interrupt Control/Status Register
MSHC	0x0021A008	MSPPCD	Memory Stick Parallel Port Control/Data Register
MSHC	0x0021A00A	MSC2	Memory Stick Control 2 Register
MSHC	0x0021A00C	MSACD	Memory Stick Auto Command Register
MSHC	0x0021A00E	MSFAECS	Memory Stick FIFO Access Error Control/Status Register
MSHC	0x0021A010	MSCLKD	Memory Stick Serial Clock Divider Register
MSHC	0x0021A012	MSDRQC	Memory Stick DMA Request Control Register
PLLCLK	0x0021B000	CSCR	Clock Source Control Register

Table 3-2. Internal Registers Sorted by Address (continued)

Module Name	Address	Name	Description
PLLCLK	0x0021B004	MPCTL0	MCU PLL Control Register 0
PLLCLK	0x0021B0008	MPCTL1	MCU PLL and System Clock Control Register 1
PLLCLK	0x0021B00C	SPCTL0	System PLL Control Register 0
PLLCLK	0x0021B010	SPCTL1	System PLL Control Register 1
PLLCLK	0x0021B020	PCDR	Peripheral Clock Divider Register
RESET	0x0021B800	RSR	Reset Source Register
SYS CTRL	0x0021B804	SIDR	Silicon ID Register
SYS CTRL	0x0021B808	FMCR	Function Multiplexing Control Register
SYS CTRL	0x0021B80C	GPCR	Global Peripheral Control Register
SYS CTRL	0x0021B810	GCCR	Global Clock Control Register
GPIO A	0x0021C000	DDIR_A	Port A Data Direction Register
GPIO A	0x0021C004	OCR1_A	Port A Output Configuration Register 1
GPIO A	0x0021C008	OCR2_A	Port A Output Configuration Register 2
GPIO A	0x0021C00C	ICONFA1_A	Port A Input Configuration Register A1
GPIO A	0x0021C010	ICONFA2_A	Port A Input Configuration Register A2
GPIO A	0x0021C014	ICONFB1_A	Port A Input Configuration Register B1
GPIO A	0x0021C018	ICONFB2_A	Port A Input Configuration Register B2
GPIO A	0x0021C01C	DR_A	Port A Data Register
GPIO A	0x0021C020	GIUS_A	Port A GPIO In Use Register
GPIO A	0x0021C024	SSR_A	Port A Sample Status Register
GPIO A	0x0021C028	ICR1_A	Port A Interrupt Configuration Register 1
GPIO A	0x0021C02C	ICR2_A	Port A Interrupt Configuration Register 2
GPIO A	0x0021C030	IMR_A	Port A Interrupt Mask Register
GPIO A	0x0021C034	ISR_A	Port A Interrupt Status Register
GPIO A	0x0021C038	GPR_A	Port A General Purpose Register
GPIO A	0x0021C03C	SWR_A	Port A Software Reset Register
GPIO A	0x0021C040	PUEN_A	Port A Pull_Up Enable Register
GPIO B	0x0021C100	DDIR_B	Port B Data Direction Register

Table 3-2. Internal Registers Sorted by Address (continued)

Module Name	Address	Name	Description
GPIO B	0x0021C104	OCR1_B	Port B Output Configuration Register 1
GPIO B	0x0021C108	OCR2_B	Port B Output Configuration Register 2
GPIO B	0x0021C10C	ICONFA1_B	Port B Input Configuration Register A1
GPIO B	0x0021C110	ICONFA2_B	Port B Input Configuration Register A2
GPIO B	0x0021C114	ICONFB1_B	Port B Input Configuration Register B1
GPIO B	0x0021C118	ICONFB2_B	Port B Input Configuration Register B2
GPIO B	0x0021C11C	DR_B	Port B Data Register
GPIO B	0x0021C120	GIUS_B	Port B GPIO In Use Register
GPIO B	0x0021C124	SSR_B	Port B Sample Status Register
GPIO B	0x0021C128	ICR1_B	Port B Interrupt Configuration Register 1
GPIO B	0x0021C12C	ICR2_B	Port B Interrupt Configuration Register 2
GPIO B	0x0021C130	IMR_B	Port B Interrupt Mask Register
GPIO B	0x0021C134	ISR_B	Port B Interrupt Status Register
GPIO B	0x0021C138	GPR_B	Port B General Purpose Register
GPIO B	0x0021C13C	SWR_B	Port B Software Reset Register
GPIO B	0x0021C140	PUEN_B	Port B Pull_Up Enable Register
GPIO C	0x0021C200	DDIR_C	Port C Data Direction Register
GPIO C	0x0021C204	OCR1_C	Port C Output Configuration Register 1
GPIO C	0x0021C208	OCR2_C	Port C Output Configuration Register 2
GPIO C	0x0021C20C	ICONFA1_C	Port C Input Configuration Register A1
GPIO C	0x0021C210	ICONFA2_C	Port C Input Configuration Register A2
GPIO C	0x0021C214	ICONFB1_C	Port C Input Configuration Register B1
GPIO C	0x0021C218	ICONFB2_C	Port C Input Configuration Register B2
GPIO C	0x0021C21C	DR_C	Port C Data Register
GPIO C	0x0021C220	GIUS_C	Port C GPIO In Use Register
GPIO C	0x0021C224	SSR_C	Port C Sample Status Register
GPIO C	0x0021C228	ICR1_C	Port C Interrupt Configuration Register 1
GPIO C	0x0021C22C	ICR2_C	Port C Interrupt Configuration Register 2

Table 3-2. Internal Registers Sorted by Address (continued)

Module Name	Address	Name	Description
GPIO C	0x0021C230	IMR_C	Port C Interrupt Mask Register
GPIO C	0x0021C234	ISR_C	Port C Interrupt Status Register
GPIO C	0x0021C238	GPR_C	Port C General Purpose Register
GPIO C	0x0021C23C	SWR_C	Port C Software Reset Register
GPIO C	0x0021C240	PUEN_C	Port C Pull_Up Enable Register
GPIO D	0x0021C300	DDIR_D	Port D Data Direction Register
GPIO D	0x0021C304	OCR1_D	Port D Output Configuration Register 1
GPIO D	0x0021C308	OCR2_D	Port D Output Configuration Register 2
GPIO D	0x0021C30C	ICONFA1_D	Port D Input Configuration Register A1
GPIO D	0x0021C310	ICONFA2_D	Port D Input Configuration Register A2
GPIO D	0x0021C314	ICONFB1_D	Port D Input Configuration Register B1
GPIO D	0x0021C318	ICONFB2_D	Port D Input Configuration Register B2
GPIO D	0x0021C31C	DR_D	Port D Data Register
GPIO D	0x0021C320	GIUS_D	Port D GPIO In Use Register
GPIO D	0x0021C324	SSR_D	Port D Sample Status Register
GPIO D	0x0021C328	ICR1_D	Port D Interrupt Configuration Register 1
GPIO D	0x0021C32C	ICR2_D	Port D Interrupt Configuration Register 2
GPIO D	0x0021C330	IMR_D	Port D Interrupt Mask Register
GPIO D	0x0021C334	ISR_D	Port D Interrupt Status Register
GPIO D	0x0021C338	GPR_D	Port D General Purpose Register
GPIO D	0x0021C33C	SWR_D	Port D Software Reset Register
GPIO D	0x0021C340	PUEN_D	Port D Pull_Up Enable Register
EIM	0x00220000	CS0U	Chip Select 0 Upper Control Register
EIM	0x00220004	CS0L	Chip Select 0 Lower Control Register
EIM	0x00220008	CS1U	Chip Select 1 Upper Control Register
EIM	0x0022000C	CS1L	Chip Select 1 Lower Control Register
EIM	0x00220010	CS2U	Chip Select 2 Upper Control Register
EIM	0x00220014	CS2L	Chip Select 2 Lower Control Register

Table 3-2. Internal Registers Sorted by Address (continued)

Module Name	Address	Name	Description
EIM	0x00220018	CS3U	Chip Select 3 Upper Control Register
EIM	0x0022001C	CS3L	Chip Select 3 Lower Control Register
EIM	0x00220020	CS4U	Chip Select 4 Upper Control Register
EIM	0x00220024	CS4L	Chip Select 4 Lower Control Register
EIM	0x00220028	CS5U	Chip Select 5 Upper Control Register
EIM	0x0022002C	CS5L	Chip Select 5 Lower Control Register
EIM	0x00220030	EIM	EIM Configuration Register
SDRAMC	0x00221000	SDCTL0	SDRAM 0 Control Register
SDRAMC	0x00221004	SDCTL1	SDRAM 1 Control Register
SDRAMC	0x00221014	MISCELLANEOUS	Miscellaneous Register
SDRAMC	0x00221018	SDRST	SDRAM Reset Register
MMA	0x00222000	MMA_MAC_MOD	MMA MAC Module Register
MMA	0x00222004	MMA_MAC_CTRL	MMA MAC Control Register
MMA	0x00222008	MMA_MAC_MULT	MMA MAC Multiply Counter Register
MMA	0x0022200C	MMA_MAC_ACCU	MMA MAC Accumulate Counter Register
MMA	0x00222010	MMA_MAC_INTR	MMA MAC Interrupt Register
MMA	0x00222014	MMA_MAC_INTR_MASK	MMA MAC Interrupt Mask Register
MMA	0x00222018	MMA_MAC_FIFO	MMA MAC FIFO Register
MMA	0x0022201C	MMA_MAC_FIFO_STAT	MMA MAC FIFO Status Register
MMA	0x00222020	MMA_MAC_BURST	MMA MAC Burst Count Register
MMA	0x00222024	MMA_MAC_BITSEL	MMA MAC Bit Select Register
MMA	0x00222040	MMA_MAC_XYACCU	MMA MAC XY Count Accumulate Register
MMA	0x00222200	MMA_MAC_XBASE	MMA MAC X Base Address Register
MMA	0x00222204	MMA_MAC_XINDEX	MMA MAC X Index Register
MMA	0x00222208	MMA_MAC_XLENGTH	MMA MAC X Length Register
MMA	0x0022220C	MMA_MAC_XMODIFY	MMA MAC X Modify Register
MMA	0x00222210	MMA_MAC_XINCR	MMA MAC X Increment Register
MMA	0x00222214	MMA_MAC_XCOUNT	MMA MAC X Count Register

Table 3-2. Internal Registers Sorted by Address (continued)

Module Name	Address	Name	Description
MMA	0x00222300	MMA_MAC_YBASE	MMA MAC Y Base Address Register
MMA	0x00222304	MMA_MAC_YINDEX	MMA MAC Y Index Register
MMA	0x00222308	MMA_MAC_YLENGTH	MMA MAC Y Length Register
MMA	0x0022230C	MMA_MAC_YMODIFY	MMA MAC Y Modify Register
MMA	0x00222310	MMA_MAC_YINCR	MMA MAC Y Increment Register
MMA	0x00222314	MMA_MAC_YCOUNT	MMA MAC Y Count Register
MMA	0x00222400	MMA_DCTCTRL	DCT/iDCT Control Register
MMA	0x00222404	MMA_DCTVERSION	DCT/iDCT Version Register
MMA	0x00222408	MMA_DCTIRQENA	DCT/iDCT IRQ Enable Register
MMA	0x0022240C	MMA_DCTIRQSTAT	DCT/iDCT IRQ Status Register
MMA	0x00222410	DSA_DCTSRCDATA	DCT/iDCT Source Data Address
MMA	0x00222414	MMA_DCTDESDATA	DCT/iDCT Destination Data Address
MMA	0x00222418	MMA_DCTXOFF	DCT/iDCT X-Offset Address
MMA	0x0022241C	MMA_DCTYOFF	DCT/iDCT Y-Offset Address
MMA	0x00222420	MMA_DCTXYCNT	DCT/iDCT XY Count
MMA	0x00222424	MMA_DCTSKIP	DCT/iDCT Skip Address
MMA	0x00222500	MMA_DCTFIFO	DCT/iDCT Data FIFO
AITC	0x00223000	INTCNTL	Interrupt Control Register
AITC	0x00223004	NIMASK	Normal Interrupt Mask Register
AITC	0x00223008	INTENNUM	Interrupt Enable Number Register
AITC	0x0022300C	INTDISNUM	Interrupt Disable Number Register
AITC	0x00223010	INTENABLEH	Interrupt Enable Register High
AITC	0x00223014	INTENABLEL	Interrupt Enable Register Low
AITC	0x00223018	INTTYPEH	Interrupt Type Register High
AITC	0x0022301C	INTTYPEL	Interrupt Type Register Low
AITC	0x00223020	NIPRIORITY7	Normal Interrupt Priority Level Register 7
AITC	0x00223024	NIPRIORITY6	Normal Interrupt Priority Level Register 6
AITC	0x00223028	NIPRIORITY5	Normal Interrupt Priority Level Register 5

Table 3-2. Internal Registers Sorted by Address (continued)

Module Name	Address	Name	Description
AITC	0x0022302C	NIPRIORITY4	Normal Interrupt Priority Level Register 4
AITC	0x00223030	NIPRIORITY3	Normal Interrupt Priority Level Register 3
AITC	0x00223034	NIPRIORITY2	Normal Interrupt Priority Level Register 2
AITC	0x00223038	NIPRIORITY1	Normal Interrupt Priority Level Register 1
AITC	0x0022303C	NIPRIORITY0	Normal Interrupt Priority Level Register 0
AITC	0x00223040	NIVECSR	Normal Interrupt Vector and Status Register
AITC	0x00223044	FIVECSR	Fast Interrupt Vector and Status Register
AITC	0x00223048	INTSRCH	Interrupt Source Register High
AITC	0x0022304C	INTSRCL	Interrupt Source Register Low
AITC	0x00223050	INTFRCH	Interrupt Force Register High
AITC	0x00223054	INTFRCL	Interrupt Force Register Low
AITC	0x00223058	NIPNDH	Normal Interrupt Pending Register High
AITC	0x0022305C	NIPNDL	Normal Interrupt Pending Register Low
AITC	0x00223060	FIPNDH	Fast Interrupt Pending Register High
AITC	0x00223064	FIPNDL	Fast Interrupt Pending Register Low
CSI	0x00224000	CSICR1	CSI Control Register 1
CSI	0x00224004	CSICR2	CSI Control Register 2
CSI	0x00224008	CSISR	CSI Status Register 1
CSI	0x0022400C	CSISTATR	CSI Statistic FIFO Register 1
CSI	0x00224010	CSIRXR	CSI RxFIFO Register 1

Chapter 4
ARM920T Processor

This chapter describes the operational features of the ARM920T™ high-performance processor that includes an overall summary of both the ARM920T processor core and the Thumb® instruction set as well as the operational modes. For detailed technical and programming information about the ARM920T processor refer to the *ARM920T Technical Reference Manual* (ARM Limited: 2001, order number DDI 0151C).

4.1 Introduction

The ARM920T processor is a high-performance 32-bit RISC integer processor macrocell combining an ARM9TDMI™ core with:

- 16 kbyte instruction and 16 kbyte data caches
- Instruction and data Memory Management Units (MMUs)
- Write buffer
- AMBA™ (Advanced Microprocessor Bus Architecture) bus interface
- Embedded Trace Macrocell (ETM) interface

An enhanced ARM® architecture v4 MMU implementation provides translation and access permission checks for instruction and data addresses. The ARM920T high-performance processor solution gives considerable savings in chip complexity and area, chip system design, and power consumption. The ARM920T processor is 100% user code binary compatible with ARM7TDMI®, and backwards compatible with the ARM7™ Thumb® Family and the StrongARM® processor families, giving designers software-compatible processors with a range of price/performance points from 60 MIPS to 200+ MIPS.

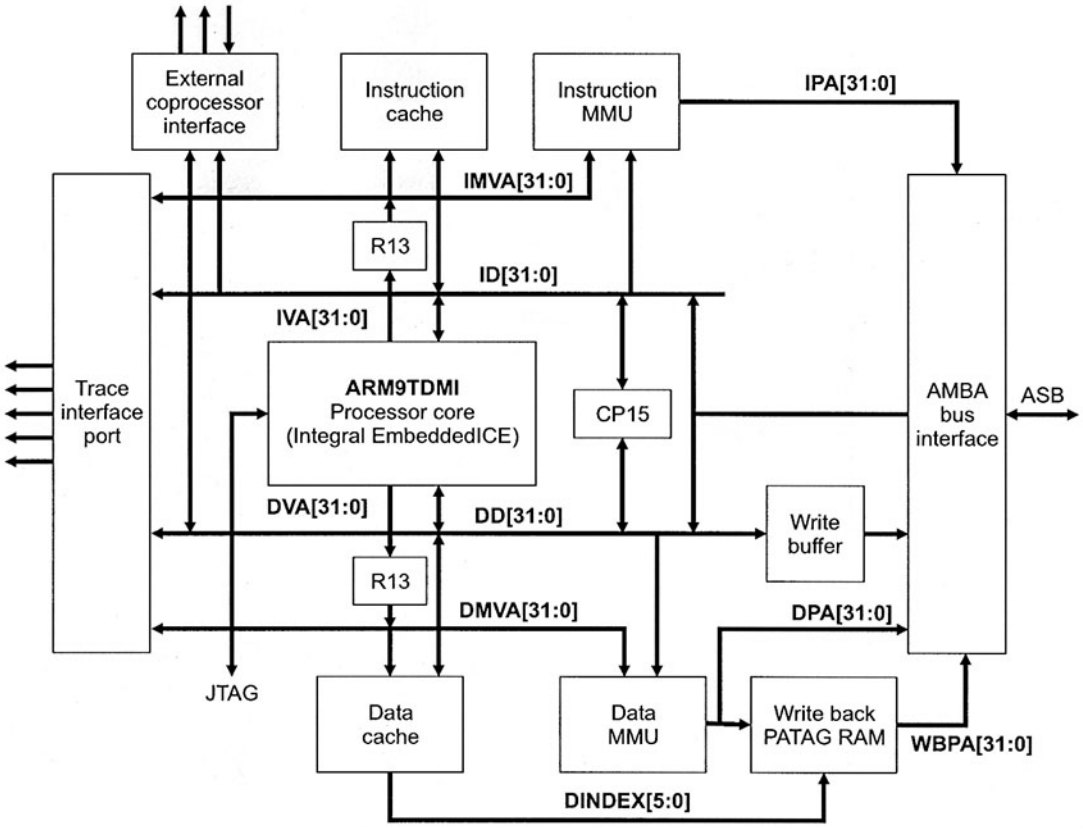


Figure 4-1. ARM920T Core Functional Block Diagram

4.2 ARM920T Macrocell

The ARM920T macrocell is based on the ARM9TDMI™ Harvard architecture processor core, with an efficient 5-stage pipeline. The architecture of the processor core or integer unit is described in more detail later in this chapter.

To reduce the effect of main memory bandwidth and latency on performance, the ARM920T processor includes:

- Instruction cache
- Data cache
- MMU
- TLBs
- Write buffer
- Physical address TAG RAM

4.2.1 Caches

Two 16 kbyte caches are implemented, one for instructions, the other for data, both with an 8-word line size. A 32-bit data bus connects each cache to the ARM9TDMI core allowing a 32-bit instruction to be fetched and fed into the instruction Decode stage of the pipeline at the same time as a 32-bit data access for the Memory stage of the pipeline.

4.2.2 Cache Lock-Down

Cache lock-down is provided to allow critical code sequences to be locked into the cache to ensure predictability for real-time code. The cache replacement algorithm can be selected by the operating system as either pseudo random or round-robin. Both caches are 64-way set associative. Lock-down operates on a per-set basis.

4.2.3 Write Buffer

The ARM920T processor also incorporates a 16-entry write buffer, to avoid stalling the processor when writes to external memory are performed.

4.2.4 PATAG RAM

The ARM920T processor implements PATAG RAM to perform write-backs from the data cache. The physical address of all the lines held in the data cache is stored by the PATAG memory, removing the need for address translation when evicting a line from the cache.

4.2.5 MMUs

The standard ARM920T processor implements an enhanced ARM v4 memory management unit (MMU) to provide translation and access permission checks for the instruction and data address ports of the ARM9TDMI core.

The MMU features are:

- Standard ARM9™ v4 MMU mapping sizes, domains, and access protection scheme
- Mapping sizes are 1 Mbyte sections, 64 kbyte large pages, 4 kbyte small pages, and new 1 kbyte tiny pages
- Access permissions for sections
- Access permissions for large pages and small pages can be specified separately for each quarter of the page (these quarters are called subpages)
- 16 domains implemented in hardware
- 64-entry instruction TLB and 64-entry data TLB
- Hardware page table walks
- Round-robin replacement algorithm (also called cyclic)

4.2.6 System Controller

The system controller oversees the interaction between the instruction and data caches and the Bus Interface Unit. It controls internal arbitration between the blocks and stalls appropriate blocks when required.

The system controller arbitrates between instruction and data access to schedule single or simultaneous requests to the MMUs and the Bus Interface Unit. The system controller receives acknowledgement from each resource to allow execution to continue.

The physical address of all the lines held in the data cache is stored by the PATAG memory, removing the need for address translation when evicting a line from the cache.

4.2.7 Control Coprocessor (CP15)

The CP15 allows configuration of the caches, the write buffer, and other ARM920T processor options.

Several registers within CP15 are available for program control, providing access to features such as:

- Invalidate whole TLB using CP15
- Invalidate TLB entry, selected by modified virtual address, using CP15
- Independent lock-down of instruction TLB and data TLB using CP15 register 10
- Big or little-endian operation
- Low-power state
- Memory partitioning and protection
- Page table address
- Cache and TLB maintenance operations

4.3 ARMv4T Architecture

The following sections summarize the registers and instruction sets of the ARMv4T architecture.

4.3.1 Registers

The ARM920T processor core consists of a 32-bit data path and associated control logic. This data path contains 31 general purpose registers, coupled to a full shifter, Arithmetic Logic Unit, and multiplier. At any one time 16 registers are visible to the user. The remainder are synonyms used to speed up exception processing. Register 15 is the Program Counter (PC) and can be used in all instructions to reference data relative to the current instruction. R14 holds the return address after a subroutine call. R13 is used (by software convention) as a stack pointer.

4.3.2 Modes and Exception Handling

All exceptions have banked registers for R14 and R13. After an exception, R14 holds the return address for exception processing. This address is used both to return after the exception is processed and to address the instruction that caused the exception.

R13 is banked across exception modes to provide each exception handler with a private stack pointer. The fast interrupt mode also banks registers 8 to 12 so that interrupt processing can begin without the need to save or restore these registers.

A seventh processing mode, System mode, does not have any banked registers. It uses the User mode registers. System mode runs tasks that require a privileged processor mode and allows them to invoke all classes of exceptions.

4.3.3 Status Registers

All other processor states are held in status registers. The current operating processor status is in the Current Program Status Register (CPSR). The CPSR holds:

- Four ALU flags (Negative, Zero, Carry, and Overflow)
- Two interrupt disable bits (one for each type of interrupt)
- A bit to indicate ARM9 or Thumb execution
- Five bits to encode the current processor mode

All five exception modes also have a Saved Program Status Register (SPSR) that holds the CPSR of the task immediately before the exception occurred.

4.3.4 Exception Types

ARM9TDMI core supports five types of exception, and a privileged processing mode for each type. The types of exceptions are:

- Fast interrupt (FIQ)
- Normal interrupt (IRQ)
- Memory aborts (used to implement memory protection or virtual memory)
- Attempted execution of an undefined instruction
- Software interrupts (SWIs)

4.3.5 Conditional Execution

All ARM9 instructions (with the exception of BLX) are conditionally executed. Instructions optionally update the four condition code flags (Negative, Zero, Carry, and Overflow) according to their result. Subsequent instructions are conditionally executed according to the status of flags. Fifteen conditions are implemented.

4.4 Four Classes of Instructions

The ARM9 and Thumb instruction sets can be divided into four broad classes of instruction:

- Data processing instructions
- Load and store instructions
- Branch instructions
- Coprocessor instructions

4.4.1 Data Processing Instructions

The data processing instructions operate on data held in general purpose registers. Of the two source operands, one is always a register. The other has two basic forms:

- An immediate value
- A register value optionally shifted

If the operand is a shifted register, the shift amount might have an immediate value or the value of another register. Four types of shift can be specified. Most data processing instructions can perform a shift followed by a logical or arithmetic operation. Multiply instructions come in two classes:

- Normal, 32-bit result
- Long, 32-bit result variants

Both types of multiply instruction can optionally perform an accumulate operation.

4.4.2 Load and Store Instructions

The second class of instruction is load and store instructions. These instructions come in two main types:

- Load or store the value of a single register
- Load and store multiple register values

Load and store single register instructions can transfer a 32-bit word, a 16-bit halfword and an 8-bit byte between memory and a register. Byte and halfword loads might be automatically zero extended or sign extended as they are loaded. Swap instructions perform an atomic load and store as a synchronization primitive.

4.4.2.1 Addressing Modes

Load and store instructions have three primary addressing modes:

- Offset
- Pre-indexed
- Post-indexed

They are formed by adding or subtracting an immediate or register based offset to or from a base register. Register-based offsets can also be scaled with shift operations. Pre-indexed and post-indexed addressing modes update the base register with the base plus offset calculation. As the PC is a general purpose register, a 32-bit value can be loaded directly into the PC to perform a jump to any address in the 4 Gbyte memory space.

4.4.2.2 Block Transfers

Load and store multiple instructions perform a block transfer of any number of the general purpose registers to or from memory. Four addressing modes are provided:

- Pre-increment addressing
- Post-increment addressing
- Pre-decrement addressing
- Post-decrement addressing

The base address is specified by a register value (that can be optionally updated after the transfer). As the subroutine return address and the PC values are in general-purpose registers, very efficient subroutine calls can be constructed.

4.4.3 Branch Instructions

As well as allowing any data processing or load instruction to change control flow (by writing the PC) a standard branch instruction is provided with 24-bit signed offset, allowing forward and backward branches of up to 32 Mbyte.

4.4.3.1 Branch with Link

There is a Branch with Link (BL) that allows efficient subroutine calls. BL preserves the address of the instruction after the branch in R14 (the Link Register, or LR). This allows a move instruction to put the LR in to the PC and return to the instruction after the branch.

The third type of branch (BX and BLX) switches between ARM9 and Thumb instruction sets optionally with the return address preserving link option.

4.4.4 Coprocessor Instructions

There are three types of coprocessor instructions:

- Coprocessor data processing instructions invoke a coprocessor-specific internal operation
- Coprocessor register transfer instructions allow a coprocessor value to be transferred to or from an ARM920T processor register
- Coprocessor data transfer instructions transfer coprocessor data to or from memory, where the ARM920T calculates the address of the transfer

4.5 The ARM9 Instruction Set

The instruction set used by the ARM920T processor is summarized in Table 4-1.

Table 4-1. ARM920T Instruction Set

Mnemonic	Operation	Mnemonic	Operation
MOV	Move	MVN	Move Not
ADD	Add	ADC	Add with Carry
SUB	Subtract	SBC	Subtract with Carry
RSB	Reverse Subtract	RSC	Reverse Subtract with Carry
CMP	Compare	CMN	Compare Negated
TST	Test	TEQ	Test Equivalence
AND	Logical AND	BIC	Bit Clear
FOR	Logical Exclusive OR	ORR	Logical (inclusive) OR
MUL	Multiply	MLA	Multiply Accumulate
SMULL	Sign Long Multiply	SMLAL	Signed Long Multiply Accumulate
UMULL	Unsigned Long Multiply	UMLAL	Unsigned Long Multiply Accumulate
CLZ	Count Leading Zeroes	BKPT	Breakpoint

Table 4-1. ARM920T Instruction Set (continued)

Mnemonic	Operation	Mnemonic	Operation
MRS	Move From Status Register	MSR	Move to Status Register
B	Branch	–	–
BL	Branch and Link	BLX	Branch and Link and Exchange
BX	Branch and Exchange	SWI	Software Interrupt
LDR	Load Word	STR	Store Word
LDRH	Load Halfword	STRH	Store Halfword
LDRB	Load Byte	STRB	Store Byte
LDRSH	Load Signed Halfword	LDRSB	Load Signed Byte
LDMIA	Load Multiple	STMIA	Store Multiple
SWP	Swap Word	SWPB	Swap Byte
CDP	Coprocessor Data Processing	–	–
MRC	Move From Coprocessor	MCR	Move to Coprocessor
LDC	Load To Coprocessor	STC	Store From Coprocessor

4.6 The ARM Thumb Instruction Set

The ARM Thumb instruction set is summarized in Table 4-2.

Table 4-2. ARM Thumb Instruction Set

Mnemonic	Operation	Mnemonic	Operation
MOV	Move	MVN	Move Not
ADD	Add	ADC	Add with Carry
SUB	Subtract	SBC	Subtract with Carry
RSB	Reverse Subtract	RSC	Reverse Subtract with Carry
CMP	Compare	CMN	Compare Negated
TST	Test	NEG	Negate
AND	Logical AND	BIC	Bit Clear
FOR	Logical Exclusive OR	ORR	Logical (inclusive) OR
LSL	Logical Shift Left	LSR	Logical Shift Right
ASR	Arithmetic Shift Right	ROR	Rotate Right
MUL	Multiply	BKPT	Breakpoint
B	Unconditional Branch	Bcc	Conditional Branch
BL	Branch and Link	BLX	Branch and Link and Exchange

Table 4-2. ARM Thumb Instruction Set (continued)

Mnemonic	Operation	Mnemonic	Operation
BX	Branch and Exchange	SWI	Software Interrupt
LDR	Load Word	STR	Store Word
LDRH	Load Halfword	STRH	Store Halfword
LDRB	Load Byte	STRB	Store Byte
LDRSH	Load Signed Halfword	LDRSB	Load Signed Byte
LDMIA	Load Multiple	STMIA	Store Multiple
PUSH	Push Registers to stack	POP	Pop Registers from stack

4.6.1 ARM920T Modes and Registers

The modes and registers of the ARM920T processor are shown in Table 4-3.

Table 4-3. Register Availability by Mode

User and System Modes	Supervisor Mode	Abort Mode	Undefined Mode	Interrupt Mode	Fast Interrupt Mode
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8_FIQ
R9	R9	R9	R9	R9	R9_FIQ
R10	R10	R10	R10	R10	R10_FIQ
R11	R11	R11	R11	R11	R11_FIQ
R12	R12	R12	R12	R12	R12_FIQ
R13	R13_SVC	R13_ABORT	R13_UNDEF	R13_IRQ	R13_FIQ
R14	R14_SVC	R14_ABORT	R14_UNDEF	R14_IRQ	R14_FIQ
PC	PC	PC	PC	PC	PC
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_SVC	SPSR_ABORT	SPSR_UNDEF	SPSR_IRQ	SPSR_FIQ
	= Mode-specific banked registers				

Chapter 5
Embedded Trace Macrocell (ETM)

The MC9328MXL is equipped with an ARM9 Embedded Trace Macrocell™ (ETM9) module for real time debugging which is a great help to a system designer because the MC9328MXL is a highly integrated processor, a very limited number of pins are available for debug purposes. ETM signals in MC9328MXL are multiplexed with other function pins. This chapter contains a brief summary of the ETM features, for details of ETM operation, please refer to the *ETM9 Technical Reference Manual* Rev.2a (ARM Limited: 2001, order number DDI0157E).

5.1 Introduction to the ETM

The ETM provides instruction and data trace for the ARM9™ family of microprocessors. This document describes the interface between an ARM Thumb® family processor and the ETM. For details of the interface between an ARM7™ processor and ETM7, refer to the *ETM7 Technical Reference Manual* Rev.1 (ARM Limited: 2001, order number DDI0158D). The block diagram of the ETM is shown in Figure 5-1.

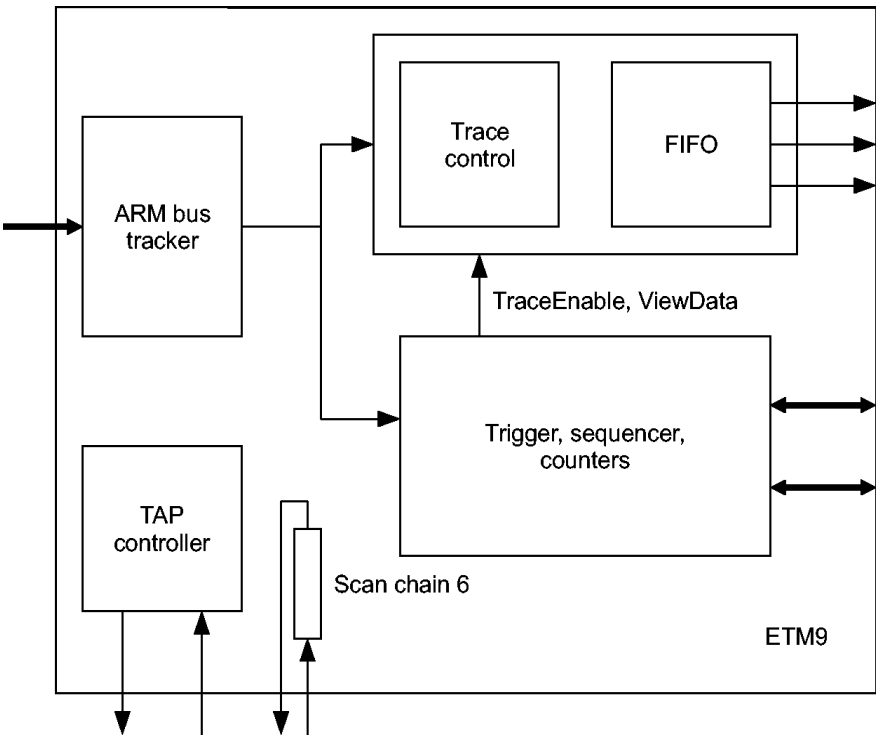


Figure 5-1. ETM Block Diagram

5.2 Programming and Reading ETM Registers

All registers in the ETM9 are programmed through a JTAG interface. The interface is an extension of the ARM920T processor's TAP controller, and is assigned scan chain 6. The scan chain consists of a 40-bit shift register comprising:

- 32-bit data field
- 7-bit address field
- A read/write bit

The data to be written is scanned into the 32-bit data field, the address of the register into the 7-bit address field, and 1 into the read/write bit.

A register is read by scanning its address into the address field and a 0 into the read/write bit. The 32-bit data field is ignored. A read or a write takes place when the TAP controller enters the UPDATE-DR state.

For further details of ETM registers, see the Embedded Trace Macrocell specification.

5.3 Pin Configuration for ETM

The ETM module uses 13 pins on the MC9328MXL. These pins are multiplexed with other functions on the device, and must be configured for ETM operation. Table 5-1 identifies the pin configuration, however, only the 5 pins of the 13 that are multiplexed are shown.

NOTE:

The user must ensure that the data direction bits in the GPIO are set to the correct direction for proper operation. See Section 29.5.1, "Data Direction Registers," on page 29-8 for details.

Table 5-1. ETM Pin Configuration

Pin	Setting	Configuration Procedure
ETMTRACESYNC	Alternate function of GPIO Port A [0]	Clear bit 0 of Port A GPIO In Use Register (GIUS_A) Set bit 0 of Port A General Purpose Register (GPR_A)
ETMTRACECLK	Alternate function of GPIO Port A [31]	Clear bit 31 of Port A GPIO In Use Register (GIUS_A) Set bit 31 of Port A General Purpose Register (GPR_A)
ETMPIPESTAT [2:0]	Alternate function of GPIO Port A [30:28]	Clear bits [30:28] of Port A GPIO In Use Register (GIUS_A) Set bits [30:28] of Port A General Purpose Register (GPR_A)
ETMTRACEPKT [7:4]	Alternate function of GPIO Port A [20:17]	Clear bits [20:17] of Port A GPIO In Use Register (GIUS_A) Set bits [20:17] of Port A General Purpose Register (GPR_A)
ETMTRACEPKT [3:0]	Alternate function of GPIO Port A [27:24]	Clear bits [27:24] of Port A GPIO In Use Register (GIUS_A) Set bits [27:24] of Port A General Purpose Register (GPR_A)

Chapter 6
Reset Module

The reset module controls or distributes all of the system reset signals used by the MC9328MXL. This chapter provides a detailed description of the operation of this module.

6.1 Functional Description of the Reset Module

A simplified block diagram of the reset module is shown in Figure 6-1. The reset module generates two distinct events—a global reset and an ARM920T processor reset.

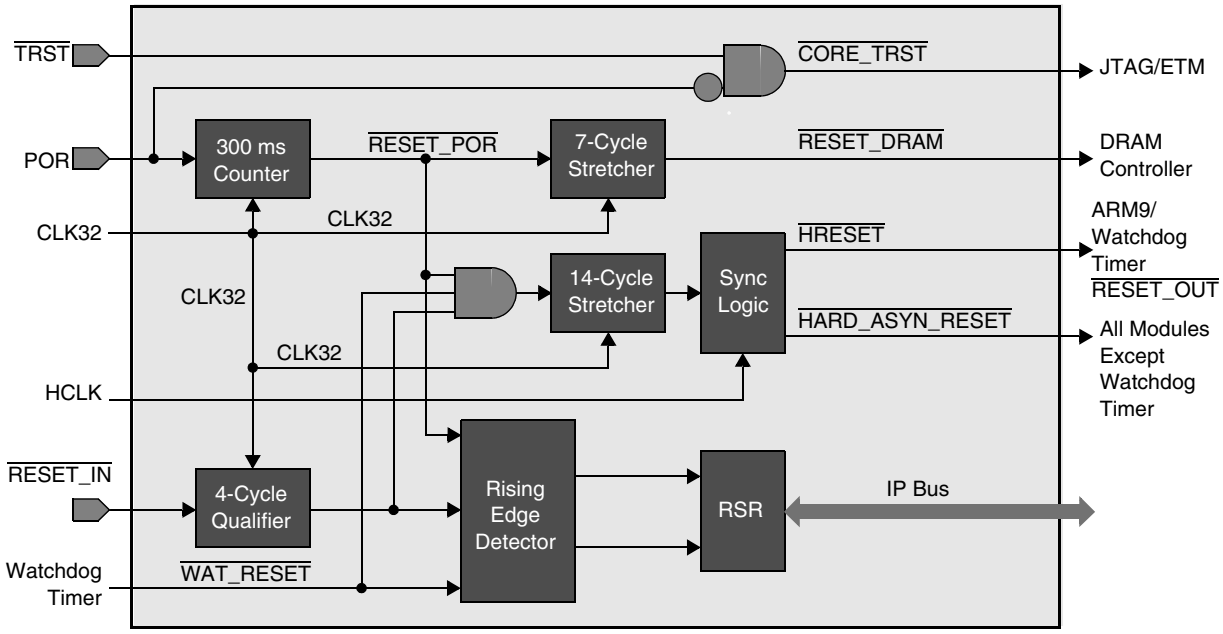


Figure 6-1. Reset Module Block Diagram

6.1.1 Global Reset

A global reset simultaneously asserts three reset signals: $\overline{\text{HRESET}}$, $\overline{\text{RESET_DRAM}}$, and $\overline{\text{CORE_TRST}}$. These signals remain asserted for 14 CLK32 cycles. The $\overline{\text{RESET_DRAM}}$ signal is deasserted 7 CLK32 cycles before $\overline{\text{HRESET}}$ and $\overline{\text{HARD_ASYN_RESET}}$. This 7-cycle period provides the DRAM with time to execute any necessary self-refresh operations. The timing diagram in Figure 6-2 on page 6-2 shows the relationship of the reset signal timings. See Table 6-1 for reset module signal and pin definitions.

There is one source capable of generating a global reset: A high condition on the POR pin for at least 4 × 32 kHz clocks when the 32 kHz crystal oscillator is running.

The following signal conditions are not capable of generating a global reset, however they reset the ARM920T core:

- An external qualified low condition on the $\overline{\text{RESET_IN}}$ pin
- A low condition on $\overline{\text{WAT_RESET}}$

NOTE:

Due to the asynchronous nature of the $\overline{\text{RESET}}$ signal, the time period required to qualify the signal may vary, and the $\overline{\text{HRESET}}$ timing relative to the rising edge of $\overline{\text{RESET}}$ is also affected. A $\overline{\text{RESET}}$ signal shorter than 3 CLK32 cycles will not be qualified, a $\overline{\text{RESET}}$ signal equal to or longer than 4 CLK32 cycles will always be qualified, and any period length that is more than 3 and less than 4 CLK32 cycles is undefined.

IMPORTANT:

POR is the reset signal for all the reset module flip-flops. For this reason, an external reset signal is qualified if it lasts more than 4 CLK32 cycles when POR is deasserted.

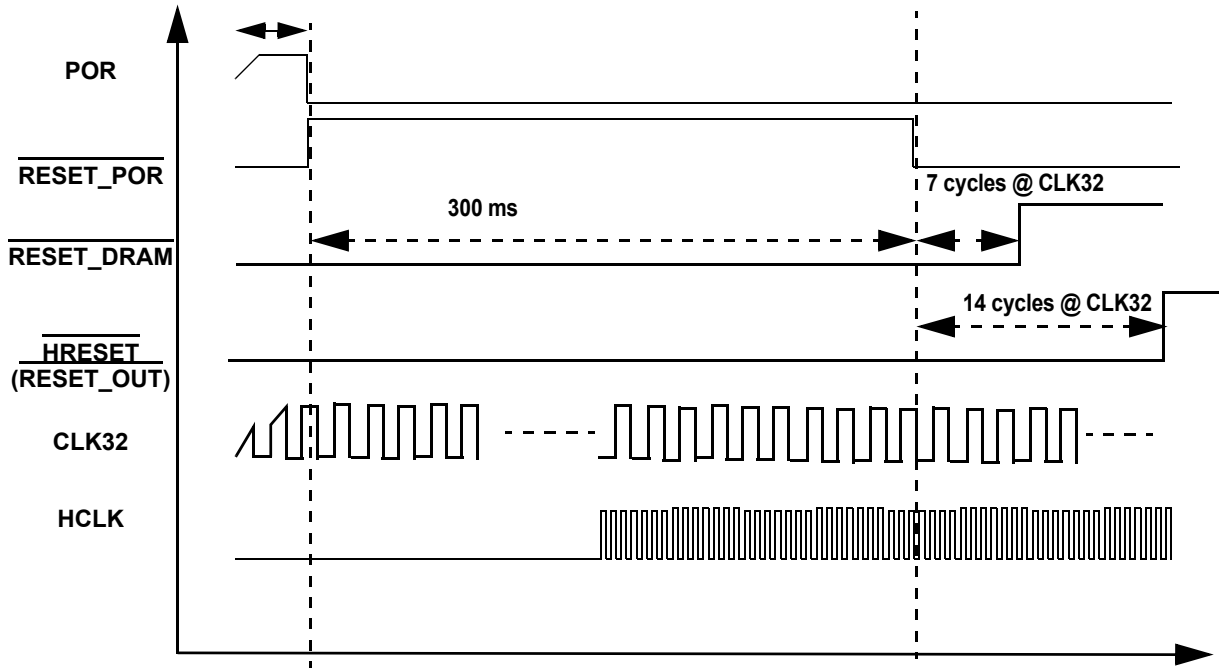


Figure 6-2. DRAM and Internal Reset Timing Diagram

6.1.2 ARM920T Processor Reset

Any qualified global reset signal resets the ARM920T processor and all related peripherals to their default state. After the internal reset is deasserted, the ARM920T processor begins fetching code from the internal bootstrap ROM, sync flash, or CS0 space. The memory location of the fetch depends on the configuration of the BOOT pins on the rising edge of $\overline{\text{HRESET}}$ (see Section 8.2, “System Boot Mode Selection,” on page 8-7).

Table 6-1. Reset Module Pin and Signal Descriptions

Signal Name	Direction	Signal Description
CLK32	IN	**32 kHz Clock**—A 32 kHz clock signal derived from the input crystal oscillator circuit in the PLL.
POR	IN	**Power-On Reset**—An internal active high Schmitt trigger signal from the POR pin. The POR signal is normally generated by an external RC circuit designed to detect a power-up event.
$\overline{\text{RESET}}$	IN	**Reset**—An external active low Schmitt trigger signal from the $\overline{\text{RESET_IN}}$ pin. When this signal goes active, all modules (except the reset module and the clock control module) are reset.
$\overline{\text{TRST}}$	IN	**Test Reset Pin**—An external active low signal from the $\overline{\text{TRST}}$ pin. The Test Reset Pin is used to asynchronously initialize the JTAG controller.
$\overline{\text{WAT_RESET}}$	IN	**Watchdog Timer Reset**—An active low signal generated by the watchdog timer when a time-out period has expired.
$\overline{\text{CORE_TRST}}$	OUT	**Core Test Reset**—An active low signal that resets the JTAG module and the ETM.
$\overline{\text{HARD_ASYN_RESET}}$	OUT	**Hard Asynchronous Reset**—An active low signal that resets all peripheral modules except the watchdog timer module. The rising edge of this signal is synchronous with HCLK.
$\overline{\text{HRESET}}$	OUT	**Hard Reset**—An active low signal that resets the ARM920T processor and the watchdog timer module.This signal is deasserted during the low phase of HCLK. This signal also appears on the $\overline{\text{RESET_OUT}}$ pin of the MC9328MXL.
$\overline{\text{RESET_DRAM}}$	OUT	**DRAM Reset**—An active low signal that resets the DRAM controller.

6.2 Programming Model

The Reset Source Register (RSR), the only register in the reset module, can be written to or read by the ARM920T processor through the IP bus interface.

6.2.1 Reset Source Register (RSR)

The Reset Source Register is a 16-bit read-only register used by the ARM920T processor to determine the source of the last MC9328MXL. hardware reset. The source of the last hardware reset is defined in Table 6-2 and Table 6-3 on page 6-4.

If several sources' signals overlap and if the signals are released during the same CLK32 cycle (which also causes the assertion of the $\overline{\text{RESET_OUT}}$ signal), only the highest-priority event is registered by the RSR using the following priority order:

1. POR signal
2. Qualified external reset signal
3. Watchdog signal

Otherwise, the last signal that is released is honored.

RSR Reset Source Register **Addr 0x0021B800**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															WDR	EXR
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 6-2. RSR Register Description

Name	Description	Settings
Reserved Bits 31–2	Reserved—These bits are reserved and should read 0.	
WDR Bit 1	**Watchdog Reset**—Indicates whether the last reset was caused by a Watchdog count expiration.	0 = Reset was NOT a Watchdog count expiration 1 = Reset WAS a Watchdog count expiration
EXR Bit 0	**External Reset**—Indicates whether the last reset was caused by a $\overline{\text{RESET_IN}}$ pin assertion.	0 = Reset was NOT a $\overline{\text{RESET_IN}}$ pin assertion 1 = Reset WAS a $\overline{\text{RESET_IN}}$ pin assertion

Table 6-3. Hardware Reset Source Matrix

Source	WDR	EXR
POR	0	0
Qualified external reset	0	1
Watchdog time-out	1	0

Chapter 7
AHB to IP Bus Interface (AIPI)

7.1 Overview

This chapter provides an overview of the R-AHB to IP bus interface (AIPI). The AIPI module in the MC9328MXL acts as an interface between the R-AHB (Reduced ARM Advanced High-performance Bus) and lower bandwidth peripherals.

7.1.1 Features

The AIPI provides the following features:

- All peripheral read transactions require a minimum of two system clocks (R-AHB side) and all write transactions require a minimum of three system clocks (R-AHB side).
- Support of 8-bit, 16-bit, and 32-bit IP bus peripherals.
- Byte, half word, and word read and write are supported to and from each peripheral in both big and little endian mode.
- Support of multi-cycle accesses (16-bit operations to 8-bit peripherals, and 32-bit operations to 16-bit and 8-bit peripherals).
- Ability to restrict user to limit access to peripherals in their natural size only.
- Support of 15 external IP bus peripherals. Muxiplexers are incorporated to support the 15 separate read data buses, and the transfer wait and transfer error from peripherals.
- A watchdog timer is provided to time-out peripheral access if operation does not terminate with 512 clock cycles.
- Use of a single asynchronous reset and one global clock with both edges.
- The AIPI module is implemented using MUX-D scan methodology for testability.

7.1.2 General Information

The AIPI is the interface between the R-AHB and on-chip IP bus peripherals as shown in Figure 7-1 on page 7-2.

IP bus peripherals are modules that contain readable/writable control and status registers. The R-AHB master reads and writes these registers through the AIPI. The AIPI generates module enables, the module address, transfer attributes, byte enables and write data as inputs to the IP bus peripherals. The AIPI captures read data (qualified by IPS_XFR_WAIT) from the IP bus interface and drives it on the R-AHB. The AIPI module terminates the transfer by asserting AIPI_HREADY_OUT.

The register maps of all IP bus peripherals are located on 4096 byte boundaries. Each IP bus peripheral is allocated one 4-kbyte block (minimum block size) of the memory map, configured as 1024 32-bit internal registers (or 2048 16-bit internal registers, or 4096 8-bit internal registers), activated by one of 15 module enables from the AIPI. Up

to 15 IP bus peripherals may be implemented, occupying contiguous blocks of 4 kbytes, for a total of 60 kbytes. The exact address assignment for the IP bus peripherals is system dependent, and is defined in the system specification. Each IP bus peripheral will select its internal registers based on the address driven on the IPS_ADDR signals.

The AIPI is responsible for telling the IP bus peripherals if the access is in supervisor or user mode. The AIPI may block user mode accesses to certain IP bus peripherals or it may allow the individual IP bus peripherals to determine if user mode accesses are allowed. Please see Section 7.2, “Programming Model,” for more information.

The AIPI supports multi-cycle accesses to IP bus peripherals when the R-AHB master requests data transfers that are larger than the targeted IP bus peripheral’s data bus width. Table 7-1 through Table 7-4 provides more information on both single-cycle and multi-cycle accesses. For data access that are larger than the target IP bus peripheral, the AIPI will duplicated the data across all the byte lanes on the AHB, i.e. for a word read from 8 bit peripheral, the same data read will appear on byte lanes [31:24, [23:16], [15:8] and [7:0]. Similarly for a byte write to the peripheral, the core will duplicate the same byte over the byte lanes of the AHB for the write operation.

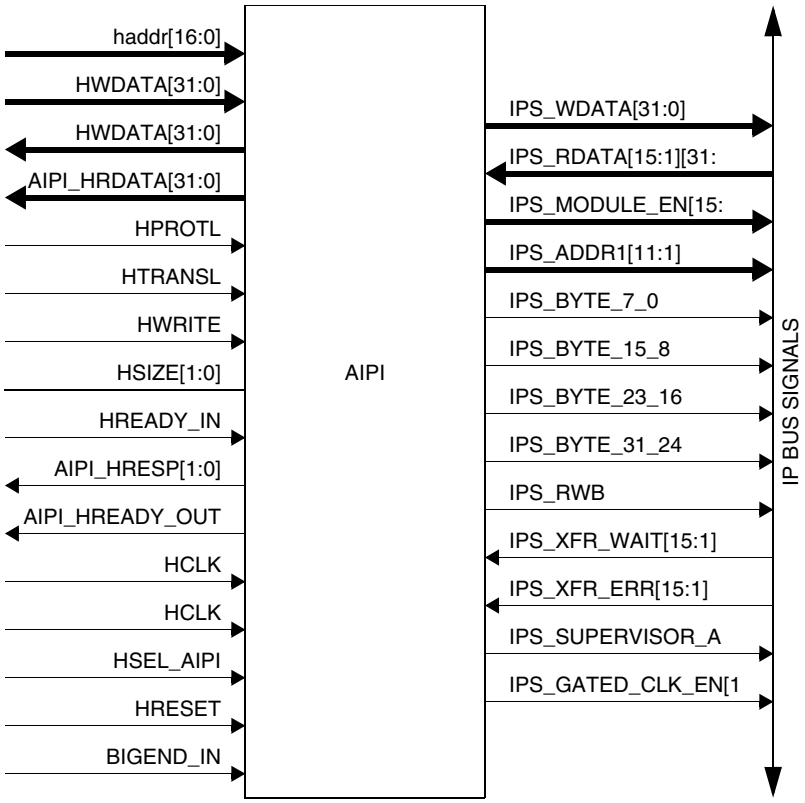


Figure 7-1. AIPI Interface

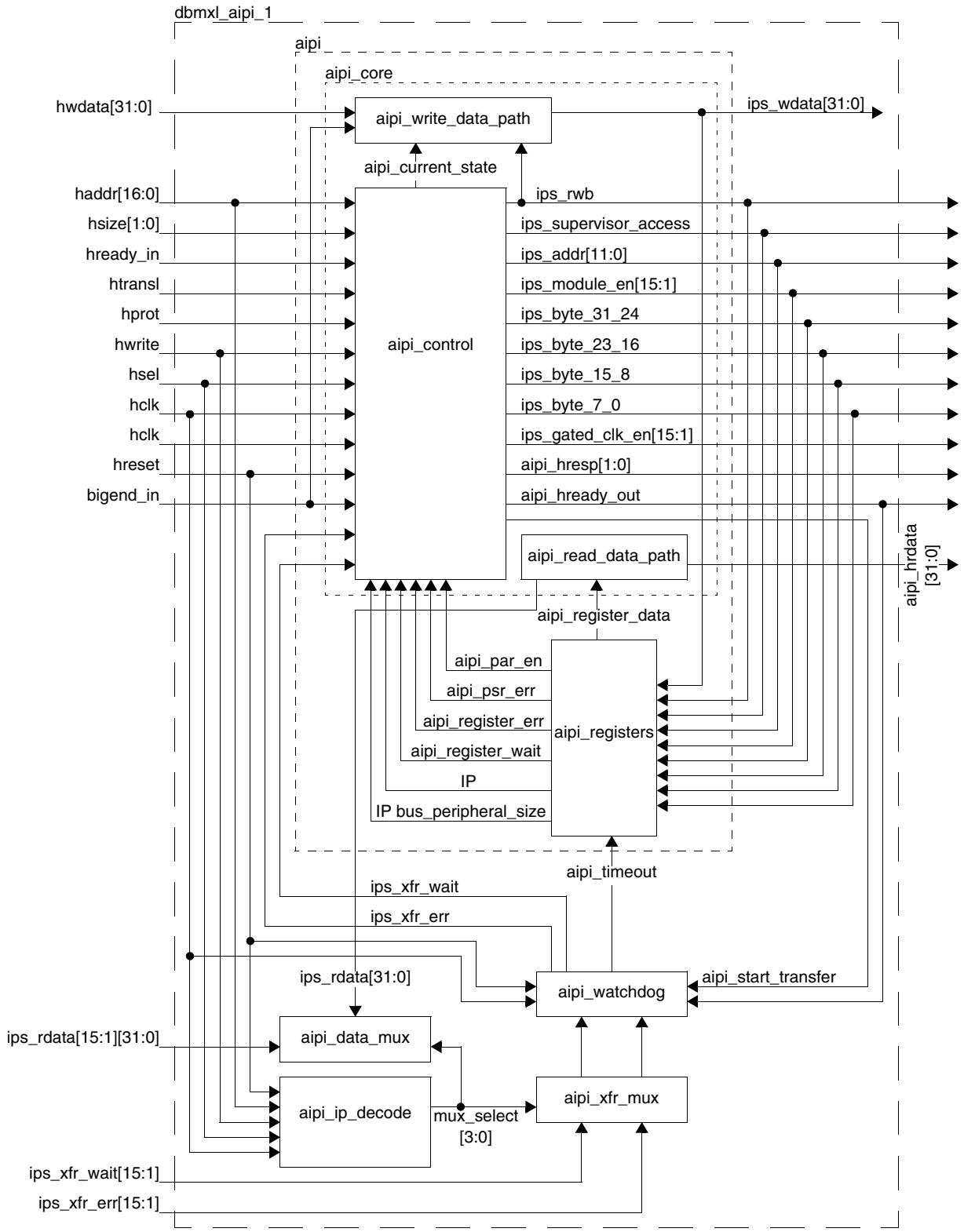


Figure 7-2. Block Diagram of the AIPI Module

Table 7-1. R-AHB to IP Bus Interface Operation (Big Endian—Read Operation)

Transfer Size	haddr		IP Bus Size	ips_addr		Active Bus Section (IP Bus to R-AHB)			
	[1]	[0]		[1]	[0]	R-AHB [31:24]	R-AHB [23:16]	R-AHB [15:8]	R-AHB [7:0]
Byte	0	0	8-bit	0	0	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
	0	1		0	1	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
	1	0		1	0	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
	1	1		1	1	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
	0	0	16-bit	0	X	ips_rdata[15:8]	ips_rdata[7:0]	–	–
	0	1				ips_rdata[15:8]	ips_rdata[7:0]	–	–
	1	0		1	X	–	–	ips_rdata[15:8]	ips_rdata[7:0]
	1	1				–	–	ips_rdata[15:8]	ips_rdata[7:0]
	0	0	32-bit	X	X	ips_rdata[31:24]	–	–	–
	0	1				–	ips_rdata[23:16]	–	–
	1	0		X	X	–	–	ips_rdata[15:8]	–
	1	1				–	–	–	ips_rdata[7:0]
Half Word	0	NA	8-bit	0	0	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
					1	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
	1			1	0	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
					1	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
	0		16-bit	0	X	ips_rdata[15:8]	ips_rdata[7:0]	ips_rdata[15:8]	ips_rdata[7:0]
	1			1	X	ips_rdata[15:8]	ips_rdata[7:0]	ips_rdata[15:8]	ips_rdata[7:0]
	0		32-bit	X	X	ips_rdata[31:24]	ips_rdata[23:16]	–	–
	1			X	X	–	–	ips_rdata[15:8]	ips_rdata[7:0]
Word	NA	NA	8-bit	0	0	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
					1	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
				1	0	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
					1	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
			16-bit	0	X	ips_rdata[15:8]	ips_rdata[7:0]	ips_rdata[15:8]	ips_rdata[7:0]
				1	X	ips_rdata[15:8]	ips_rdata[7:0]	ips_rdata[15:8]	ips_rdata[7:0]
			32-bit	X	X	ips_rdata[31:24]	ips_rdata[23:16]	ips_rdata[15:8]	ips_rdata[7:0]

Table 7-2. R-AHB to IP Bus Interface Operation (Big Endian—Write Operation)

Transfer Size	haddr		IP Bus Size	ips_addr		Active Bus Section (R-AHB to IP Bus)			
	[1]	[0]		[1]	[0]	R-AHB [31:24]	R-AHB [23:16]	R-AHB [15:8]	R-AHB [7:0]
Byte	0	0	8-bit	0	0	ips_wdata[7:0]	–	–	–
	0	1		0	1	–	ips_wdata[7:0]	–	–
	1	0		1	0	–	–	ips_wdata[7:0]	–
	1	1		1	1	–	–	–	ips_wdata[7:0]
	0	0	16-bit	0	X	ips_wdata[15:8]	–	–	–
	0	1				–	ips_wdata[7:0]	–	–
	1	0		1	X	–	–	ips_wdata[15:8]	–
	1	1				–	–	–	ips_wdata[7:0]
	0	0	32-bit	X	X	ips_wdata[31:24]	–	–	–
	0	1				–	ips_wdata[23:16]	–	–
	1	0		X	X	–	–	ips_wdata[15:8]	–
	1	1				–	–	–	ips_wdata[7:0]
Half Word	0	NA	8-bit	0	0	ips_wdata[7:0]	–	–	–
					1	–	ips_wdata[7:0]	–	–
	1			1	0	–	–	ips_wdata[7:0]	–
					1	–	–	–	ips_wdata[7:0]
	0		16-bit	0	X	ips_wdata[15:8]	ips_wdata[7:0]	–	–
	1			1	X	–	–	ips_wdata[15:8]	ips_wdata[7:0]
	0		32-bit	X	X	ips_wdata[31:24]	ips_wdata[23:16]	–	–
	1			X	X	–	–	ips_wdata[15:8]	ips_wdata[7:0]
Word	NA	NA	8-bit	0	0	ips_wdata[7:0]	–	–	–
					1	–	ips_wdata[7:0]	–	–
				1	0	–	–	ips_wdata[7:0]	–
					1	–	–	–	ips_wdata[7:0]
			16-bit	0	X	ips_wdata[15:8]	ips_wdata[7:0]	–	–
				1	X	–	–	ips_wdata[15:8]	ips_wdata[7:0]
			32-bit	X	X	ips_wdata[31:24]	ips_wdata[23:16]	ips_wdata[15:8]	ips_wdata[7:0]

Table 7-3. R-AHB to IP Bus Interface Operation (Little Endian—Read Operation)

Transfer Size	haddr		IP Bus Size	ips_addr		Active Bus Section (IP Bus to R-AHB)			
	[1]	[0]		[1]	[0]	R-AHB [31:24]	R-AHB [23:16]	R-AHB [15:8]	R-AHB [7:0]
Byte	0	0	8-bit	0	0	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
	0	1		0	1	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
	1	0		1	0	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
	1	1		1	1	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
	0	0	16-bit	0	X	–	–	ips_rdata[15:8]	ips_rdata[7:0]
	0	1				–	–	ips_rdata[15:8]	ips_rdata[7:0]
	1	0		1	X	ips_rdata[15:8]	ips_rdata[7:0]	–	–
	1	1				ips_rdata[15:8]	ips_rdata[7:0]	–	–
	0	0	32-bit	X	X	–	–	–	ips_rdata[7:0]
	0	1				–	–	ips_rdata[15:8]	–
	1	0		X	X	–	ips_rdata[23:16]	–	–
	1	1				ips_rdata[31:24]	–	–	–
Half Word	0	NA	8-bit	0	0	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
					1	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
	1			1	0	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
					1	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
	0		16-bit	0	X	ips_rdata[15:8]	ips_rdata[7:0]	ips_rdata[15:8]	ips_rdata[7:0]
	1			1	X	ips_rdata[15:8]	ips_rdata[7:0]	ips_rdata[15:8]	ips_rdata[7:0]
	0		32-bit	X	X	–	–	ips_rdata[15:8]	ips_rdata[7:0]
	1			X	X	ips_rdata[31:24]	ips_rdata[23:16]	–	–
Word	NA	NA	8-bit	0	0	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
					1	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
				1	0	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
					1	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]	ips_rdata[7:0]
			16-bit	0	X	ips_rdata[15:8]	ips_rdata[7:0]	ips_rdata[15:8]	ips_rdata[7:0]
				1	X	ips_rdata[15:8]	ips_rdata[7:0]	ips_rdata[15:8]	ips_rdata[7:0]
			32-bit	X	X	ips_rdata[31:24]	ips_rdata[23:16]	ips_rdata[15:8]	ips_rdata[7:0]

Table 7-4. R-AHB to IP Bus Interface Operation (Little Endian—Write Operation)

Transfer Size	haddr		IP Bus Size	ips_addr		Active Bus Section (R-AHB to IP Bus)			
	[1]	[0]		[1]	[0]	R-AHB [31:24]	R-AHB [23:16]	R-AHB [15:8]	R-AHB [7:0]
Byte	0	0	8-bit	0	0	–	–	–	ips_wdata[7:0]
	0	1		0	1	–	–	ips_wdata[7:0]	–
	1	0		1	0	–	ips_wdata[7:0]	–	–
	1	1		1	1	ips_wdata[7:0]	–	–	–
	0	0	16-bit	0	X	–	–	–	ips_wdata[7:0]
	0	1				–	–	ips_wdata[15:8]	–
	1	0		1	X	–	ips_wdata[7:0]	–	–
	1	1				ips_wdata[15:8]	–	–	–
	0	0	32-bit	X	X	–	–	–	ips_wdata[7:0]
	0	1				–	–	ips_wdata[15:8]	–
	1	0		X	X	–	ips_wdata[23:16]	–	–
	1	1				ips_wdata[31:24]	–	–	–
Half Word	0	NA	8-bit	0	0	–	–	–	ips_wdata[7:0]
					1	–	–	ips_wdata[7:0]	–
	1			1	0	–	ips_wdata[7:0]	–	–
					1	ips_wdata[7:0]	–	–	–
	0		16-bit	0	X	–	–	ips_wdata[15:8]	ips_wdata[7:0]
	1			1	X	ips_wdata[15:8]	ips_wdata[7:0]	–	–
	0		32-bit	X	X	–	–	ips_wdata[15:8]	ips_wdata[7:0]
	1			X	X	ips_wdata[31:24]	ips_wdata[23:16]	–	–
Word	NA	NA	8-bit	0	0	–	–	–	ips_wdata[7:0]
					1	–	–	ips_wdata[7:0]	–
				1	0	–	ips_wdata[7:0]	–	–
					1	ips_wdata[7:0]	–	–	–
			16-bit	0	X	–	–	ips_wdata[15:8]	ips_wdata[7:0]
				1	X	ips_wdata[15:8]	ips_wdata[7:0]	–	–
			32-bit	X	X	ips_wdata[31:24]	ips_wdata[23:16]	ips_wdata[15:8]	ips_wdata[7:0]

7.2 Programming Model

There are six registers that reside inside the AIPI module. Two system clocks are required for read accesses and three system clocks are required for write accesses to the AIPI registers. Table 7-5 is a summary of these registers and their addresses.

Table 7-5. AIPI Module Register Memory Map

Description	Name	Address
AIPI1		
AIPI1 Peripheral Size Register 0	PSR0_1	0x00200000
AIPI1 Peripheral Size Register 1	PSR1_1	0x00200004
AIPI1 Peripheral Access Register	PAR_1	0x00200008
AIPI1 Peripheral Control Register	PCR_1	0x0020000C
AIPI1 Time-Out Status Register	TSR_1	0x00200010
AIPI2		
AIPI2 Peripheral Size Register 0	PSR0_2	0x00210000
AIPI2 Peripheral Size Register 1	PSR1_2	0x00210004
AIPI2 Peripheral Access Register	PAR_2	0x00210008
AIPI2 Peripheral Control Register	PCR_2	0x0021000C
AIPI2Time-Out Status Register	TSR_2	0x00210010

Table 7-6 illustrates the peripheral address associated with the corresponding module_en number. Refer to Chapter 3, “Memory Map,” to see the corresponding address assigned to each peripheral.

Table 7-6. Peripheral Address MODULE_EN Numbers

AIPI 1		AIPI 2	
Address	**MODULE_EN**	**Address**	**MODULE_EN**
0x0020 1000 – 0x0020 1FFF	1	0x0021 1000 – 0x0021 1FFF	1
0x0020 2000 – 0x0020 2FFF	2	0x0021 2000 – 0x0021 2FFF	2
0x0020 3000 – 0x0020 3FFF	3	0x0021 3000 – 0x0021 3FFF	3
0x0020 4000 – 0x0020 4FFF	4	0x0021 4000 – 0x0021 4FFF	4
0x0020 5000 – 0x0020 5FFF	5	0x0021 5000 – 0x0021 5FFF	5
0x0020 6000 – 0x0020 6FFF	6	0x0021 6000 – 0x0021 6FFF	6
0x0020 7000 – 0x0020 7FFF	7	0x0021 7000 – 0x0021 7FFF	7
0x0020 8000 – 0x0020 8FFF	8	0x0021 8000 – 0x0021 8FFF	8
0x0020 9000 – 0x0020 9FFF	9	0x0021 9000 – 0x0021 9FFF	9
0x0020 A000 – 0x0020 AFFF	10	0x0021 A000 – 0x0021 AFFF	10
0x0020 B000 – 0x0020 BFFF	11	0x0021 B000 – 0x0021 BFFF	11

Table 7-6. Peripheral Address MODULE_EN Numbers (continued)

AIPI 1		AIPI 2	
Address	**MODULE_EN**	**Address**	**MODULE_EN**
0x0020 C000 – 0x0020 CFFF	12	0x0021 C000 – 0x0021 CFFF	12
0x0020 D000 – 0x0020 DFFF	13	0x0021 D000 – 0x0021 DFFF	13
0x0020 E000 – 0x0020 EFFF	14	0x0021 E000 – 0x0021 EFFF	14
0x0020 F000 – 0x0020 FFFF	15	0x0021 F000 – 0x0021 FFFF	15

7.2.1 Peripheral Size Registers[1:0]

These registers control the size of the IP bus peripheral in each IP bus peripheral location. Peripheral locations that are not occupied must have their corresponding bits in the PSRs (Peripheral Size Registers) programmed to 1 in each register.

The least significant bit in the PSRs is a read only bit as it governs the AIPI registers themselves. They are set and cleared appropriately to indicate the registers are 32 bits. Bits 31 through 16 in both registers are preset to 1 and the fields are reserved and can only be read.

7.2.1.1 AIPI1 Peripheral Size Register 0 and AIPI2 Peripheral Size Register 0

		Addr
PSR0_1	AIPI1 Peripheral Size Register 0	**0x00200000**
PSR0_2	AIPI2 Peripheral Size Register 0	**0x00210000**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE																
PSR0_1 RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
								0xFFFF								
PSR0_2 RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
								0xFFFF								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MOD_EN_L															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r
PSR0_1 RESET	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0
								0xFC00								
PSR0_2 RESET	1	1	1	0	0	1	0	0	0	1	1	1	0	0	1	0
								0xE472								

Table 7-7. AIPI1 Peripheral Size Register 0 and AIPI2 Peripheral Size Register 0 Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 1.	
MOD_EN_L Bits 15–1	**Module_En (lower)**—Each bit represents the lower bit of the 2-bit field (PSR1 + PSR0) that represents the Module_En number.	See Table 7-9 for bit settings
Reserved Bit 0	Reserved—This bit is reserved and should read 0.	

7.2.1.2 AIPI1 Peripheral Size Register 1 and AIPI2 Peripheral Size Register 1

		Addr
PSR1_1	AIPI1 Peripheral Size Register 1	**0x00200004**
PSR1_2	AIPI2 Peripheral Size Register 1	**0x00210004**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE																
PSR_1 RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
								0xFFFF								
PSR_2 RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
								0xFFFF								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MOD_EN_U															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r
PSR_1 RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
								0xFFFF								
PSR_2 RESET	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1
								0xFBEF								

Table 7-8. AIPI1 Peripheral Size Register 1 and AIPI2 Peripheral Size Register 1 Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 1.	
MOD_EN_U Bits 15–1	**Module_En (upper)**—Each bit represents the upper bit of the 2-bit field (PSR1 + PSR0) that represents the Module_En number.	See Table 7-9 for bit settings
Reserved Bit 0	Reserved—This bit is reserved and should read 0.	

The PSRs work together to indicate the size of the IP bus peripheral occupying the corresponding **ips_module_en** location, or to indicate there is no IP bus peripheral occupying the corresponding **ips_module_en** location. A good example of how the PSRs work is the AIPI registers themselves. When haddr[16:12] is decoded to select the AIPI registers, {PSR1[bit0], PSR0[bit0]} returns a value of 10, indicating that the AIPI registers are word width registers. Table 7-9 shows how to program the PSR registers based on the size or availability of an IP bus peripheral.

Table 7-9. PSR Data Bus Size Encoding

PSR[1:0] Bits		IP Bus Peripheral Size [x] (module_en [x])
PSR1[x]	PSR0[x]	
0	0	8-bit
0	1	16-bit
1	0	32-bit
1	1	Unoccupied

7.2.2 Peripheral Access Registers

These registers are used to tell the AIPI whether or not the IP bus peripheral corresponding to the bit location in the register may be accessed in user mode. If the peripheral may be accessed in supervisor mode only and a user mode access is attempted an abort will be generated and no IP bus activity occurs. If the peripheral can be accessed in user mode, then the IPS_SUPERVISOR_ACCESS bit reflects whether the attempted access is in supervisor or user mode and the peripheral itself can decide whether to accept a user access (if one is attempted) or issue an error response.

The least significant bit in the PAR is a read only bit as it governs the AIPI registers themselves. It is set to indicate supervisor access only. Bits 31 through 16 in both registers are preset to 1 and the fields are reserved and can only be read.

		Addr
PAR_1	AIPI1 Peripheral Access Register	**0x00200008**
PAR_2	AIPI2 Peripheral Access Register	**0x00210008**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE																
PAR_1 RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
								0xFFFF								
PAR_2 RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
								0xFFFF								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ACCESS															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r
PAR_1 RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
								0xFFFF								
PAR_2 RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
								0xFFFF								

Table 7-10. Peripheral Access Register Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 1.	
ACCESS Bits 15–1	**Access Control**—Each bit controls the access mode of the corresponding peripheral.	0 = Assigned peripheral determines access mode. 1 = the corresponding peripheral is a supervisor access only peripheral.
Reserved Bit 0	Reserved—This bit is reserved and should read 1.	

7.2.3 Peripheral Control Register

These registers are to tell the AIPI whether or not the IP bus peripheral corresponding to the bit location in the register can be accessed in their natural size only. When set to 1, only byte access is allowed on an 8-bit peripheral, only halfword access is allowed on a 16-bit peripheral, and only word access is allowed on a 32-bit peripheral. When set to 1, any access other than natural size that is attempted on the peripheral results in an error response and no IP bus activity occurs.

The least significant bit in the PCR is a read-only bit and the AIPI registers are not governed by this bit. Bits 31 through 16 in both registers are preset to 0 and the fields are reserved and can only be read.

		Addr
PCR_1	AIPI1 Peripheral Control Register	**0x0020000C**
PCR_2	AIPI2 Peripheral Control Register	**0x0021000C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE																
PCR_1 RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								
PCR_2 RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ACCESS_MODE															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r
PCR_1 RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								
PCR_2 RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 7-11. Peripheral Control Register Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read zero.	
ACCESS_MODE Bits 15–1	**Module Access Mode—**Each bit controls the method of access used by the corresponding peripheral assigned to the Module_en.	0 = sub-word and word access is allowed on the peripheral 1 = corresponding peripheral can only be accessed in the natural size. i.e. byte accesses on 8-bit peripherals, half-word accesses on 16-bit peripherals and word accesses on 32-bit peripherals.
Reserved Bit 0	Reserved—This bit is reserved and should read 1.	

7.2.4 Time-Out Status Register

These registers contain status of the AIPI module prior to the occurrence of the time-out event. The Time-Out registers are read-only and status is updated due to time-out operation and module_en must be active. The register is clear during initial reset.

		Addr
TSR_1	AIPI1 Time-Out Status Register	**0x00200010**
TSR_2	AIPI2 Time-Out Status Register	**0x00210010**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TO	RW	ADDR										BE4	BE3	BE2	BE1
TYPE	rw	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
TSR_1 RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															
TSR_2 RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MODULE_EN															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
TSR_1 RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															
TSR_2 RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 7-12. Time-Out Status Register Description

Name	Description	Settings
TO Bit 31	**Time-Out**—This bit when set to 1 indicates a time-out event and may be cleared by the user.	0 = No time-out event 1 = time-out event
RW Bit 30	This bit contains the ips_rwb status prior to time-out event.	–
ADDR Bits 29–20	**Address**—These bits contains the ips_addr[11:2] status prior to time-out event.	–
BE4 Bit 19	This bit contains the ips_byte_31_24 status prior to time-out event.	–
BE3 Bit 18	This bit contains the ips_byte_23_16 status prior to time-out event.	–
BE2 Bit 17	This bit contains the ips_byte_15_8 status prior to time-out event.	–
BE1 Bit 16	This bit contains the ips_byte_7_0 status prior to time-out event.	–
MODULE_EN Bits 15–1	**Module Enable Status**—These bits contains the module_en[15:1] status prior to time-out event. Refer to Table 7-6 to determine which peripheral is assigned to which module_en number.	0= Corresponding module has not timed out 1 = Corresponding module has timed out
Reserved Bit 0	Reserved—This bit is reserved and should read 0.	

7.3 Programming Example

This section covers programming examples written in assembly code to illustrate the data access through the AIPI module.

7.3.1 Data Access to 8-Bit Peripherals

The followings codes are executed with the ARM920T core set to big and little endian modes:

```
LDR    r0,    =0x11223344
LDR    r1,    =0x55667788
LDR    r2,    =8BIT_PERIPHERAL_ADDRESS
STRB   r0,    [r2, #0x0]
STRB   r1,    [r2, #0x1]
STRH   r0,    [r2, #0x2]
STR    r1,    [r2, #0x4]
LDRB   r3,    [r2, #0x0]
LDRB   r4,    [r2, #0x1]
LDRH   r5,    [r2, #0x2]
LDR    r6,    [r2, #0x4]
```

The Table 7-13 on page 7-16 illustrates the difference in the 8-bit peripheral register content.

Table 7-13. Core and 8-Bit Peripheral Register Content After Code Execution

Address	Peripheral Registers			
0	44	–		
1	88	–		
2	44	–		
3	33	–		
4	88	–		
5	77	–		
6	66	–		
7	55	–		
Address	**Core Registers**			
r3	00	00	00	44
r4	00	00	00	88
r5	00	00	33	44
r6	55	66	77	88

7.3.2 Data Access to 16-Bit Peripherals

The followings codes are executed with the ARM core set to big and little endian modes.

```
LDR    r0,    =0x11223344
LDR    r1,    =0x55667788
LDR    r2,    =16BIT_PERIPHERAL_ADDRESS
STRB   r0,    [r2, #0x0]
STRB   r1,    [r2, #0x1]
STRH   r0,    [r2, #0x2]
STR    r1,    [r2, #0x4]
LDRB   r3,    [r2, #0x0]
LDRB   r4,    [r2, #0x1]
LDRH   r5,    [r2, #0x2]
LDR    r6,    [r2, #0x4]
```

The Table 7-14 and Table 7-15 illustrate the difference in the 16-bit peripheral register content.

Table 7-14. Core and 16-Bit Peripheral Register Content (Little Endian)

Address	Peripheral Registers		
0	88	44	–
2	33	44	–
4	77	88	–

Table 7-14. Core and 16-Bit Peripheral Register Content (Little Endian) (continued)

Address	Peripheral Registers			
6	55	66	–	
Address	**Core Registers**			
r3	00	00	00	44
r4	00	00	00	88
r5	00	00	33	44
r6	55	66	77	88

Table 7-15. Core and 16-Bit Peripheral Register Content (Big Endian)

Address	Peripheral Registers			
0	44	88	–	
2	33	44	–	
4	55	66	–	
6	77	88	–	
Address	**Core Registers**			
r3	00	00	00	44
r4	00	00	00	88
r5	00	00	33	44
r6	55	66	77	88

7.3.3 Data Access to 32-Bit Peripherals

The followings codes are executed with the ARM core set to big and little endian modes.

```
LDR    r0,    =0x11223344
LDR    r1,    =0x55667788
LDR    r2,    =32BIT_PERIPHERAL_ADDRESS
STRB   r0,    [r2, #0x0]
STRB   r1,    [r2, #0x1]
STRH   r0,           [r2, #0x2]
STR    r1,    [r2, #0x4]
LDRB   r3,    [r2, #0x0]
LDRB   r4,    [r2, #0x1]
LDRH   r5,    [r2, #0x2]
LDR    r6,    [r2, #0x4]
```

The Table 7-16 and Table 7-17 on page 7-18 illustrate the difference in the 32-bit peripheral register content.

Table 7-16. Core and 32-bit Peripheral Register Content (Little Endian)

Address	Peripheral Registers			
0	33	44	88	44
4	55	66	77	88
Address	**Core Registers**			
r3	00	00	00	44
r4	00	00	00	88
r5	00	00	33	44
r6	55	66	77	88

Table 7-17. Core and 32-bit Peripheral Register Content (Big Endian)

Address	Peripheral Registers			
0	44	88	33	44
4	55	66	77	88
Address	**Core Registers**			
r3	00	00	00	44
r4	00	00	00	88
r5	00	00	33	44
r6	55	66	77	88

7.3.4 Special Consideration for Non-Natural Size Access

A programmer must exercise care when accessing peripherals using access other than their natural size. An example of such access includes byte access to a 32-bits peripheral and word access to 8-bit peripheral. The examples in the previous section clearly illustrate the difference in byte accessing a 32-bits peripheral in both big and little endian modes. An instruction such as:

```
STRB        r1,    [r2, #0x1]
```

is accessing using byte lane[15:8] in little endian, while byte lane[23:16] is accessed using big endian mode. Therefore, if a programmer is using byte access to set up control information in 32-bit register, extreme care must be taken to ensure the desired byte is written during the desired endian mode.

Chapter 8
System Control

This chapter describes the system control module of the MC9328MXL microprocessor. The system control module enables system software to control, customize, or read the status of the following functions:

- Multiplexing of SSI signals
- Multiplexing of the SDRAM/SyncFlash chip select signal
- Chip ID
- System boot mode selection

8.1 Programming Model

The system control module includes four user-accessible 32-bit registers. Table 8-1 summarizes these registers and their addresses.

Table 8-1. System Control Module Register Memory Map

Description	Name	Address
Silicon ID Register	SIDR	0x0021B804
Function Multiplexing Control Register	FMCR	0x0021B808
Global Peripheral Control Register	GPCR	0x0021B80C
Global Clock Control Register	GCCR	0x0021B810

8.1.1 Silicon ID Register

This 32-bit read-only register shows the chip identification. The bit assignments for the register are shown in the following register display. The settings for the bits in the register are listed in Table 8-2.

SIDR Silicon ID Register **Addr 0x0021B804**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SID															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
	0x04D4															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SID															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	0	0	1	0	0	0	0	0	0	0	1	1	1	0	1
	0xC01D															

Table 8-2. Silicon ID Register Description

Name	Description	Settings
SID Bits 31–0	**Silicon ID**—Contains the chip identification number of the MC9328MXL.	Silicon ID: Mask 1L = 0x04D4 C01D Mask 2L = 0x00D4 C01D

8.1.2 Function Multiplexing Control Register

The Function Multiplexing Control Register (FMCR) controls the multiplexing of the signal lines shared by the SSI module. It also controls the SDRAM/SyncFlash chip select lines and masking of the external bus request. See Table 8-3 for detailed description of bit settings.

FMCR Function Multiplexing Control Register **Addr 0x0021B808**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
									0x0000							

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								SPI2_RXD_SEL	SSI_RXFS_SEL	SSI_RXCLK_SEL	SSI_RXDAT_SEL	SSI_TXFS_SEL	SSI_TXCLK_SEL	EXT_BR_EN	SDCS1_SEL	SDCS0_SEL
TYPE	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
									0x0003							

Table 8-3. Function Multiplexing Control Register Description

Name	Description	Settings
Reserved Bits 31–9	Reserved—These bits are reserved and should read 0.	
SPI2_RXD_SEL Bit 8	**SPI2 Receive Data Input Select**—Selects the SPI2 Receive Data input source.	0 = input from SPI2_RXD_0 pin (AOUT of GPIO port A[1]) 1 = input from SPI2_RXD_1 pin (AOUT of Port D[9])
SSI_RXFS_SEL Bit 7	**SSI Receive Frame Sync Input Select**—Selects the Receive Frame Sync input source. See Figure 27-1 on page 27-2.	0 = input from Port C 1 = input from Port B
SSI_RXCLK_SEL Bit 6	**SSI Receive Clock Select**—Selects the Receive Clock input source. See Figure 27-1 on page 27-2.	0 = input from Port C 1 = input from Port B
SSI_RXDAT_SEL Bit 5	**SSI Receive Data Select**—Selects the Receive Data input source. See Figure 27-1 on page 27-2.	0 = input from Port C 1 = input from Port B
SSI_TXFS_SEL Bit 4	**SSI Transmit Frame Sync Select** —Selects the Transmit Frame Sync input source. See Figure 27-1 on page 27-2.	0 = input from Port C 1 = input from Port B
SSI_TXCLK_SEL Bit 3	**SSI Transmit Clock Select**—Selects the Transmit Clock input source. See Figure 27-1 on page 27-2.	0 = input from Port C 1 = input from Port B
EXT_BR_EN Bit 2	**External Bus Request Control**—Chooses whether the external bus request function is masked or enabled. The external bus request is a test signal and the EXT_BR_EN bit must be clear during normal operation.	0 = External bus request masked 1 = External bus request enabled

Table 8-3. Function Multiplexing Control Register Description (continued)

Name	Description	Settings
SDCS1_SEL Bit 1	**SDRAM/SyncFlash Chip-Select**—Selects the function of the $\overline{\text{CS3}}$/$\overline{\text{CSD1}}$ pin.	0 = $\overline{\text{CS3}}$ selected 1 = $\overline{\text{CSD1}}$ selected
SDCS0_SEL Bit 0	**SDRAM/SyncFlash Chip-Select**—Selects the function of the $\overline{\text{CS2}}$/$\overline{\text{CSD0}}$ pin.	0 = $\overline{\text{CS2}}$ selected 1 = $\overline{\text{CSD0}}$ selected

8.1.3 Global Peripheral Control Register

The Global Peripheral Control Register (GPCR) controls the driving force parameters of the bus and several other functions in the MC9328MXL. Descriptions of the register settings appear in Table 8-6.

GPCR Global Peripheral Control Register **Addr 0x0021B80C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MEMC_TEST_EN				DS_SLOW		DS_CNTL		DS_ADDR		DS_DATA		IP_CLK_GATING_EN	HCLK_GATING_EN	MMA_PROT_EN	CSI_PROT_EN
TYPE	rw	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw
RESET	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1
	0x03FC															

Table 8-4. Global Peripheral Control Register Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
MEMC_TEST_EN Bits15	Active high of this bit switches the inner test signals from MEMC to GPIO for debug purposes.	0 = Normal 1 = Switch MEMC inner test signals to GPIO for debug purpose.

Table 8-4. Global Peripheral Control Register Description (continued)

Name	Description	Settings
Reserved Bits 14–12	Reserved—These bits are reserved and should read 0.	
DS_SLOW Bits 11–10	**Driving Strength Slow I/O**—Controls the driving strength of all slow I/O signals.	00 = 26 MHz/15 pF 01 = 26 MHz/30 pF 10 = 26 MHz/45 pF 11 = 26 MHz/greater than 45 pF
DS_CNTL Bits 9–8	**Driving Strength Bus Control Signals**—Controls the driving strength of bus control signals.	00 = 50 MHz/15 pF 01 = 50 MHz/30 pF 10 = 100 MHz/15 pF 11 = 100 MHz/30 pF
DS_ADDR Bits 7–6	**Driving Strength Address Bus**—Controls the driving strength of the address bus.	00 = 50 MHz/15 pF 01 = 50 MHz/30 pF 10 = 100 MHz/15 pF 11 = 100 MHz/30 pF
DS_DATA Bits 5–4	**Driving Strength Data Bus**—Controls the driving strength of the data bus.	00 = 50 MHz/15 pF 01 = 50 MHz/30 pF 10 = 100 MHz/15 pF 11 = 100 MHz/30 pF
IP_CLK_GATING_EN Bit 3	**IP Clock Gating Enabled**—Controls ips_gated_clk signal.	0 = ips_gated_clk is continuous clock 1 = ips_gated_clk will be gated by aipi1_gated_clk_en or aipi2_gated_clk_en
HCLK_GATING_EN Bit 2	**HCLK_GATING_Enabled**—This bit only applies to functional and register access clocks; DMA, CSI, MMA and AITC. The functional clock and register access clock can be either a gated clock or continuous clock. Low value of this bit can disable clock gating for DMA/ AITC/ CSI/ MMA and make their clock become a continuous clock. This bit is write only and when read, will always be 0. **Note:** Clocks of DMA/CSI/DSPA can also be turned off by corresponding bits in GCCR. When their clocks are turned off in GCCR, this bit has no effect on these modules because their clocks are off. Please refer GCCR register description for detail information.	1 = hclk gating enabled, some hclk to DMA/AITC/CSI/MMA is gated clock 0 = hclk gating disabled, all hclk becomes continuous clock
MMA_PROT_EN Bit 1	**MMA Privileged Mode Access**—Selects whether the MMA can only be accessed in privileged mode or if it can be accessed in all modes.	0 = All access modes available 1 = Privileged mode access only
CSI_PROT_EN Bit 0	**CMOS Sensor Interface Privileged Mode Access**—Selects whether the CSI can only be accessed in privileged mode or if it can be accessed in all modes.	0 = All access modes available 1 = Privileged mode access only

8.1.4 Global Clock Control Register

The Global Clock Control Register (GCCR) provides additional power saving capabilities by controlling the clocks in the following MC9328MXL modules: DMA, CSI, MMA and USB. It also controls the clock source for Bootstrap mode.

GCCR Global Clock Control Register **Addr 0x0021B810**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
										0x0000						

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												BROM_CLK_EN	DMA_CLK_EN	CSI_CLK_EN	MMA_CLK_EN	USBD_CLK_EN
TYPE	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
										0x000F						

Table 8-5. Global Clock Control Register Description

Name	Description	Settings
Reserved Bits 31–5	Reserved—These bits are reserved and should read 0.	
BROM_CLK_EN Bit 4	**BROM Clock Enable**—Only available in Bootstrap mode. This bit enables/disables the operational system boot mode of the MC9328MXL upon system reset. The boot mode is determined by the settings of these pins.	0 = Clock gating is controlled by setting of BOOT[3:0] pins. 1 = Overrides the setting of the BOOT[3:0] pins and forces the HCLK to be used as clock.
DMA_CLK_EN Bit 3	**DMA Clock Enable**—Enables/Disables clock input to the DMA module.	0 = DMA clock input is disabled. 1 = DMA clock input is enabled (default).
CSI_CLK_EN Bit 2	**CMOS Sensor Interface Clock Enable**— Enables/Disables clock input to the CSI module.	0 = CSI clock input is disabled. 1 = CSI clock input is enabled (default).
MMA_CLK_EN Bit 1	**Multimedia Accelerator Clock Enable**— Enables/Disables clock input to the MMA module.	0 = CSI clock input is disabled. 1 = CSI clock input is enabled (default).
USBD_CLK_EN Bit 0	**USBD Clock Enable**—Enables/Disables clock input to the USB module.	0 = USB clock input is disabled. 1 = USB clock input is enabled (default).

8.2 System Boot Mode Selection

The operational system boot mode of the MC9328MXL upon system reset is determined by the configuration of the four external input pins BOOT[3:0]. The settings of these pins control the following functions:

- CS0 boot function of the EIM module
- Control of the SyncFlash chip select (CSD1) boot function of the SDRAM controller

The settings of the system control module for the system boot mode selection are displayed in Table 8-6.

The MC9328MXL always begins fetching instructions from address 0x00000000 after reset. The BOOT[3:0] pins control the memory region that is mapped to address 0x0. The boot modes are defined in Table 8-6. The BOOT[3:0] pins also control the initial configuration (for example bus width) for the external memory regions. When an external chip select is enabled by the BOOT[3:0] pins, the first 1 Mbyte range (0x0000000–0x000FFFFF) of the chip select's memory space is also mapped to address 0x0.

For example, by setting BOOT[3:0] to 0110, the MC9328MXL will boot from the CS0 memory region using a 32-bit data bus width. The first 1 Mbyte of the CS0 memory space (0x10000000–0x100FFFFF) will be mapped to addresses 0x00000000–0x000FFFFF.

NOTE:

The BOOT pins must not change once the MC9328MXL is out of reset. To achieve logic 0, a BOOT input must be tied to GND through a 1 kohm resistor. Otherwise, excessive current may occur at power up. BOOT[3] must always be terminated with a 1 kohm resistor to GND.

NOTE:

If Bootstrap ROM is not selected for the boot mode, the internal ROM is not accessible.

Table 8-6. System Boot Mode Selection

Inputs BOOT[3:0]	Output Signals Active Device
0000	Bootstrap ROM
0001	16-bit SyncFlash D[15:0]
0010	32-bit SyncFlash
0011	8-bit CS0 at D[7:0]
0100	16-bit CS0 at D[31:16]
0101	16-bit CS0 at D[15:0]
0110	32-bit CS0 at D[31:0]
0111	Reserved

Chapter 9
Bootstrap Mode Operation

Bootstrap mode is designed to allow you to initialize a target system and download programs or data to the target system's RAM using the UART 1 or UART 2 controller. After a program is downloaded, it can be executed, which gives you a simple debugging environment for failure analysis and a channel to update programs stored in flash memory. Bootstrap mode has the following capabilities:

- Allows you to initialize your system and download programs and data to system memory using UART 1 or UART 2.
- Accepts execution commands to run programs stored in system memory
- Supports memory and register read and write operations of selectable data size (byte, half-word, or word)
- Provides an 8-word instruction buffer for ARM920T vector table storage, instruction storage and execution

9.1 Operation

In bootstrap mode, only MC9328MXL's UART 1 and UART 2 controllers are initialized. They are configured to auto-baud detection mode, ignore RTS, keep CTS always active, no parity, 8-bit character length, and one stop bit. Then they are ready to accept bootstrap data download. The first character received must be a or A. This character determines the baud rate to be used and which UART port is being used for bootstrapping. The first character is not part of a program or data being downloaded. To download the data or program, the code must be converted to a bootstrap format file, which is a text file that contains bootstrap records. A DOS-executable program, STOB.EXE, can be downloaded from the i.MX Web site to convert an S-record file to a bootstrap format file.

The MC9328MXL's internal registers must be initialized as the target system before a program can be downloaded to system memory. Because these internal registers can be treated as a type of memory, each of them can be initialized by issuing a bootstrap record.

The bootstrap design provides an 8-word instruction buffer to which ARM920T core instructions can be downloaded. The buffers are word-access only. This feature enables the ARM920T core instructions to be run even if the memory systems are disabled or in a core stand-alone system. The instruction buffer starts at 0x00000004. Regardless of the operation (initializing internal registers, downloading a program to system RAM, or issuing a core instruction), bootstrap mode will only accept bootstrap record transfers that are made with the UART. The record type determines what action will occur.

The instruction buffer allows users to download the vector table onto the buffer without the use of external ROM or Flash, the feature provides a fast and easy environment to users when using IRQ during program debug.

9.1.1 Entering Bootstrap Mode

Bootstrap mode is the debug mode of the MC9328MXL. To enter bootstrap mode, the BOOT pins must be properly configured during system reset. After reset, the bootstrap ROM is selected for reset vector fetch cycles.

9.1.2 Bootstrap Record Format

Only bootstrap records (b-records) are accepted for data transfers in bootstrap mode. The b-record format is shown in Table 9-1, and Table 9-2 further defines the COUNT/MODE byte. All b-records are in uppercase. Each byte is represented by two ASCII characters during transfer (for example, a byte of value 0x12 will be represented by the characters 12).

Table 9-1. Bootstrap Record Format

4 Bytes	1 Byte	N (Count) Bytes
Address	Count/mode	Data

Table 9-2. Definition of COUNT/MODE Byte

Bit(s)	Definition	Settings
7–6	Data size	00 = Byte 01 = Half-word 10 = Reserved 11 = Word
5	Read/write flag	0 = Write 1 = Read
4–0	Data count in number of bytes	Value from 0 to 31

NOTE:

1. A half-word is defined as 16 bits, while a word is defined as 32 bits.
2. The address specified must fall on a data size boundary: for word access, the last 2 bits of the address must be 0, while for half-word access, the last bit of the address must be 0. The data count in the COUNT/MODE byte must be a multiple of the data size: for word access, the data count must be a multiple of four, while for half-word access, the data count must be in multiple of two. If either the address or the data count is not on an appropriate data size boundary, the bootloader program will return a * character (asterisk) to indicate that an error has occurred, and the bootloader will then start waiting for a new b-record.
3. During a read operation, a / character (forward slash) is returned after the last data has been returned.
4. A data count of zero (disregard the value of data size and the status of the mode flag) has a special meaning: execute from the address specified. In this case, no data will follow the COUNT/MODE byte.

Comments can be added to files of b-records. As described above, the shortest b-record consists of 10 ASCII characters (when the data count is 0) of 0 to 9 or A to F (hexadecimal digits). Comments included must not contain patterns to prevent the comments from being considered a b-record.

9.1.3 Registers Used in Bootloader Program

The bootloader program uses general-purpose registers r5 to r12 as well as r13 as the return register and r14 as the link register. All the other registers can be used by target programs.

9.1.4 Setting Up the RS-232 Terminal

To set up communication between your target system and the PC, set the communication specifications to the baud rate desired, no parity, 8-bit character length, and 1 stop bit. You may pause after each line (b-record) is transferred to make sure each transferred ASCII character is echoed.

After setting up the hardware, powering up the system, and entering bootstrap mode, send an a or A character to the target system to initiate the link. Once the bootloader receives this character, it adjusts the baud rate. If the link is successful, the bootloader will return the special character : (colon) as an acknowledgment.

9.1.5 Changing the Speed of Communication

You can change the communication baud rate after communication is set up in the RS-232 terminal. Simply issue a b-record to re-initialize the baud control register of the UART controller. After the last character of this b-record is sent, the echo of this last character will be transmitted at the new speed. The maximum speed recommended for Bootstrap is 57600 baud.

9.2 B-Record Example

Before you can download a program to system memory, the target system may need to be initialized using the internal registers. An init file can be built using a text editor. Code Example 9-1 initializes the external SRAM memory location 0x13010000 to 0x12345678 in word access mode, the location 0x13010006 to 0x7788 in half-word access mode, and the location 0x13010009 to 0x55 in byte access mode.

Code Example 9-1. init.b Example

```
// init.b -- Initialization Example
13010000C412345678        initialize 0x13010000 to 0x12345678
13010006427788            initialize 0x13010006 to 0x7788
130100090155              initialize 0x13010009 to 0x55
```

With b-records similar to those stated above, a target program can be downloaded to memory and executed from the address chosen with the following b-record:

```
1122334400      execute from 11223344
```

The target program may exit and return to the bootloader program by jumping to address 0x00000100, where the bootloader program starts.

9.3 Instruction Buffer Usage

A 8-word instruction buffer is provided for ARM920T core vector table storage, instruction and data storage. The buffer starts at 0x00000004. Up to eight instructions can be loaded to the instruction buffer for execution.Usually, the last instruction is an unconditional jump instruction (jmp) that jumps to the start of the bootloader program (0x00000100).

Code Example 9-2 fills memory locations starting from 0x00310000 to 0x130100FF (the length of 0x100) with 0x12345678.

Code Example 9-2. Instruction Buffer Sample

```
        ldr     r1,=0x13010000          // starting address is 0x130100FF
        ldr     r2,=0x100               // length is 0x100
        ldr     r3,=0x12345678          // data to fill is 0x12345678
        ldr     r4,=0x00000100          // bootloader program
loop:
        str     r3,[r1, r2]             // store data
        subs    r2,r2, #4               // decrement address
        bne     loop                    // loop back till r2 down to 0x0
        mov     pc, r4                  // return to bootloader program
```

Because the instruction buffer is of limited size, the programmer cannot do everything at the same time. The program can be broken into five parts, as shown in Table 9-3.

Table 9-3. Program Breakdown

Part	Code
1	`ldr r4,=0x00000100 // bootloader address 0x00000100` `mov pc, r4 // return to bootloader program`
2	`ldr r1,=0x13010000 // starting address is 0x13010000` `mov pc, r4 // return to bootloader program`
3	`ldr r2,=0x00000100 // length is 0x100` `mov pc, r4 // return to bootloader program`
4	`ldr r3,=0x12345678 // data to fill is 0x12345678` `mov pc, r4 // return to bootloader program`

5	`loop`
	`str r3,[r1, r2] // store data`
	`subs r2,r2, #4 // decrement address`
	`bne loop // loop back till r2 down to 0x0`
	`mov pc, r4 // return to bootloader program`

Breaking down the register initialization into three parts is not mandatory, however it produces similar b-records and therefore is easier to manage.

The resulting b-records appear in Table 9-4.

Table 9-4. Resulting B-Records

B-Record Number	B-Record
1	00000004 08E3A04F40E1A0F004 0000000400
2	00000004 08E3A019C4E1A0F004 0000000400
3	00000004 08E3A02F40E1A0F004 0000000400
4	00000004 0CE59F3000E1A0F00412345678 0000000400
5	00000004 0FE7813002E25220041AFFFFFCE1A0F004 0000000400

Note that all b-records start at the same address, 0x00000004, which is the starting address of the instruction buffer. B-records 1, 2, and 3 are very similar and can be used as prototype b-records for general-purpose register initialization.

Therefore, the resulting b-record file will be as follows:

Code Example 9-3. Bootloader B-Record

```
00000004 08E3A04F40E1A0F004initialize r4 to 0x00000100 (bootloader start)
0000000400   execute and return to bootloader
00000004 08E3A019C4E1A0F004initialize r1 to 0x13010000 (start)
0000000400   execute and return to bootloader
00000004 08E3A02F40E1A0F004initialize r2 to 0x100 (length)
0000000400   execute and return to bootloader
00000004 0CE59F3000E1A0F00412345678initialize r3 to 0x12345678 (content)
0000000400   execute and return to bootloader
00000004 0FE7813002E25220041AFFFFFCE1A0F004memory fill
0000000400   execute and return to bootloader
```

9.4 Simple Read/Write Examples

Table 9-5 provides examples demonstrating how to perform memory and register reads/writes of various data sizes. Code Example 9-4 shows an example of the code used for Vector Tables

Code Example 9-4.

```
NOP                     ; 0x00
                NOP                     ; 0x04 (programmable buffer)
                NOP                     ; 0x08 (programmable buffer)
IRQ_Addr        DCD     C_IRQ_Handler   ; 0x0C (programmable buffer)
FIQ_Addr        DCD     C_FIQ_Handler   ; 0x10 (programmable buffer)
                NOP                     ; 0x14 (programmable buffer)
                LDR     PC, IRQ_Addr    ; 0x18 (programmable buffer)
                LDR     PC, FIQ_Addr    ; 0x1C (programmable buffer)
                DCD     0               ; 0x20 (programmable buffer)
```

Table 9-5. Read/Write Examples

Example Type	B-Record	Return Value
Read 3 bytes starting from location 0x00310000	0031000023	0031000003XXYYZZ/ (where XX, YY, and ZZ are data in byte)
Read 3 half-words starting from location 0x00310000	0031000066 (6 bytes = 3 half-words)	0031000066XXXXYYYYZZZZ/ (where XXXX, YYYY, and ZZZZ are data in half-word)
Read 3 words starting from location 0x00310000	00310000EC (12 bytes = 3 words)	00310000ECXXXXXXXXYYYYYYYYZZZZZZZZ/ (where XXXXXXXX, YYYYYYYY, and ZZZZZZZZ are data in word)
Write 3 bytes starting from location 0x00310000	0031000003112233	0031000003112233/
Write 3 half-words starting from location 0x00310000	00310000461111222233 33 (6 bytes = 3 half-words)	0031000046111122223333/

Table 9-5. Read/Write Examples (continued)

Example Type	B-Record	Return Value
Write 3 words starting from location 0x00310000	00310000CC1111111122222222 33333333 (12 bytes = 3 words)	00310000CC1111111122222222233333333/

9.5 Bootloader Flowchart

Figure 9-1 on page 9-7 illustrates how the bootloader program operates inside the MC9328MXL. The bootloader starts when the MC9328MXL enters bootstrap mode.

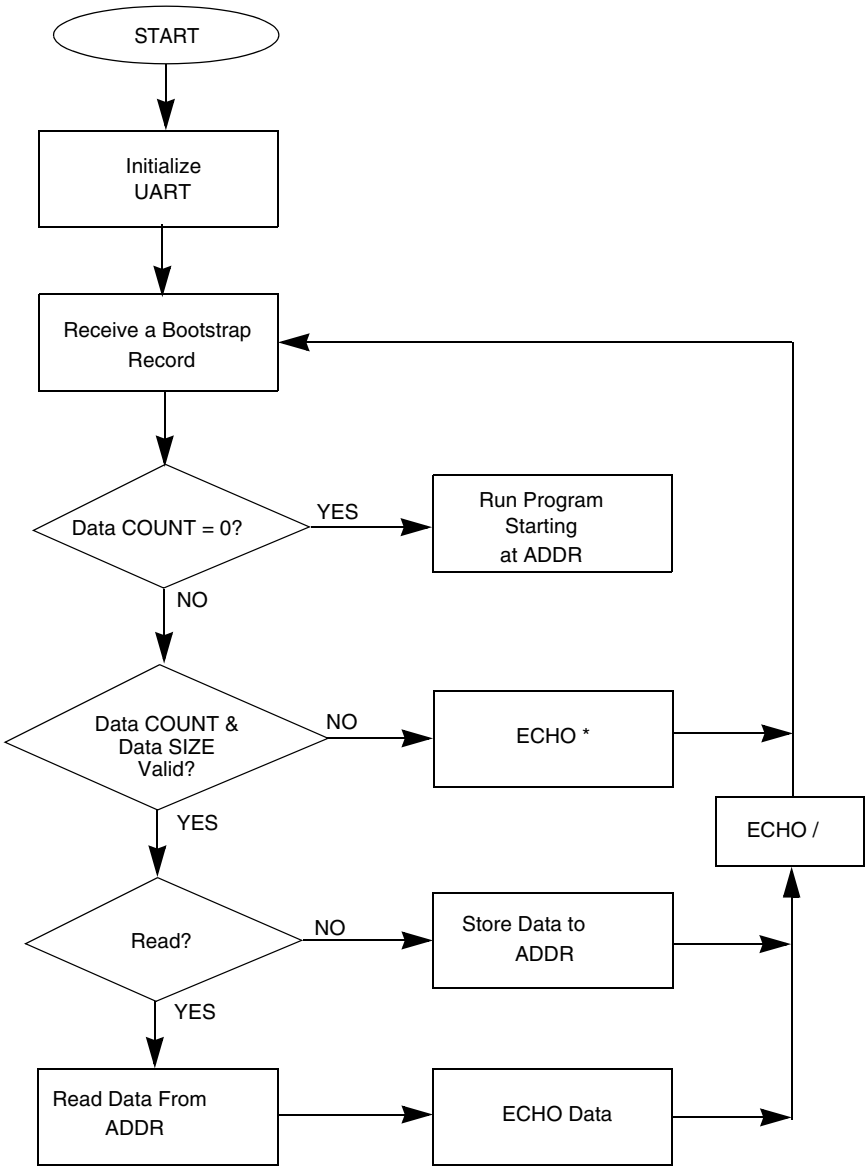


Figure 9-1. Bootloader Program Operation

9.6 Special Notes

The following summary items may be helpful when working in bootstrap mode.

- A b-record is a string of uppercase hex characters with optional comments that follow.

- Comments in a b-record or b-record file must not contain any word or symbol that is longer than nine characters. However, the following characters can be used in a string of any length (all of these have an ASCII code value that is less than 0x30):
  - space
  - ! (exclamation point)
  - “ (quotation mark)
  - # (number sign)
  - $ (dollar sign)
  - % (percentage symbol)
  - & (ampersand)
  - ((opening parenthesis)
  -) (closing parenthesis)
  - * (asterisk)
  - + (plus sign)
  - - (minus sign)
  - . (period)
  - / (forward slash)
  - , (comma)
- The bootloader program echoes all characters being received, however only those having an ASCII code value greater than or equal to 0x30 are kept for b-record assembling. Sending a character that is not a b-record (ASCII code value less than 0x30) will force the bootloader to start a new b-record.
- General-purpose registers r5–r14 and supervisor scratch register are used by the bootloader program. Writing to these registers may corrupt the bootloader program.
- Please visit the DragonBall Web site for bootstrap utility programs.

Chapter 10
Interrupt Controller (AITC)

This chapter describes the ARM9 Interrupt Controller (AITC) that is used to control and prioritize up to 64 interrupts in the MC9328MXL. This chapter describes the registers and bit settings plus all other information necessary to write the software necessary to write interrupt service routines.

10.1 Introduction

The MC9328MXL interrupt controller (AITC) is a 32-bit peripheral that collects interrupt requests from a maximum of 64 sources and provides an interface to the ARM920T processor.

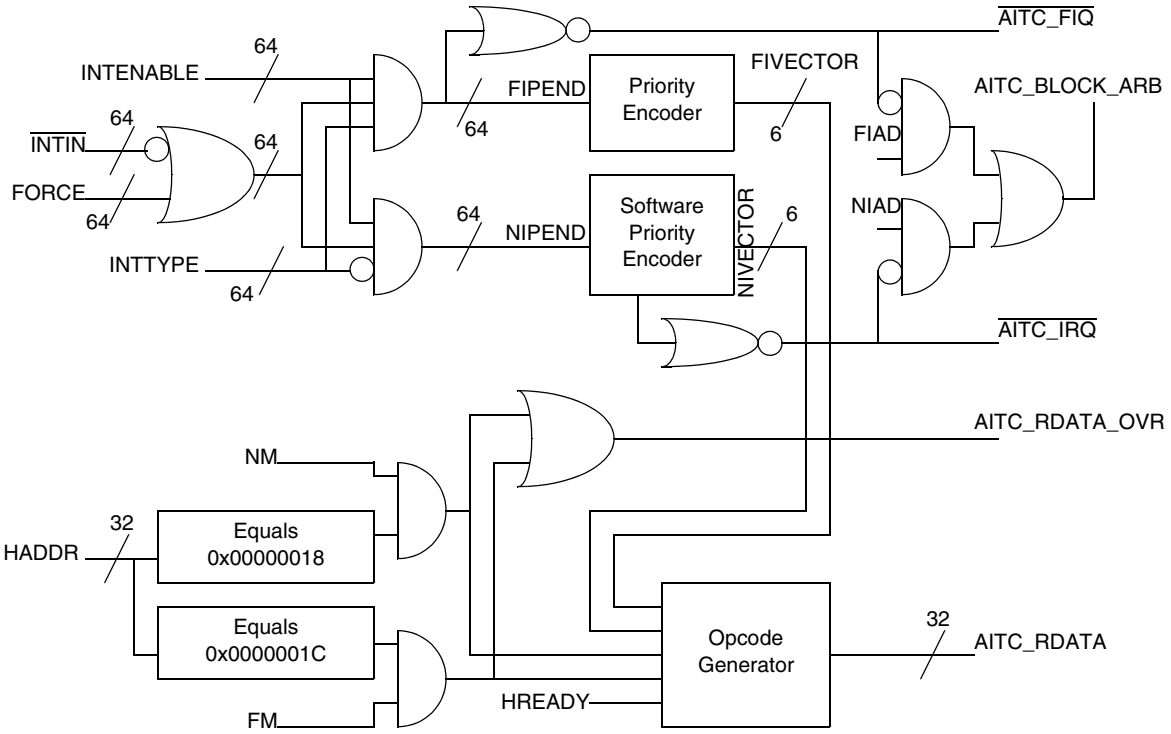


Figure 10-1. AITC Block Diagram

The AITC performs the following functions:

- Supports a maximum of 64 interrupt sources
- Supports fast and normal interrupts
- Selects normal or fast interrupt request for any interrupt source
- Indicates pending interrupt sources via a register for normal and fast interrupts

- Detects all pending interrupts and distinguishes by priority level
- Independently enables or disables any interrupt source
- Provides a mechanism for software to schedule an interrupt
- Supports a maximum of 16 software controlled priority levels for normal interrupts and priority masking

10.2 Operation

The interrupt controller consists of a set of control registers and associated logic to perform interrupt masking, priority support, and hardware acceleration of normal interrupts.

The interrupt source registers (INTSRCH and INTFRCL) are a pair of 32-bit status registers with a single interrupt source associated with each of the 64 bits. An interrupt line or set of interrupt lines is routed from each interrupt source to the INTSRCH or INTFRCL register. This configuration allows the ARM920T processor of the MC9328MXL to monitor a maximum of 64 distinct interrupt sources.

Interrupt requests can be forcibly asserted through the interrupt force registers (INTFRCH and INTFRCL). Each bit in these registers is logically ORed with the corresponding hardware request line prior to input to the INTSRCH or INTFRCL registers.

There is a corresponding set of interrupt enable registers (INTENABLEH and INTENABLEL), each 32 bits wide, that allow individual bit masking of the INTSRCH and INTFRCL registers. There is also a corresponding set of interrupt type registers (INTTYPEH and INTTYPEL) that selects whether an interrupt source generates a normal or fast interrupt to the ARM920T processor.

There is a corresponding set of normal interrupt pending registers (NIPNDH and NIPNDL) that indicate pending normal interrupt requests, and are equivalent to the logical AND of the interrupt source registers (INTSRCH and INTSRCL), the interrupt enable registers (INTENABLEH and INTENABLEL), and the NOT of the interrupt type registers (INTTYPEH and INTTYPEL). The NIPNDH and NIPNDL register bits are bit-wise NORed together to generate the nIRQ signal that is routed to the ARM920T processor. This ARM920T processor input signal is maskable by the normal interrupt disable bit (I bit) in the program status register (CPSR). The normal interrupt vector register (NIVECSR) indicates the vector index of highest priority pending normal interrupt.

There is a corresponding set of fast interrupt pending registers (FIPNDH and FIPNDL) that indicate pending fast interrupt requests, and are equivalent to the logical AND of the interrupt source registers (INTSRCH and INTSRCL), the interrupt enable registers (INTENABLEH and INTENABLEL), and the interrupt type registers (INTTYPEH and INTTYPEL). The FIPNDH and FIPNDL register bits are bit-wise NORed together to generate the nFIQ signal that is routed to the ARM920T processor. This ARM920T processor input signal is maskable by the fast interrupt disable bit (F bit) in the CPSR. The fast interrupt vector register (FIVECSR) indicates the vector index of highest priority pending fast interrupt.

All interrupt controller registers are readable and writable in supervisor mode only. Writes attempted to read-only registers are ignored. These registers must be written with 32-bit stores only.

The INTFRCH and INTFRCL registers are provided for software generation of interrupts. By enabling interrupts for these bit positions, software can force an interrupt request. This register also provides an alternate method of interrupt assertion for debugging hardware interrupt service routines.

The interrupt requests are prioritized in the following order:

1. Fast interrupt requests, in order of highest number
2. Normal interrupt requests, in order of highest priority level, then highest source number with the same priority

The AITC provides 16 software controlled priority levels for normal interrupts and every interrupt can be placed in any priority level. The AITC also provides a normal interrupt priority level mask (NIMASK) that disables any interrupt with a priority level less than or equal to the mask. When a level 0 normal interrupt and a level 1 normal interrupt are asserted at the same time, the level 1 normal interrupt is selected unless NIMASK has disabled level 1 normal interrupts. When two level 1 normal interrupts are asserted at the same time, the level 1 normal interrupt with the highest source number is selected unless NIMASK has disabled level 1 normal interrupts.

10.3 AITC Interrupt Controller Signals

The active-low $\overline{\text{INTIN}}$ [63:0] input signals indicate that a peripheral device is requesting an interrupt to the interrupt controller. The interrupt controller recognizes an interrupt is asserted on the rising edge of the clock and does not latch and hold the interrupt. The peripheral must keep the interrupt request asserted until the software acknowledges and clears the interrupt request.

The interrupt source assignment of $\overline{\text{INTIN}}$ [63:0] is shown Table 10-1. Interrupt sources in the table that are labeled ‘unused’ may be used by software to force an interrupt request for a specific source using either the INTFRCH or INTRFRCL registers.

In Table 10-1, some signals are shown with overbars to represent the logic inside the chip. However, all asserted interrupts result in the associated bit being a 1 in the Interrupt Source Registers

Table 10-1. Interrupt Assignment

Bit #	Name of Interrupt	Bit #	Name of Interrupt	Bit #	Name of Interrupt	Bit #	Name of Interrupt
0	Unused	16	Unused	32	Unused	48	$\overline{\text{USBD_INT}}$ [1]
1	Unused	17	$\overline{\text{RTC_INT}}$	33	Unused	49	$\overline{\text{USBD_INT}}$ [2]
2	Unused	18	$\overline{\text{RTC_SAM_INT}}$	34	$\overline{\text{PWM_INT}}$	50	$\overline{\text{USBD_INT}}$ [3]
3	Unused	19	$\overline{\text{UART2_MINT_PFERR}}$	35	$\overline{\text{MMC_IRQ}}$	51	$\overline{\text{USBD_INT}}$ [4]
4	Unused	20	$\overline{\text{UART2_MINT_RTS}}$	36	Unused	52	$\overline{\text{USBD_INT}}$ [5]
5	Unused	21	$\overline{\text{UART2_MINT_DTR}}$	37	Unused	53	$\overline{\text{USBD_INT}}$ [6]
6	$\overline{\text{CSI_INT}}$	22	$\overline{\text{UART2_MINT_UARTC}}$	38	Unused	54	Unused
7	$\overline{\text{MMA_MAC_INT}}$	23	$\overline{\text{UART2_MINT_TX}}$	39	$\overline{\text{I2C_INT}}$	55	Unused
8	$\overline{\text{MMA_INT}}$	24	$\overline{\text{UART2_MINT_RX}}$	40	SPI2_INT	56	Unused
9	Unused	25	$\overline{\text{UART1_MINT_PFERR}}$	41	SPI1_INT	57	Unused
10	$\overline{\text{MSIRQ}}$	26	$\overline{\text{UART1_MINT_RTS}}$	42	$\overline{\text{SSI_TX_INT}}$	58	TIMER2_INT
11	GPIO_INT_PORTA	27	$\overline{\text{UART1_MINT_DTR}}$	43	$\overline{\text{SSI_TX_ERR_INT}}$	59	TIMER1_INT
12	GPIO_INT_PORTB	28	$\overline{\text{UART1_MINT_UARTC}}$	44	$\overline{\text{SSI_RX_INT}}$	60	$\overline{\text{DMA_ERR}}$
13	GPIO_INT_PORTC	29	$\overline{\text{UART1_MINT_TX}}$	45	$\overline{\text{SSI_RX_ERR_INT}}$	61	$\overline{\text{DMA_INT}}$
14	$\overline{\text{LCDC_INT}}$	30	$\overline{\text{UART1_MINT_RX}}$	46	Unused	62	$\overline{\text{GPIO_INT_PORTD}}$
15	Unused	31	Unused	47	$\overline{\text{USBD_INT}}$ [0]	63	$\overline{\text{WDT_INT}}$

10.4 Programming Model

The AITC module includes 26 user-accessible 32-bit registers. All of these registers are single cycle access because the AITC sits on the native bus of the ARM920T processor. Table 10-2 summarizes these registers and their addresses. Table 10-3 provides an overview of the register fields.

Table 10-2. AITC Module Register Memory Map

Description	Name	Address
Interrupt Control Register	INTCNTL	0x00223000
Normal Interrupt Mask Register	NIMASK	0x00223004
Interrupt Enable Number Register	INTENNUM	0x00223008
Interrupt Disable Number Register	INTDISNUM	0x0022300C
Interrupt Enable Register High	INTENABLEH	0x00223010
Interrupt Enable Register Low	INTENABLEL	0x00223014
Interrupt Type Register High	INTTYPEH	0x00223018
Interrupt Type Register Low	INTTYPEL	0x0022301C
Normal Interrupt Priority Level Register 7	NIPRIORITY7	0x00223020
Normal Interrupt Priority Level Register 6	NIPRIORITY6	0x00223024
Normal Interrupt Priority Level Register 5	NIPRIORITY5	0x00223028
Normal Interrupt Priority Level Register 4	NIPRIORITY4	0x0022302C
Normal Interrupt Priority Level Register 3	NIPRIORITY3	0x00223030
Normal Interrupt Priority Level Register 2	NIPRIORITY2	0x00223034
Normal Interrupt Priority Level Register 1	NIPRIORITY1	0x00223038
Normal Interrupt Priority Level Register 0	NIPRIORITY0	0x0022303C
Normal Interrupt Vector and Status Register	NIVECSR	0x00223040
Fast Interrupt Vector and Status Register	FIVECSR	0x00223044
Interrupt Source Register High	INTSRCH	0x00223048
Interrupt Source Register Low	INTSRCL	0x0022304C
Interrupt Force Register High	INTFRCH	0x00223050
Interrupt Force Register Low	INTFRCL	0x00223054
Normal Interrupt Pending Register High	NIPNDH	0x00223058
Normal Interrupt Pending Register Low	NIPNDL	0x0022305C
Fast Interrupt Pending Register High	FIPNDH	0x00223060
Fast Interrupt Pending Register Low	FIPNDL	0x00223064

Table 10-3. Register Field Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTCNTL	R	0	0	0	0	0	0	0	0	0	0	0	NIAD	FIAD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																																
NIMASK	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NIMASK				
	W																																
INTENNUM	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																											ENNUM					
INTDISNUM	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																											DISNUM					
INTENABLEH	R	INTENABLE [63:32]																															
	W																																
INTENABLEL	R	INTENABLE [31:0]																															
	W																																
INTTYPEH	R	INTTYPE [63:32]																															
	W																																
INTTYPEL	R	INTTYPE [31:0]																															
	W																																
NIPRIORITY7	R	NIPR63				NIPR62				NIPR61				NIPR60				NIPR59				NIPR58				NIPR57				NIPR56			
	W																																
NIPRIORITY6	R	NIPR55				NIPR54				NIPR53				NIPR52				NIPR51				NIPR50				NIPR49				NIPR48			
	W																																
NIPRIORITY5	R	NIPR47				NIPR46				NIPR45				NIPR44				NIPR43				NIPR42				NIPR41				NIPR40			
	W																																
NIPRIORITY4	R	NIPR39				NIPR38				NIPR37				NIPR36				NIPR35				NIPR34				NIPR33				NIPR32			
	W																																
NIPRIORITY3	R	NIPR31				NIPR30				NIPR29				NIPR28				NIPR27				NIPR26				NIPR25				NIPR24			
	W																																
NIPRIORITY2	R	NIPR23				NIPR22				NIPR21				NIPR20				NIPR19				NIPR18				NIPR17				NIPR16			
	W																																
NIPRIORITY1	R	NIPR15				NIPR14				NIPR13				NIPR12				NIPR11				NIPR10				NIPR9				NIPR8			
	W																																
NIPRIORITY0	R	NIPR7				NIPR6				NIPR5				NIPR4				NIPR3				NIPR2				NIPR1				NIPR0			
	W																																
NIVECSR	R	NIVECTOR																NIPRILVL															
	W																																
FIVECSR	R	FIVECTOR																															
	W																																

Table 10-3. Register Field Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTSRCH	R	INTIN [63:32]																															
	W																																
INTSRCL	R	INTIN [31:0]																															
	W																																
INTFRCH	R	FORCE [63:32]																															
	W																																
INTFRCL	R	FORCE [31:0]																															
	W																																
NIPNDH	R	NIPEND [63:32]																															
	W																																
NIPNDL	R	NIPEND [31:0]																															
	W																																
FIPNDH	R	FIPEND [63:32]																															
	W																																
FIPNDL	R	FIPEND [31:0]																															
	W																																

10.4.1 Interrupt Control Register

The Interrupt Control Register (INTCNTL) is located on the ARM920T processor's native bus, is accessible in 1 cycle, and can be accessed only in supervisor mode. This register must be accessed only on word (32-bit) boundaries.

INTCNTL Interrupt Control Register **Addr 0x00223000**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
												NIAD	FIAD			
TYPE	r	r	r	r	r	r	r	r	r	r	r	rw	rw	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-4. Interrupt Control Register Description

Name	Description	Settings
Reserved Bits 31–21	Reserved—These bits are reserved and should read 0.	
NIAD Bit 20	**Normal Interrupt Arbiter Disable**—Enables/Disables the assertion of a bus request to the ARM9 core when the normal interrupt signal (nIRQ) is asserted. When an alternate master has ownership of the bus when a normal interrupt occurs, the bus is given back to the ARM9 core after the DMA device has completed its accesses, so the IRQ_DIS bit does not affect alternate master accesses that are in progress. **Note:** To prevent an alternate master from accessing the bus during an interrupt service routine, do not clear the interrupt flag until the end of the service routine.	0 = Disregard the normal interrupt flag when evaluating bus requests 1 = Normal interrupt flag prevents alternate masters from accessing the system bus
FIAD Bit 19	**Fast Interrupt Arbiter Disable**—Enables/Disables the assertion of a bus request to the ARM9 core when the fast interrupt signal (nFIQ) is asserted. When an alternate master has ownership of the bus when a fast interrupt occurs, the bus is given back to the ARM9 core after the DMA device has completed its accesses, so the IRQ_DIS bit does not affect alternate master accesses that are in progress. **Note:** To prevent an alternate master from accessing the bus during an interrupt service routine, do not clear the interrupt flag until the end of the service routine.	0 = Disregard the fast interrupt flag when evaluating bus requests 1 = Fast interrupt flag prevents alternate masters from accessing the system bus
Reserved Bits 18–0	Reserved—These bits are reserved and should read 0.	

10.4.2 Normal Interrupt Mask Register

The Normal Interrupt Mask Register (NIMASK) controls the normal interrupt mask level. All normal interrupts with a priority level less than or equal to the NIMASK are disabled. The priority levels of normal interrupts are determined by the normal interrupt priority level registers (NIPRIORITY7, NIPRIORITY6, NIPRIORITY5, NIPRIORITY4, NIPRIORITY3, NIPRIORITY2, NIPRIORITY1, and NIPRIORITY0). The reset state of this register does not disable any normal interrupts.

Writing all 1's, or –1, to the NIMASK sets the normal interrupt mask to –1 and does not disable any normal interrupt priority levels.

This hardware mechanism creates reentrant normal interrupt routines by disabling lower priority normal interrupts. Refer to Section 10.5.6, "Writing Reentrant Normal Interrupt Routines," on page 10-35 for more details on the use of the NIMASK register.

This register is located on the ARM920T processor's native bus, is accessible in 1 cycle, and can be accessed only in supervisor mode. This register must be accessed only on word (32-bit) boundaries.

NIMASK Normal Interrupt Mask Register **Addr 0x00223004**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												NIMASK				
TYPE	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
								0x001F								

Table 10-5. Normal Interrupt Mask Register Description

Name	Description	Settings
Reserved Bits 31–5	Reserved—These bits are reserved and should read 0.	
NIMASK Bits 4–0	**Normal Interrupt Mask**—Controls normal interrupt mask level. All normal interrupts of priority level less than or equal to the NIMASK are disabled. Settings are shown in decimal. Setting bit 4 disables all normal interrupts.	0 = Disable priority level 0 normal interrupts 1 = Disable priority level 1 and lower normal interrupts ... 16+ = Disable all normal interrupts.

10.4.3 Interrupt Enable Number Register

The Interrupt Enable Number Register (INTENNUM) provides hardware accelerated enabling of interrupts. Any write to INTENNUM enables one interrupt source. For example, when the 6 LSBs = 000000, interrupt source 0 is enabled; when the 6 LSBs = 000001, interrupt source 1 is enabled, and so forth. This register is decoded into a single hot mask that is logically ORed with the INTENABLEH and the INTENABLEL registers.

This hardware mechanism removes the requirement for an atomic read/modify/write sequence to enable an interrupt source. For example, to enable interrupts 10 and 20, the software performs two writes to the AITC: first write 10, then write 20 to the INTENNUM register (the order of the writes is irrelevant to the AITC).

This register is located on the ARM920T processor's native bus, is accessible in 1 cycle, and can be accessed only in supervisor mode. This register must be accessed only on word (32-bit) boundaries. This register is self-clearing and therefore always reads back all 0s.

INTENNUM Interrupt Enable Number Register **Addr 0x00223008**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											ENNUM					
TYPE	r	r	r	r	r	r	r	r	r	r	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 10-6. Interrupt Enable Number Register Description

Name	Description	Settings
Reserved Bits 31–6	Reserved—These bits are reserved and should read 0.	
ENNUM Bits 5–0	**Interrupt Enable Number**—Enables/Disables the interrupt source associated with this value.	0x00 = Enable interrupt source 0 0x01 = Enable interrupt source 1 ... 0x3F = Enable interrupt source 63

10.4.4 Interrupt Disable Number Register

The Interrupt Disable Number Register (INTDISNUM) provides hardware accelerated disabling of interrupts. Any write to this register disables one interrupt source. When the 6 LSBs = 000000, then interrupt source 0 is disabled; when the 6 LSBs = 000001, then interrupt source 1 is disabled, and so forth. This register is decoded into a single hot mask that is inverted and logically ANDed with the INTENABLEH and the INTENABLEL registers.

This hardware mechanism removes the requirement for an atomic read/modify/write sequence to disable an interrupt source. To disable interrupts 10 and 20, the software performs two writes to the AITC: first write 10, then write 20 to INTDISNUM register (the order of the writes is irrelevant to the AITC).

This register is located on the ARM920T processor's native bus, is accessible in 1 cycle, and can be accessed only in supervisor mode. This register must be accessed only on word (32-bit) boundaries. This register is self-clearing and therefore always reads back all 0s.

INTDISNUM Interrupt Disable Number Register **Addr 0x0022300C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											DISNUM					
TYPE	r	r	r	r	r	r	r	r	r	r	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-7. Interrupt Disable Number Register Description

Name	Description	Settings
Reserved Bits 31–6	Reserved—These bits are reserved and should read 0.	
DISNUM Bits 5–0	**Interrupt Disable Number**—Enables/Disables the interrupt source associated with this value.	0x00 = Disable interrupt source 0 0x01 = Disable interrupt source 1 ... 0x3F = Disable interrupt source 63

10.4.5 Interrupt Enable Register High and Interrupt Enable Register Low

The Interrupt Enable Register High (INTENABLEH) and the Interrupt Enable Register Low (INTENABLEL) registers enable pending interrupt requests to the ARM920T processor. Each bit in these registers corresponds to an interrupt source available in the system. The reset state of both registers is all interrupts masked.

These registers are updated by the following methods:

- Write directly to the INTENABLEH and INTENABLEL registers
- Set bits with the INTENNUM register
- Clear bits with the INTDISNUM register

These registers are located on the ARM920T processor's native bus, are accessible in 1 cycle, and can be accessed only in supervisor mode. These registers must be accessed only on word (32-bit) boundaries.

10.4.5.1 Interrupt Enable Register High

INTENABLEH Interrupt Enable Register High **Addr 0x00223010**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	INTENABLE [63:48]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	INTENABLE [47:32]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-8. Interrupt Enable Register High Description

Name	Description	Settings
INTENABLE Bits 31–0	**Interrupt Enable**—Enables/Disables the individual bit interrupt sources to request a normal interrupt or a fast interrupt. When INTENABLE is set and the corresponding interrupt source is asserted, the interrupt controller asserts a normal or a fast interrupt request depending on the associated INTTYPEH and INTTYPEL setting.	0 = Interrupt disabled 1 = Interrupt enabled and generates a normal or fast interrupt upon assertion

10.4.5.2 Interrupt Enable Register Low

INTENABLEL Interrupt Enable Register Low **Addr 0x00223014**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	INTENABLE [31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	INTENABLE [15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-9. Interrupt Enable Register Low Description

Name	Description	Settings
INTENABLE Bits 31–0	**Interrupt Enable**—Enables/Disables the individual bit interrupt sources to request a normal interrupt or a fast interrupt. When INTENABLE is set and the corresponding interrupt source is asserted, the interrupt controller asserts a normal or a fast interrupt request depending on the associated INTTYPEH and INTTYPEL setting.	0 = Interrupt disabled 1 = Interrupt enabled and generates a normal or fast interrupt upon assertion

10.4.6 Interrupt Type Register High and Interrupt Type Register Low

The Interrupt Type Register High (INTTYPEH) and the Interrupt Type Register Low (INTTYPEL) registers select whether a pending interrupt source, when enabled with the INTENABLEH and INTENABLEL registers, creates a normal interrupt or a fast interrupt to the ARM920T processor. Each bit in this register corresponds to an interrupt source available in the system. The reset state of both registers is all interrupts generate a normal interrupt.

These registers are located on the ARM920T processor's native bus, are accessible in 1 cycle, and can be accessed only in supervisor mode. These registers must be accessed only on word (32-bit) boundaries.

10.4.6.1 Interrupt Type Register High

INTTYPEH Interrupt Type Register High **Addr 0x00223018**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	INTTYPE [63:48]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	INTTYPE [47:32]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-10. Interrupt Type Register High Description

Name	Description	Settings
INTTYPE Bits 31–0	**Interrupt Type**—Controls whether the individual interrupt sources request a normal interrupt or a fast interrupt. When a INTTYPE bit is set and the corresponding interrupt source is asserted, the interrupt controller asserts a fast interrupt request.	0 = Interrupt source generates a normal interrupt (nIRQ) 1 = Interrupt source generates a fast interrupt (nFIQ)

10.4.6.2 Interrupt Type Register Low

INTTYPEL Interrupt Type Register Low **Addr 0x0022301C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	INTTYPE [31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	INTTYPE [15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-11. Interrupt Type Register Low Description

Name	Description	Settings
INTTYPE Bits 31–0	**Interrupt Type**—Controls whether the individual interrupt sources request a normal interrupt or a fast interrupt. When a bit is set in INTTYPE and the corresponding interrupt source is asserted, the interrupt controller asserts a fast interrupt request.	0 = Interrupt source generates a normal interrupt (nIRQ) 1 = Interrupt source generates a fast interrupt (nFIQ)

10.4.7 Normal Interrupt Priority Level Registers

The normal interrupt priority level registers (NIPRIORITY7, NIPRIORITY6, NIPRIORITY5, NIPRIORITY4, NIPRIORITY3, NIPRIORITY2, NIPRIORITY1, and NIPRIORITY0) provide a software controllable prioritization of normal interrupts. Normal interrupts with a higher priority level preempt normal interrupts with a lower priority. The reset state of these registers forces all normal interrupts to the lowest priority level.

When a level 0 normal interrupt and a level 1 normal interrupt are asserted at the same time, the level 1 normal interrupt is selected assuming that NIMASK has not disabled level 1 normal interrupts. When two level 1 normal interrupts are asserted at the same time, the level 1 normal interrupt with the highest source number is selected, also assuming that NIMASK has not disabled level 1 normal interrupts.

These registers are located on the ARM920T processor's native bus, are accessible in 1 cycle, and can be accessed only in supervisor mode. These registers must be accessed only on word (32-bit) boundaries.

10.4.7.1 Normal Interrupt Priority Level Register 7

NIPRIORITY7 Normal Interrupt Priority Level Register 7 **Addr 0x00223020**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NIPR63				NIPR62				NIPR61				NIPR60			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NIPR59				NIPR58				NIPR57				NIPR56			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-12. Normal Interrupt Priority Level Register 7 Description

Name	Description	Settings
NIPR63 Bits 31–28	**Normal Interrupt Priority Level**—Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities.	0000 = Lowest priority normal interrupt ... 1111 = Highest priority normal interrupt
NIPR62 Bits 27–24		
NIPR61 Bits 23–20		
NIPR60 Bits 19–16		
NIPR59 Bits 15–12		
NIPR58 Bits 11–8		
NIPR57 Bits 7–4		
NIPR56 Bits 3–0		

10.4.7.2 Normal Interrupt Priority Level Register 6

NIPRIORITY6 Normal Interrupt Priority Level Register 6 **Addr 0x00223024**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NIPR55				NIPR54				NIPR53				NIPR52			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	l6	5	4	3	2	1	0
	NIPR51				NIPR50				NIPR49				NIPR48			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-13. Normal Interrupt Priority Level Register 6 Description

Name	Description	Settings
NIPR55 Bits 31–28	**Normal Interrupt Priority Level**—Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities.	0000 = Lowest priority normal interrupt ... 1111 = Highest priority normal interrupt
NIPR54 Bits 27–24		
NIPR53 Bits 23–20		
NIPR52 Bits 19–16		
NIPR51 Bits 15–12		
NIPR50 Bits 11–8		
NIPR49 Bits 7–4		
NIPR48 Bits 3–0		

10.4.7.3 Normal Interrupt Priority Level Register 5

NIPRIORITY5 Normal Interrupt Priority Level Register 5 **Addr 0x00223028**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NIPR47				NIPR46				NIPR45				NIPR44			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NIPR43				NIPR42				NIPR41				NIPR40			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-14. Normal Interrupt Priority Level Register 5 Description

Name	Description	Settings
NIPR47 Bits 31–28	**Normal Interrupt Priority Level**—Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities.	0000 = Lowest priority normal interrupt ... 1111 = Highest priority normal interrupt
NIPR46 Bits 27–24		
NIPR45 Bits 23–20		
NIPR44 Bits 19–16		
NIPR43 Bits 15–12		
NIPR42 Bits 11–8		
NIPR41 Bits 7–4		
NIPR40 Bits 3–0		

10.4.7.4 Normal Interrupt Priority Level Register 4

NIPRIORITY4 Normal Interrupt Priority Level Register 4 **Addr 0x0022302C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NIPR39				NIPR38				NIPR37				NIPR36			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NIPR35				NIPR34				NIPR33				NIPR32			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-15. Normal Interrupt Priority Level Register 4 Description

Name	Description	Settings
NIPR39 Bits 31–28	**Normal Interrupt Priority Level**—Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities.	0000 = Lowest priority normal interrupt ... 1111 = Highest priority normal interrupt
NIPR38 Bits 27–24		
NIPR37 Bits 23–20		
NIPR36 Bits 19–16		
NIPR35 Bits 15–12		
NIPR34 Bits 11–8		
NIPR33 Bits 7–4		
NIPR32 Bits 3–0		

10.4.7.5 Normal Interrupt Priority Level Register 3

NIPRIORITY3 Normal Interrupt Priority Level Register 3 **Addr 0x00223030**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NIPR31				NIPR30				NIPR29				NIPR28			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NIPR27				NIPR26				NIPR25				NIPR24			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-16. Normal Interrupt Priority Level Register 3 Description

Name	Description	Settings
NIPR31 Bits 31–28	**Normal Interrupt Priority Level**—Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities.	0000 = Lowest priority normal interrupt ... 1111 = Highest priority normal interrupt
NIPR30 Bits 27–24		
NIPR29 Bits 23–20		
NIPR28 Bits 19–16		
NIPR27 Bits 15–12		
NIPR26 Bits 11–8		
NIPR25 Bits 7–4		
NIPR24 Bits 3–0		

10.4.7.6 Normal Interrupt Priority Level Register 2

NIPRIORITY2 Normal Interrupt Priority Level Register 2 **Addr 0x00223034**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NIPR23				NIPR22				NIPR21				NIPR20			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NIPR19				NIPR18				NIPR17				NIPR16			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-17. Normal Interrupt Priority Level Register 2 Description

Name	Description	Settings
NIPR23 Bits 31–28	**Normal Interrupt Priority Level**—Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities.	0000 = Lowest priority normal interrupt ... 1111 = Highest priority normal interrupt
NIPR22 Bits 27–24		
NIPR21 Bits 23–20		
NIPR20 Bits 19–16		
NIPR19 Bits 15–12		
NIPR18 Bits 11–8		
NIPR17 Bits 7–4		
NIPR16 Bits 3–0		

10.4.7.7 Normal Interrupt Priority Level Register 1

NIPRIORITY1 Normal Interrupt Priority Level Register 1 **Addr 0x00223038**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NIPR15				NIPR14				NIPR13				NIPR12			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NIPR11				NIPR10				NIPR9				NIPR8			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-18. Normal Interrupt Priority Level Register 1 Description

Name	Description	Settings
NIPR15 Bits 31–28	**Normal Interrupt Priority Level**—Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities.	0000 = Lowest priority normal interrupt ... 1111 = Highest priority normal interrupt
NIPR14 Bits 27–24		
NIPR13 Bits 23–20		
NIPR12 Bits 19–16		
NIPR11 Bits 15–12		
NIPR10 Bits 11–8		
NIPR9 Bits 7–4		
NIPR8 Bits 3–0		

10.4.7.8 Normal Interrupt Priority Level Register 0

NIPRIORITY0 Normal Interrupt Priority Level Register 0 **Addr 0x0022303C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NIPR7				NIPR6				NIPR5				NIPR4			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NIPR3				NIPR2				NIPR1				NIPR0			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-19. Normal Interrupt Priority Level Register 0 Description

Name	Description	Settings
NIPR7 Bits 31–28	**Normal Interrupt Priority Level**—Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities.	0000 = Lowest priority normal interrupt ... 1111 = Highest priority normal interrupt
NIPR6 Bits 27–24		
NIPR5 Bits 23–20		
NIPR4 Bits 19–16		
NIPR3 Bits 15–12		
NIPR2 Bits 11–8		
NIPR1 Bits 7–4		
NIPR0 Bits 3–0		

10.4.8 Normal Interrupt Vector and Status Register

The Normal Interrupt Vector and Status Register (NIVECSR) specifies the priority of the highest pending normal interrupt and provides the vector index of the interrupt's service routine. This number can be directly used as an index into a vector table to select the highest pending normal interrupt source.

This read-only register is located on the ARM920T processor's native bus, is accessible in 1 cycle, and can be accessed only in supervisor mode. This register must be accessed only on word (32-bit) boundaries.

NIVECSR Normal Interrupt Vector and Status Register **Addr 0x00223040**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NIVECTOR															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0xFFFF															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NIPRILVL															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0xFFFF															

Table 10-20. Normal Interrupt Vector and Status Register Description

Name	Description	Settings
NIVECTOR Bits 31–16	**Normal Interrupt Vector**—Indicates vector index for the highest pending normal interrupt. Settings are shown in decimal.	0 = Interrupt 0 highest priority pending normal interrupt 1 = Interrupt 1 highest priority pending normal interrupt ... 63 = Interrupt 63 highest priority pending normal interrupt 64+ = No normal interrupt request pending
NIPRILVL Bits 15–0	**Normal Interrupt Priority Level**—Indicates the priority level of the highest priority normal interrupt. This number can be written to the NIMASK to disable the current priority normal interrupts to build a reentrant normal interrupt system. Settings are shown in decimal.	0 = Highest priority normal interrupt is level 0 1 = Highest priority normal interrupt is level 1 ... 15 = Highest priority normal interrupt is level 15 16+ = No normal interrupt request pending

10.4.9 Fast Interrupt Vector and Status Register

The Fast Interrupt Vector and Status Register (FIVECSR) specifies the priority of the highest pending fast interrupt and provides the vector index for the interrupt's service routine. This number can be directly used as an index into a vector table to select the highest pending fast interrupt source.

This read-only register is located on the ARM920T processor's native bus, is accessible in 1 cycle, and can be accessed only in supervisor mode. This register must be accessed only on word (32-bit) boundaries.

FIVECSR Fast Interrupt Vector and Status Register **Addr 0x00223044**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	FIVECTOR [31:16]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0xFFFF															

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

Table 10-21. Fast Interrupt Vector and Status Register Description

Name	Description	Settings
FIVECTOR Bits 31–0	**Fast Interrupt Vector**—Indicates vector index for the highest pending fast interrupt.	0 = Interrupt 0 is highest pending fast interrupt 1 = Interrupt 1 is highest pending fast interrupt ... 63 = Interrupt 63 is highest pending fast interrupt 64+ = not used, does not occur

10.4.10 Interrupt Source Register High and Interrupt Source Register Low

The Interrupt Source Register High (INTSRCH) and the Interrupt Source Register Low (INTSRCL) registers are each 32 bits wide. INTSRCH and INTSRCL reflect the status of all interrupt request inputs into the interrupt controller. Bit positions that are not used always read 0 (no request pending). The peripheral circuits generating the requests determine the state of this register out of reset; normally, the requests are inactive.

These read-only registers are located on the ARM920T processor's native bus, are accessible in 1 cycle, and can be accessed only in supervisor mode. These registers must be accessed only on word (32-bit) boundaries.

10.4.10.1 Interrupt Source Register High

INTSRCH Interrupt Source Register High **Addr 0x00223048**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	INTIN [63:48]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	INTIN [47:32]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-22. Interrupt Source Register High Description

Name	Description	Settings
INTIN Bits 31–0	**Interrupt Source**—Indicates the state of the corresponding hardware interrupt source.	0 = Interrupt source negated 1 = Interrupt source asserted

NOTE:

The peripheral circuits generating the requests determine the state of this register out of reset; normally, the requests are inactive. This read-only register must be accessed only on word (32-bit) boundaries.

10.4.10.2 Interrupt Source Register Low

INTSRCL Interrupt Source Register Low **Addr 0x0022304C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	INTIN [31:16]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	INTIN [15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-23. Interrupt Source Register Low Description

Name	Description	Settings
INTIN Bits 31–0	**Interrupt Source**—Indicates the state of the corresponding hardware interrupt source.	0 = Interrupt source negated 1 = Interrupt source asserted

NOTE:

The state of this register out of reset is determined by the peripheral circuits generating the requests; normally, the requests are inactive. This read-only register must be accessed only on word (32-bit) boundaries.

10.4.11 Interrupt Force Register High and Interrupt Force Register Low

The Interrupt Force Register High (INTFRCH) and the Interrupt Force Register Low (INTFRCL) registers are each 32 bits wide. The interrupt forces registers allow for software generation of interrupts for each of the possible interrupt sources for functional or debugging purposes. The system level design can reserve one or more sources for software purposes to allow software to self-schedule interrupts by forcing one or more of these sources in the appropriate interrupt force register(s).

These registers are located on the ARM920T processor's native bus, are accessible in 1 cycle, and can be accessed only in supervisor mode. These registers must be accessed only on word (32-bit) boundaries.

10.4.11.1 Interrupt Force Register High

INTFRCH Interrupt Force Register High **Addr 0x00223050**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	FORCE [63:48]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FORCE [47:32]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-24. Interrupt Force Register High Description

Name	Description	Settings
FORCE Bits 31–0	**Interrupt Source Force Request**—Forces a request for the corresponding interrupt source.	0 = Standard interrupt operation 1 = Interrupt forced asserted

10.4.11.2 Interrupt Force Register Low

INTFRCL Interrupt Force Register Low **Addr 0x00223054**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	FORCE [31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FORCE [15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-25. Interrupt Force Register Low Description

Name	Description	Settings
FORCE Bits 31–0	**Interrupt Source Force Request**—Forces a request for the corresponding interrupt source.	0 = Standard interrupt operation 1 = Interrupt forced asserted

10.4.12 Normal Interrupt Pending Register High and Normal Interrupt Pending Register Low

The Normal Interrupt Pending Register High (NIPNDH) and the Normal Interrupt Pending Register Low (NIPNDL) registers are 32-bits wide and monitor the outputs of the enable and masking operations. These registers are actually only a set of buffers, so the reset state of these registers is determined by the normal interrupt enable registers, the interrupt mask register, and the interrupt source registers.

These read-only registers are located on the ARM920T processor's native bus, are accessible in 1 cycle, and can be accessed only in supervisor mode. These registers must be accessed only on word (32-bit) boundaries.

10.4.12.1 Normal Interrupt Pending Register High

NIPNDH Normal Interrupt Pending Register High **Addr 0x00223058**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NIPEND [63:48]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NIPEND [47:32]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-26. Normal Interrupt Pending Register High Description

Name	Description	Settings
NIPEND Bits 31–0	**Normal Interrupt Pending Bit**—Indicates whether a normal interrupt is pending. When a normal interrupt enable bit is set and the corresponding interrupt source is asserted, the interrupt controller asserts a normal interrupt request. The normal interrupt pending bits reflect the interrupt input lines that are asserted and are currently enabled to generate a normal interrupt.	0 = No normal interrupt request 1 = Normal interrupt request pending

10.4.12.2 Normal Interrupt Pending Register Low

NIPNDL Normal Interrupt Pending Register Low **Addr 0x0022305C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NIPEND [31:16]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NIPEND [15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-27. Normal Interrupt Pending Register Low Description

Name	Description	Settings
NIPEND Bits 31–0	**Normal Interrupt Pending Bit**—Indicates whether a normal interrupt is pending. When a normal interrupt enable bit is set and the corresponding interrupt source is asserted, the interrupt controller asserts a normal interrupt request. The normal interrupt pending bits reflect the interrupt input lines that are asserted and are currently enabled to generate a normal interrupt.	0 = No normal interrupt request 1 = Normal interrupt request pending

10.4.13 Fast Interrupt Pending Register High and Fast Interrupt Pending Register Low

The Fast Interrupt Pending Register High (FIPNDH) and the Fast Interrupt Pending Register Low (FIPNDL) registers are 32-bits wide and monitor the outputs of the enable and masking operations. These registers are actually only a set of buffers, so the reset state of these registers is determined by the fast interrupt enable registers, the interrupt mask register, and the interrupt source registers.

These read-only registers are located on the ARM920T processor's native bus, are accessible in 1 cycle, and can be accessed only in supervisor mode. These registers must be accessed only on word (32-bit) boundaries.

10.4.13.1 Fast Interrupt Pending Register High

FIPNDH Fast Interrupt Pending Register High **Addr 0x00223060**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	FIPEND [63:48]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FIPEND [47:32]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-28. Fast Interrupt Pending Register High Description

Name	Description	Settings
FIPEND Bits 31–0	**Fast Interrupt Pending Bit**—Indicates if a fast interrupt request is pending. When a fast interrupt enable bit is set and the corresponding interrupt source is asserted, the interrupt controller asserts a fast interrupt request. The fast interrupt pending bits reflect the interrupt input lines that are asserted and are currently enabled to generate a fast interrupt.	0 = No fast interrupt request pending 1 = Fast interrupt request pending

10.4.13.2 Fast Interrupt Pending Register Low

FIPNDL Fast Interrupt Pending Register Low **Addr 0x00223064**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	FIPEND [31:16]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FIPEND [15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 10-29. Fast Interrupt Pending Register Low Description

Name	Description	Settings
FIPEND Bits 31–0	**Fast Interrupt Pending Bit**—Indicates if a fast interrupt request is pending. When a fast interrupt enable bit is set and the corresponding interrupt source is asserted, the interrupt controller asserts a fast interrupt request. The fast interrupt pending bits reflect the interrupt input lines that are asserted and are currently enabled to generate a fast interrupt.	0 = No fast interrupt request pending 1 = Fast interrupt request pending

10.5 ARM920T Processor Interrupt Controller Operation

This section discusses the ARM920T processor prioritization of various exceptions and interrupt sources, two methods of enabling or disabling interrupts, and provides a typical pipeline sequence.

10.5.1 ARM920T Processor Prioritization of Exception Sources

The ARM920T processor prioritizes the various exceptions as follows:

- Reset (highest priority)
- Data Abort
- Fast Interrupt
- Normal Interrupt
- Prefetch Abort
- Undefined Instruction and SWI (lowest priority)

10.5.2 AITC Prioritization of Interrupt Sources

The AITC module prioritizes the various interrupt sources by source number. Higher source numbers have higher priority. Fast interrupts always have higher priority than normal interrupts.

Interrupt requests are prioritized as follows:

1. Fast interrupt requests, in order of highest source number
2. Normal interrupt requests, in order of highest priority level, then in order of highest source number with the same priority level

10.5.3 Assigning and Enabling Interrupt Sources

The interrupt controller provides flexible assignment of any interrupt source to either of the two ARM920T processor interrupt request inputs. This is done by setting the appropriate bits in the INTENABLEH and INTENABLEL registers and the INTTYPEH and INTTYPEL registers. Interrupt assignment is usually done once during system initialization and does not affect interrupt latency.

Interrupt assignment is the first of three steps required to enable an interrupt source, and this is done by the MC9328MXL hardware. The second step is to program the source to generate interrupt requests. The final step is to enable the interrupt inputs in the ARM920T processor by clearing the normal interrupt disable (I) and/or the fast interrupt disable (F) bits in the processor status register (CPSR).

10.5.4 Enabling Interrupts Sources

There are two methods of enabling or disabling interrupts in the AITC. The first method is to directly read the INTENABLEH and INTENABLEL registers, logically OR or BIT CLEAR these registers with a generated mask, then write back to the INTENABLEH and INTENABLEL registers.

The second method is performing an atomic write to source number of the INTENNUM register. The AITC decodes this 6-bit register and enables one of the 64 interrupt sources. The AITC automatically generates a single hot enable mask and logically ORs this mask to the correct INTENABLEH and INTENABLEL register. To disable interrupts, the procedure is exactly the same except the source number is written to the INTDISNUM register.

10.5.5 Typical Interrupt Entry Sequences

The Table 10-30 is a typical pipeline sequence for the ARM920T processor when a normal interrupt occurs. Assuming single cycle memories, it takes approximately 6 clocks from the acknowledgement of the normal interrupt within the ARM920T processor until the first opcode of the interrupt routine is fetched.

Table 10-30. Typical Hardware Accelerated Normal Interrupt Entry Sequence

Address	Time										
	–2	–1	0	1	2	3	4	5	6	7	8
	nIRQ Assert		nIRQ ACK								
Last ADDR before nIRQ	Fetch	Dec	Exec	Link	Adjust						
+4 / +2		Fetch	Dec								
+8 / +4			Fetch								
0x00000018				Fetch	Dec	Exec	Data	Wrbk			
+4					Fetch	Dec					
+8						Fetch					
Vector Table							Vector				
N/A											
nIRQ Routine									Fetch	Dec	Exec
+4										Fetch	Dec
+8											Fetch

The Table 10-31 on page 10-34 is a typical pipeline sequence for the ARM920T processor when a fast interrupt occurs, assuming that the FIQ service routine begins at 0x0000001C and single cycle memories.

Table 10-31. Typical Fast Interrupt Entry Sequence

Address	Time					
	–2	–1	0	1	2	3
	nFIQ Assert		nFIQ ACK			
Last ADDR before nFIQ	Fetch	Dec	Exec	Link	Adjust	
+4 / +2		Fetch	Dec			
+8 / +4			Fetch			
0x0000001C				Fetch	Dec	Exec
+4					Fetch	Dec
+8						Fetch

10.5.6 Writing Reentrant Normal Interrupt Routines

The AITC can create a reentrant normal interrupt system. This enables preempting of lower priority level interrupts by higher priority level interrupts. This requires a small amount of software support and overhead. The following shows the steps necessary to accomplish this:

1. Push the link register (LR_IRQ) onto the stack (SP_IRQ).
2. Push the saved status register (SPSR_IRQ) onto the stack.
3. Read the current value of NIMASK and push this value onto the stack.
4. Read current priority level via NIVECSR.
5. Interrupts of the equal or lesser priority than the current priority level must be masked via the NIMASK register by writing value from NIVECSR.
6. Clear the I bit in the ARM920T processor via a MSR / MRS command sequence (a higher priority normal interrupt can preempt a lower priority one) and change the operating mode of the ARM920T processor to system mode from IRQ mode.
7. Push the System Mode Link Register (LR) onto the stack (SP_USER).
8. The traditional interrupt service routine is now included.
9. Pop the System Mode Link Register (LR) from the stack (SP_USER).
10. Set the I bit in the ARM920T processor via a MSR/MRS command sequence (disables all normal interrupts) and change the operating mode of the ARM920T processor to IRQ mode from system mode.
11. Pop the original value of the normal interrupt mask and write the value to the NIMASK register.
12. Pop the Saved Status Register from the stack (SP_IRQ).
13. Pop the link register from the stack into the PC.
14. Return from nIRQ.

NOTE:

These steps are still in development and are subject to change. Steps 1, 2, 13, and 14 are automatically done by most C compilers and are included for completeness.

Chapter 11
External Interface Module (EIM)

11.1 Overview

The External Interface Module (EIM) handles the interface to devices external to the MC9328MXL, including generation of chip selects for external peripherals and memory, and provides the following features:

- Six chip selects for external devices: $\overline{CS0}$, covering a range of 32 Mbyte, and $\overline{CS1}$–$\overline{CS5}$, covering a range of 16 Mbyte each
- Selectable protection for each chip select
- Reset programmable data port size for $\overline{CS0}$
- Programmable data port size for each chip select
- Address suppression during burst mode operations
- Synchronous burst mode support for burst flash devices
- Programmable wait-state generator for each chip select
- Supports big endian and little endian modes of operation
- Programmable general output capability for unused chip select outputs

11.2 EIM I/O Signals

The EIM I/O signals provide communication and control pathways between external devices and the MC9328MXL. A summary of the I/O signal pins is provided in Table 11-2 on page 11-4. Each signal is described in the following sections.

11.2.1 Address Bus

The A [24:0] signals are address bus outputs used to address external devices.

11.2.2 Data Bus

The D [31:0] signals are bidirectional data bus pins used to transfer data between the MC9328MXL and an external device.

11.2.3 Read/Write

The R/W output signal indicates if the current bus access is in a read or write cycle. A high (logic one) level indicates a read cycle, and a low (logic zero) level indicates a write cycle.

11.2.4 Control Signals

The OE and EB [3:0] signals are used to control external device's interface to the external data bus.

11.2.4.1 $\overline{\text{OE}}$—Output Enable

This active-low output signal indicates the bus access is a read, and enable slave devices to drive the data bus with read data.

11.2.4.2 $\overline{\text{EB}}$ [3:0]—Enable Bytes

These active-low output pins indicate active data bytes for the current access. They may be configured to assert for read and write cycles or for write cycles only as programmed in the CS configuration registers. EB [0] corresponds to D [31:24], EB [1] corresponds to D [23:16], EB [2] corresponds to D [15:8], and EB [3] corresponds to D [7:0]. In the case of a 16-bit operation, or controlling the enable byte signals for half-word operation, either EB[0] or EB[1] may be used for D[31:16], and either EB[2] or EB[3] may be used for D[15:0]. For word addressing, any of on the EB[3:0] signals may be used.

This is especially useful when interfacing to external devices or memories that require strict timing control over the write enable signal. Since there is no way to control the timing of the EIM WE signal, any one of the EB[3:0] signals (if not already being used for enable byte control), may be used as the write enable signal. The bits WEA and WEN in the Chip Select Control registers are used to vary the timing of these signals according to the bit settings in those registers. The corresponding EB[3:0] signal that can be used for a corresponding set of D[31:0] signals follows the EB[3:0] to D[31:0] mapping given above.

NOTE:

The pulse of OE, R/W, EB, and CS signals cannot be configured to less than one system clock period (HCLK).

11.2.4.3 $\overline{\text{DTACK}}$—Data Transfer Acknowledge

The DTACK signal is the external input data acknowledge signal that only supported by CS[5]. When the external DTACK signal is used as a data acknowledge signal, the bus time-out monitor generates a bus error when a bus cycle is not terminated by the external DTACK signal after 1022 clocks counts have elapsed.

The maximum wait state supported by the DTACK signal at 96 MHz is 10.645us. This can be calculated by dividing the number of maximum wait state cycles (in this case 1022) by the system clock frequency (HCLK). For designs requiring a longer wait state time greater than 10.645us, it is necessary to reduce the system clock frequency HCLK to an appropriate value that is less than 96 MHz. The system clock HCLK can be divided by setting the BLCK_DIV bits in the Clock Source Control Register to the desired value. For more detailed information about setting the BCLK_DIV bits, see Chapter 12, "Phase-Locked Loop and Clock Controller."

11.2.5 Chip Select Outputs

11.2.5.1 Chip Select 0 ($\overline{\text{CS0}}$)

The $\overline{\text{CS0}}$ output signal is active-low and is asserted based on a decode of internal address bus bits A[31:24] of the accessed address, and at reset is based on the value of the BOOTMOD [3:0] inputs. The port size is determined by the state of the BOOTMOD[3:0] inputs. See Section 8.2, "System Boot Mode Selection," on page 8-7 for more information.

11.2.5.2 Chip Select 1–Chip Select 5 ($\overline{CS1}$–$\overline{CS5}$)

The $\overline{CS1}$ through $\overline{CS5}$ output signals are active-low and are asserted based on a decode of the internal address bus bits A [31:24] of the accessed address. When disabled, these pins can be used as programmable general-purpose outputs. Table 11-1 specifies the address range for each Chip Select output.

Table 11-1. Chip Select Address Range

CSEN [x]	A [31:24]	Chip Select
Cleared	–	Inactive
Set	0001000x	$\overline{CS0}$
Set	00010010	$\overline{CS1}$
Set	00010011	$\overline{CS2}$
Set	00010100	$\overline{CS3}$
Set	00010101	$\overline{CS4}$
Set	00010110	$\overline{CS5}$

11.2.6 Burst Mode Signals

11.2.6.1 BCLK—Burst Clock

The BCLK output signal is used to clock external burst capable devices to synchronize the loading and incrementing of addresses, and delivery of burst read data to the EIM. Its behavior is affected by the BCM bit in the EIM configuration register and the SYNC, BCD, PME, and BCS bits in the EIM control registers.

11.2.6.2 $\overline{LBA}$—Load Burst Address

The $\overline{LBA}$ active-low output signal is asserted during burst mode accesses to cause the external burst capable device to load a new starting burst address. Assertion of $\overline{LBA}$ indicates that a valid address is present on the address bus. Its behavior is affected by the SYNC, BCD, PME, and BCS bits in the EIM control registers.

11.2.6.3 ECB—End Current Burst

The $\overline{ECB}$ active-low input signal is asserted by external burst capable devices to indicate the end of the current (continuous) burst sequence. Following assertion, the EIM terminates the current burst sequence and initiates a new one.

11.3 Pin Configuration for EIM

Table 11-2 lists the pins used for the EIM module. Many of these pins are multiplexed with other functions on the device, and must be configured for EIM operation.

Table 11-2. EIM Pin List

Pin Name	Direction	Description
External I/O Signals		
D [31:0]	Input/ Output	External 32-bit data bus
A [24:0]	Output	External address bus
$\overline{CS}$ [5:0]	Output	Active low external chip selects
DTACK	Input	External input data acknowledge signal for $\overline{CS5}$
$\overline{EB}$ [3:0]	Output	Active low external enable bytes signals. EB [0] controls D [31:24]*
$\overline{OE}$	Output	Active low output enable for external data bus
BCLK (burst clock)	Output	Clock for external synchronous memories (such as burst flash) - burst clock.
$\overline{LBA}$	Output	Active low signal sent to flash device causing the external device to latch the address.
$\overline{RW}$	Output	Indicates whether external access is a read (high) or write (low) cycle
$\overline{ECB}$	Input	Input signal identifies when to end an external burst access
*$\overline{EB}$ [1] controls D [23:16], $\overline{EB}$ [2] controls D [15:8], $\overline{EB}$ [3] controls D [7:0]		

NOTE:

The user must ensure that the data direction bits in the GPIO are set to the correct direction for proper operation. See Section 29.5.1, “Data Direction Registers,” on page 29-8 for details.

Table 11-3. Pin Configuration

Pins	Setting	Configuration Procedure
D [31:0]	Not Multiplexed	
A [24]	Primary function of GPIO Port A [0]	1. Clear bit 0 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 0 of Port A General Purpose Register (GPR_A)
A [23:16]	Primary function of GPIO Port A [31:24]	1. Clear bits [31:24] of Port A GPIO In Use Register (GIUS_A) 2. Clear bits [31:24] of Port A General Purpose Register (GPR_A)
A [15:1]	Not Multiplexed	
A [0]	Primary function of GPIO Port A [21]	1. Clear bit 21 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 21 of Port A General Purpose Register (GPR_A)
$\overline{CS}$ [5:4]	Primary function of GPIO Port A [23:22]	1. Clear bits [23:22] of Port A GPIO In Use Register (GIUS_A) 2. Clear bits [23:22] of Port A General Purpose Register (GPR_A)
$\overline{CS}$ [3]	Primary function of pin shared with SDRAM’s $\overline{CSD1}$	1. Clear bit 1 (SDCS1_SEL) of Function Muxing Control Register (FMCR)

Table 11-3. Pin Configuration (continued)

Pins	Setting	Configuration Procedure
$\overline{CS}$ [2]	Primary function of pin shared with SDRAM's $\overline{CSD0}$	1. Clear bit 0 (SDCS0_SEL) of Function Muxing Control Register (FMCR)
$\overline{CS}$ [1:0]	Not Multiplexed	
DTACK	Primary function of GPIO Port A[17	1. Clear bit 17 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 17 of the Port A General Purpose Register (GPR_A
$\overline{EB}$ [3:0]	Not Multiplexed	
$\overline{OE}$	Not Multiplexed	
BCLK	Primary function of GPIO Port A [18]	1. Clear bit 18 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 18 of Port A General Purpose Register (GPR_A)
$\overline{LBA}$	Primary function of GPIO Port A [19]	1. Clear bit 19 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 19 of Port A General Purpose Register (GPR_A)
$\overline{RW}$	Not Multiplexed	
$\overline{ECB}$	Primary function of GPIO Port A [20]	1. Clear bit 20 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 20 of Port A General Purpose Register (GPR_A)

11.4 Typical EIM System Connections

The following figures show example connections of the EIM with burst and asynchronous memories:

- Figure 11-1 illustrates a typical set of EIM interfaces to external memory and peripherals.
- Figure 11-2 illustrates the EIM interface to two supported external burst flash devices.

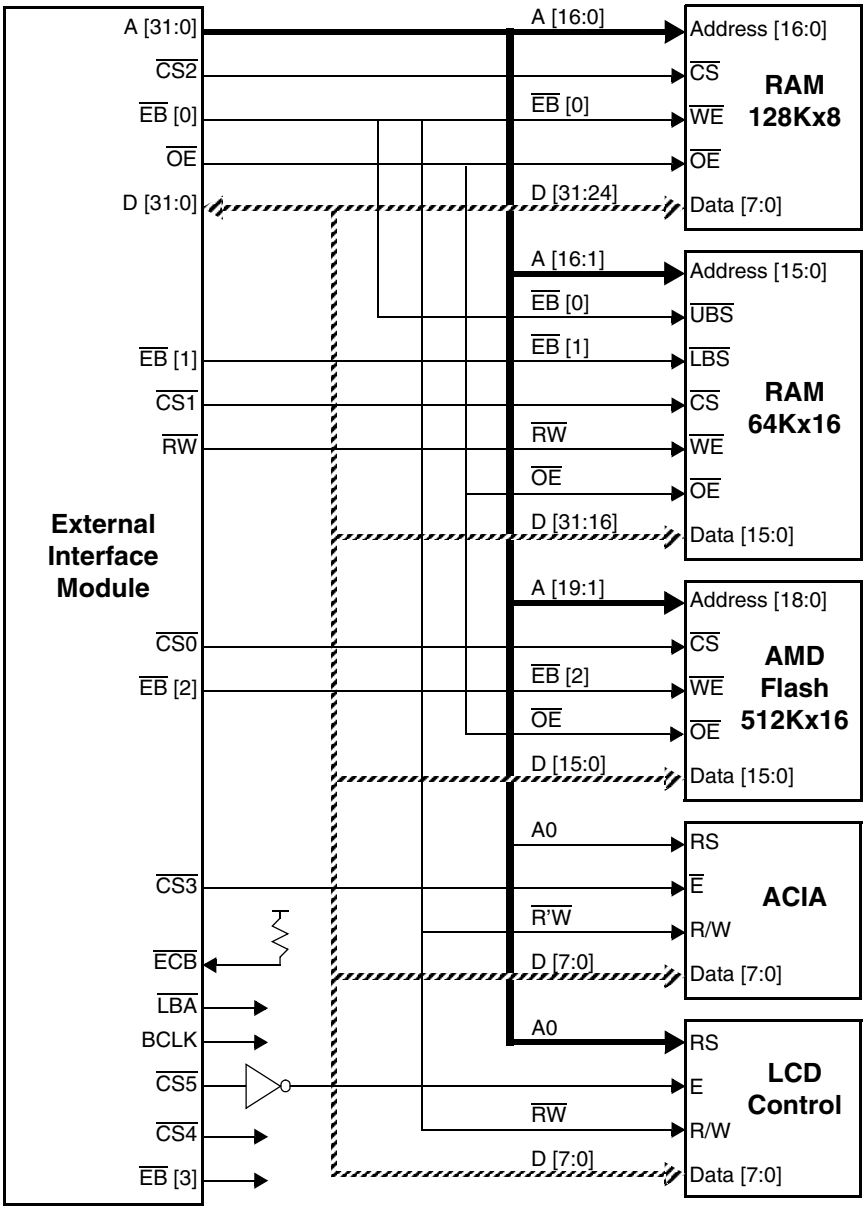


Figure 11-1. Example of EIM Interface to Memory and Peripherals

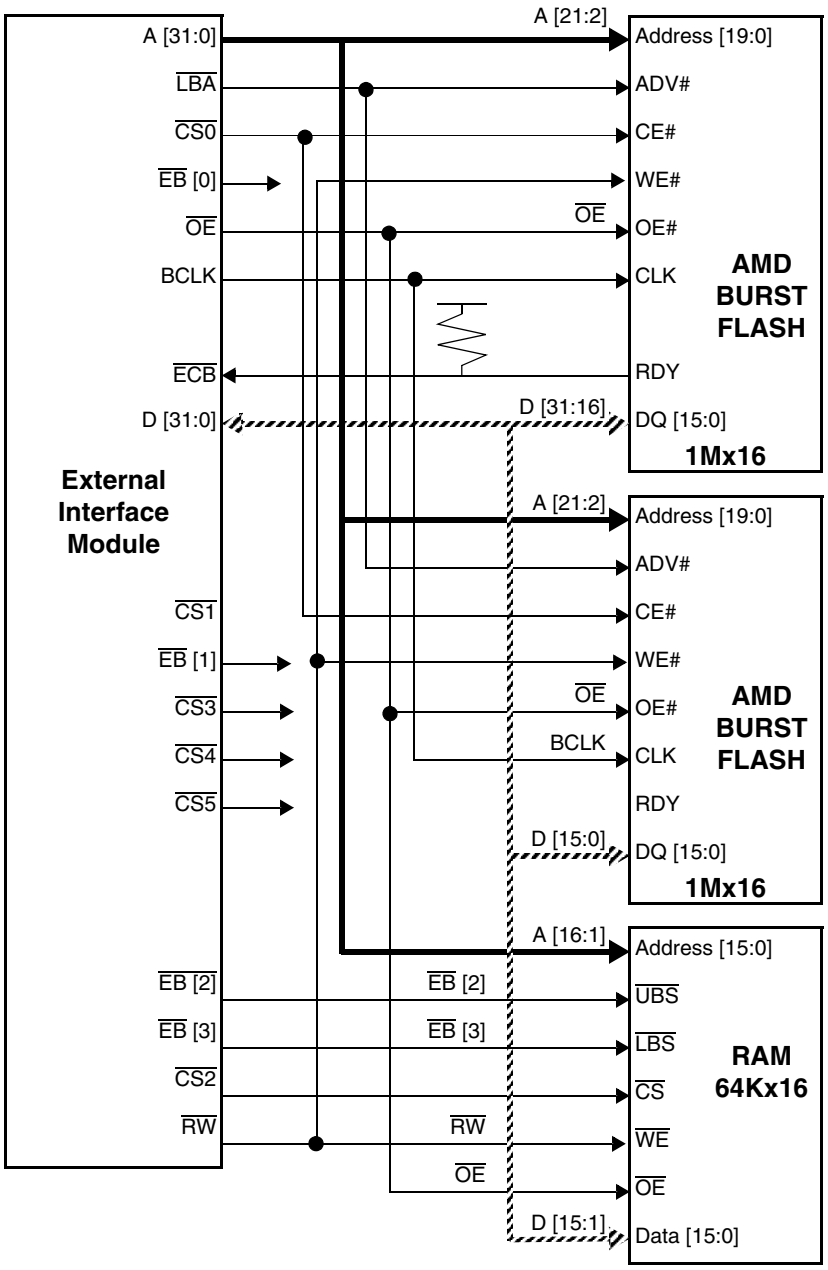


Figure 11-2. Example of EIM Interface to Burst Memory

11.5 EIM Functionality

11.5.1 Configurable Bus Sizing

The EIM supports byte, halfword, and word operands, allowing access to 8-bit ports, 16-bit ports, and 32-bit ports. It does not support misaligned transfers.

The port size is programmable via the DSZ bits in the corresponding chip select control register. In addition, the portion of the data bus used for transfer to or from an 8-bit port or 16-bit port is programmable via the same bits. An 8-bit port can reside on external data bus bits D [31:24], D [23:16], D [15:8] or D [7:0]. A 16-bit port can reside on external data bus bits D [31:16] or D [15:0].

Word access to or from an 8-bit port requires four external bus cycles to complete the transfer. Word access to or from a 16-bit port requires two external bus cycles to complete the transfer. Half-word access to or from an 8-bit port requires two external bus cycles to complete the transfer. In the case of a multi-cycle transfer, the lower two address bits, A [1:0], are incremented appropriately.

The EIM has a data multiplexer that routes the four bytes of the AHB interface data bus to the required positions to allow proper interfacing to memory and peripherals.

11.5.2 Programmable Output Generation

Unused chip select outputs can be configured to provide a programmable output signal. This functionality is not provided for the $\overline{CS}$ [0] output signal. When the CSEN bit is cleared, $\overline{CS}$ [0] is always inactive. To operate as a programmable output pin, the corresponding CSEN control bit must be cleared.

11.5.3 Burst Mode Operation

When burst mode is enabled, the EIM attempts to burst read data from as many sequential address locations as possible, limited only by the length of the burst flash internal page buffer, or the non-sequential nature of the ARM920T processor code or data stream. The EIM only displays the first address accessed in a burst sequence unless the page mode emulation (PME) bit is set.

For the first access in a burst sequence, the EIM asserts load burst address ($\overline{LBA}$), causing the external burst device to latch the starting burst address, and then toggles the burst clock (BCLK) for a predefined number of cycles to latch the first unit of data. Subsequently read data units are burst from the external device in fewer clock cycles, realizing an overall increase in bus bandwidth.

Burst accesses is terminated by the EIM when it detects that the next ARM920T processor access is not sequential in nature, or when the external burst device needs additional cycles to retrieve the next requested memory location. In the latter case, the burst flash device provides an $\overline{ECB}$ signal to the EIM whenever it must terminate the on-going burst sequence and initiate a new (long first access) burst sequence.

11.5.4 Burst Clock Divisor

In some cases it is necessary to slow the external bus in relation to the internal bus to allow accesses to burst devices that have a maximum operating frequency that is lower than the operating frequency of the internal AHB bus. The internal bus frequency can be divided by 2, 3, or 4 for presentation on the external bus in burst mode operation.

Programming the BCD bits to various values (see Table 11-5, "Chip Select Control Registers Description") affects two signals on the external bus, $\overline{\text{LBA}}$ (load burst address) and BCLK (burst clock). The $\overline{\text{LBA}}$ signal is asserted immediately and remains asserted until the first falling edge of the BCLK signal. The BCLK signal runs with a 50% duty cycle until a non-sequential internal request is received or an external $\overline{\text{ECB}}$ signal is recognized.

When programming these bits, ensure that the WSC and DOL fields are coordinated to provide the desired external bus waveforms. For example, when the BCD bits are programmed to 01, the DOL bits must be programmed to 0001, 0011, 0101, … . When the BCD bits are programmed to 10, the DOL must be programmed to 0010, 0101, 1000, … .

The BCM bit in the EIM configuration register has priority over the BCD bits. When BCM = 1, the BCLK runs at maximum frequency.

11.5.5 Burst Clock Start

To allow greater flexibility in achieving the minimum number of wait states on burst accesses, the user can determine when they want the BCLK (burst clock) to start toggling. This allows the BCLK to be skewed from the point of data capture on the system clock by any number of system clock phases. Use caution when setting these bits in conjunction with the BCD, WSC, and DOL bits. See the external timing diagrams for some examples of how to use the BCS, BCD, WSC, and DOL bits together.

11.5.6 Page Mode Emulation

Setting the PME bit causes the EIM to perform burst accesses by emulating page mode operation. The $\overline{\text{LBA}}$ signal remains asserted for the entire access, the burst clock does not send a signal, and the external address asserts for each access made. The initial access timing is dictated by the WSC bits and the page mode access timing is dictated by the DOL bits.

The EIM can take advantage of improved page timing for sequential accesses only. Accesses that are on the page, however are not sequential in nature, have their timing dictated by the WSC bits. The page size can be set via the PSZ bits to 4, 8, 16, or 32 words (the word size is determined by the data width of the external memory, such as the DSZ bits).

11.5.7 Error Conditions

The following conditions cause an error condition to be asserted to the ARM920T processor:

- Access to a disabled chip select (access to a mapped chip select address space where the CSEN bit in the corresponding chip select control register is clear)
- Write access to a write-protected chip select address space (the WP bit in the corresponding chip select control register is set)
- User access to a supervisor-protected chip select address space (the SP bit in the corresponding chip select control register is set)
- User read or write access to a chip select control register or the EIM configuration register
- Byte or halfword access to a chip select control register or the EIM configuration register

11.6 Programming Model

The EIM module includes thirteen user-accessible 32-bit registers. There is a common register called the EIM Configuration Register that contains control bits that configure the EIM for certain operation modes. The other twelve registers are pairs of control registers for each chip select. The layout of the control register is slightly different for the CS0 register output because $\overline{CS}$ [0] does not support the programmable output function. These registers are accessible only in supervisor mode with word (32-bit) reads and writes.

Complete decoding is not performed, so shadowing can occur with these registers. The user must not attempt to address these registers at any other address location other than those listed in Table 11-4.

Table 11-4. EIM Module Register Memory Map

Description	Name	Address
Chip Select 0 Upper Control Register	CS0U	0x00220000
Chip Select 0 Lower Control Register	CS0L	0x00220004
Chip Select 1 Upper Control Register	CS1U	0x00220008
Chip Select 1 Lower Control Register	CS1L	0x0022000C
Chip Select 2 Upper Control Register	CS2U	0x00220010
Chip Select 2 Lower Control Register	CS2L	0x00220014
Chip Select 3 Upper Control Register	CS3U	0x00220018
Chip Select 3 Lower Control Register	CS3L	0x0022001C
Chip Select 4 Upper Control Register	CS4U	0x00220020
Chip Select 4 Lower Control Register	CS4L	0x00220024
Chip Select 5 Upper Control Register	CS5U	0x00220028
Chip Select 5 Lower Control Register	CS5L	0x0022002C
EIM Configuration Register	EIM	0x00220030

11.6.1 Chip Select 0 Control Registers

The layout of the Chip Select 0 control registers are slightly different from the other Chip Select control registers because $\overline{CS}$ [0] does not support the programmable output function.

The 64 bits of control are divided into two registers, Chip Select 0 Upper Control Register and Chip Select 0 Lower Control Register.

- Bits [63:32] are located in Chip Select 0 Upper Control Register.
- Bits [31:0] are located in Chip Select 0 Lower Control Register.

11.6.1.1 Chip Select 0 Upper Control Register

CS0U Chip Select 0 Upper Control Register[1] **Addr 0x00220000**

BIT	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
			BCD		BCS				PSZ		PME	SYNC	DOL			
TYPE	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
	CNC		WSC							WWS			EDC			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	0x3E00															

[1]For bit descriptions, see Table 11-5 on page 11-13.

11.6.1.2 Chip Select 0 Lower Control Register

CS0L Chip Select 0 Lower Control Register[1] **Addr 0x00220004**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	OEA				OEN				WEA				WEN			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CSA				EBC	DSZ				SP		WP				CSEN
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	r	rw	r	rw	r	r	r	rw
RESET	0	0	0	0	1	*	*	*	0	0	0	0	0	0	0	1
	0x0801															

[1]For bit descriptions, see Table 11-5 on page 11-13.
*Configurable on reset.

11.6.2 Chip Select 1–Chip Select 5 Control Registers

The layout of the control registers for Chip Selects 1 through 5 are identical.

The 64 bits of control per chip select are divided into an Upper and a Lower register.

- Bits [63:32] are located in Chip Select x Upper Control Register.
- Bits [31:0] are located in Chip Select x Lower Control Register.

11.6.2.1 Chip Select 1–Chip Select 5 Upper Control Registers

For bit descriptions for all of these registers, see Table 11-5 on page 11-13.

11.6.2.2 Chip Select 1–Chip Select 5 Lower Control Registers

For bit descriptions for Chip Select 1–Chip Select 5 registers, see Table 11-5.

		Addr
CS1L	Chip Select 1 Lower Control Register	**0x0022000C**
CS2L	Chip Select 2 Lower Control Register	**0x00220014**
CS3L	Chip Select 3 Lower Control Register	**0x0022001C**
CS4L	Chip Select 4 Lower Control Register	**0x00220024**
CS5L	Chip Select 5 Lower Control Register	**0x0022002C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	OEA				OEN				WEA				WEN			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CSA				EBC	DSZ				SP		WP			PA	CSEN
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	r	rw	r	rw	r	r	rw	rw
RESET	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0
	0x0802															

Table 11-5. Chip Select Control Registers Description

Name	Description	Settings
DTACK_SEL Bit 63	**DTACK Select**—This bit is used to select the functionality of the DTACK input signal for CS5 to support either a generic DTACK signal or the Compact Flash/PCMCIA wait function. To select the DTACK functionality on CS5, the WSC bits for CS5 must be set to 111111.	0 = Generic DTACK function 1 = Compact Flash/PCMCIA wait function
BCD Bits 61–60	**Burst Clock Divisor**—Contains the value used to program the burst clock divisor. See Section 11.5.4, “Burst Clock Divisor,” for more information on the burst clock divisors. When the BCM bit is set (BCM = 1) in the EIM configuration register, BCD is ignored. BCD is cleared by a hardware reset.	00 = Divisor is 1 01 = Divisor is 2 10 = Divisor is 3 11 = Divisor is 4
BCS Bits 59–56	**Burst Clock Start**—Determines the number of half cycles after $\overline{\text{LBA}}$ assertion before the first rising edge of BCLK (burst clock) is seen. A value of 0 results in a half clock delay, not an immediate assertion. When the BCM bit is set (BCM = 1) in the EIM configuration register, this overrides the BCS bits. BCS is cleared by a hardware reset.	0000 = 1 half cycle before BCLK (burst clock) 0001 = 2 half cycles before BCLK (burst clock) ... 1111 = 16 half cycles before BCLK (burst clock)

Table 11-5. Chip Select Control Registers Description (continued)

Name	Description	Settings
PSZ Bits 55–54	**Page Size**—Indicates the number of words (where "word" is defined by the port size or DSZ bits) in a page in memory. This ensures that the EIM does not burst past a page boundary when the PME bit is set. PSZ is cleared by a hardware reset.	00 = 4 words in a page 01 = 8 words in a page 10 = 16 words in a page 11 = 32 words in a page
PME Bit 53	**Page Mode Emulation**—Enables/Disables page mode emulation in burst mode. When PME is set, the external address asserts for each piece of data requested. Additionally, the $\overline{\text{LBA}}$ and BCLK signals behave as they do when an asynchronous access is performed. PME is cleared by a hardware reset.	0 = Disables page mode emulation 1 = Enables page mode emulation
SYNC Bit 52	**Synchronous Burst Mode Enable**—Enables/Disables synchronous burst mode. When enabled, the EIM is capable of interfacing to burst flash devices through additional burst control signals: BCLK (burst clock), $\overline{\text{LBA}}$, and $\overline{\text{ECB}}$. The sequencing of these additional I/Os is controlled by other EIM configuration register bit settings as defined below. SYNC is cleared by a hardware reset.	0 = Disables synchronous burst mode 1 = Enables synchronous burst mode
DOL Bits 51–48	**Data Output Length**—Specifies the expected number of system clock cycles required for burst read data to be valid on the data bus before it is latched by the EIM. The reset value specifies that burst data is held for a single system clock period. As system clock frequencies increase, it may become necessary to delay sampling the data for multiple system clock periods to meet burst flash max frequency specifications and/or EIM data setup time requirements. DOL has no effect on EIM data latching when SYNC = 0. DOL is cleared by a hardware reset.	0000 = 1system clock delays 0001 = 1system clock delays 0010 = 2system clock delays 0011 = 3system clock delays ... 1111 = 15 system clock delays
CNC Bits 47–46	**Chip Select Negation Clock Cycles**—Specifies the minimum number of clock cycles a chip select must remain negated after it is negated. CNC has no effect on write accesses when any CSA bit is set. CNC is cleared by a hardware reset.	00 = Minimum negation is 0 clock cycles 01 = Minimum negation is 1 clock cycle 10 = Minimum negation is 2 clock cycles 11 = Minimum negation is 3 clock cycles

Table 11-5. Chip Select Control Registers Description (continued)

Name	Description	Settings
WSC Bits 45–40	**Wait State Control**— For SYNC = 0: WSC programs the number of wait-states for an access to the external device connected to the chip select. Table 11-6, "Chip Select Wait State and Burst Delay Encoding" shows the encoding of this field. When WWS is cleared, setting: • WSC = 000000 results in 2 clock transfers • WSC = 000001 results in 2 clock transfers • WSC = 001110 results in 15 clock transfers • WSC = 111110 results in 63 clock transfers • WSC=111111 selects DTACK input functionality for $\overline{CS5}$ For SYNC = 1: WSC programs the number of system clock cycles required for the initial access of a burst sequence initiated by the EIM to an external burst device. See Table 11-6, "Chip Select Wait State and Burst Delay Encoding" and to the EIM synchronous burst read timing diagrams for further details. to, the WSC WSC is set to 111110 by a hardware reset for $\overline{CS0}$. WSC is cleared by a hardware reset for $\overline{CS1}$–$\overline{CS5}$.	See Table 11-6, "Chip Select Wait State and Burst Delay Encoding"
Reserved Bit 39	Reserved—This bit is reserved and should read 0.	
WWS Bits 38–36	**Write Wait State**—Determines whether additional wait-states are required for write cycles. This is useful for writing to memories that require additional data setup time. WWS is cleared by a hardware reset.	See Table 11-6, "Chip Select Wait State and Burst Delay Encoding"
EDC Bits 35–32	**Extra Dead Cycles**—Determines whether idle cycles are inserted after a read cycle for back-to-back external transfers to eliminate data bus contention. This is useful for slow memory and peripherals that use long $\overline{CS}$ or $\overline{OE}$ to output data three-state times. Idle cycles are not inserted for back-to-back external reads from the same chip select. EDC is cleared by a hardware reset.	0000 = 0 Idle cycles inserted 0001 = 1 Idle cycle inserted ... 1111 = 15 Idle cycles inserted

Table 11-5. Chip Select Control Registers Description (continued)

Name	Description	Settings
OEA Bits 31–28	**$\overline{\text{OE}}$ Assert**—Determines when $\overline{\text{OE}}$ is asserted during read cycles. For SYNC = 0: OEA determines the number of half clocks before $\overline{\text{OE}}$ asserts during a read cycle. For SYNC = 1: After the initial burst access, $\overline{\text{OE}}$ is asserted continuously for subsequent burst accesses, and is not affected by OEA (see burst read timing diagrams for more detail). The behavior of $\overline{\text{OE}}$ on the initial burst access is the same as when SYNC = 0. When the EBC bit in the corresponding register is clear, the $\overline{\text{EB}}$ [3:0] outputs are similarly affected. The OEA bits do not affect the cycle length. OEA is cleared by a hardware reset.	0000 = 0 half clocks before assertion 0001 = 1 half clock before assertion ... 1111 = 15 half clocks before assertion
OEN Bits 27–24	**$\overline{\text{OE}}$ Negate**—Determines when $\overline{\text{OE}}$ is negated during a read cycle. Setting the SYNC bit (SYNC = 1) overrides OEN and $\overline{\text{OE}}$ negates at the end of a read access and no sooner. When EBC in the corresponding register is clear, the $\overline{\text{EB}}$ [3:0] outputs are similarly affected. OEN does not affect the cycle length. OEN is cleared by a hardware reset.	0000 = 0 half clocks before end of access 0001 = 1 half clock before end of access ... 1111 = 15 half clocks before end of access
WEA Bits 23–20	**$\overline{\text{EB}}$ [3:0] Assert**—Determines when $\overline{\text{EB}}$ [3:0] is asserted during write cycles. This is useful to meet data setup time requirements for slow memories. WEA does not affect the cycle length. WEA is cleared by a hardware reset.	0000 = 0 half clocks before assertion 0001 = 1 half clock before assertion ... 1111 = 15 half clocks before assertion
WEN Bits 19–16	**$\overline{\text{EB}}$ [3:0] Negate During Write**—Determines when $\overline{\text{EB}}$ [3:0] outputs are negated during a write cycle. This is useful to meet data hold time requirements for slow memories. WEN does not affect the cycle length. WEN is cleared by a hardware reset.	0000 = 0 half clocks before end of access 0001 = 1 half clock before end of access ... 1111 = 15 half clocks before end of access

Table 11-5. Chip Select Control Registers Description (continued)

Name	Description	Settings
CSA Bits 15–12	**Chip Select Assert**—Determines when chip select is asserted and negated for devices that require additional address setup time and additional address/data hold times. CSA affects only external writes, and is ignored on external reads. CSA does not affect the cycle length. CSA is cleared by a hardware reset.	0000 = 0 clocks before assertion and 0 clocks following negation 0001 = 1 clock before assertion and 1 clock following negation ... 1111 = 15 clocks before assertion and 15 clocks after negation
EBC Bit 11	**Enable Byte Control**—Indicates the access types that assert the enable byte outputs ($\overline{EB}$ [3:0]). EBC is set by a hardware reset.	0 = Both read and write accesses assert the $\overline{EB}$ [3:0] outputs, thus configuring the access as byte enables 1 = Only write accesses assert the $\overline{EB}$ [3:0] outputs, thus configuring the access as byte write enables; the $\overline{EB}$ [3:0] outputs are configured as byte write enables for accesses to dual x16 or quad x8 memories
DSZ Bits 10–8	**Data Port Size**—Defines the width of the external device's data port as shown in the table, DSZ Bit Encoding, to the right. At hardware reset, the value of DSZ is 000 for $\overline{CS1}$– $\overline{CS5}$. For $\overline{CS0}$, DSZ is mapped based on the value of the EIM_BOOT_DSZ [2:0] bits. EIM_BOOT_DSZ [2] maps to DSZ [2], EIM_BOOT_DSZ [1] maps to DSZ [1] and EIM_BOOT_DSZ [0] maps to DSZ [0].	000 = 8-bit port, resides on D [31:24] pins 001 = 8-bit port, resides on D [23:16] pins 010 = 8-bit port, resides on D [15:8] pins 011 = 8-bit port, resides on D [7:0] pins 100 = 16-bit port, resides on D [31:16] pins 101 = 16-bit port, resides on D [15:0] pins 11x = 32-bit port
Reserved Bit 7	Reserved—This bit is reserved and should read 0.	
SP Bit 6	**Supervisor Protect**—Prevents accesses to the address range defined by the corresponding chip select when the access is attempted in the User mode of ARM9 core operation. SPI is cleared by a hardware reset.	0 = User mode accesses are allowed in the range of chip select 1 = User mode accesses are prohibited; attempts to access an address mapped by this chip select in User mode results in a $\overline{TEA}$ to the ARM9 core and no assertion of the chip select output
Reserved Bit 5	Reserved—This bit is reserved and should read 0.	
WP Bit 4	**Write Protect**—Prevents writes to the address range defined by the corresponding chip select. WP is cleared by a hardware reset.	0 = Writes are allowed in the range of chip select 1 = Writes are prohibited; attempts to write to an address mapped by this chip select result in a $\overline{TEA}$ to the ARM9 core and no assertion of the chip select output
Reserved Bits 3–2	Reserved—These bits are reserved and should read 0.	

Table 11-5. Chip Select Control Registers Description (continued)

Name	Description	Settings
PA Bit 1	**Pin Assert**—Controls the chip select pin when it is operating as a programmable output pin (when the CSEN bit is clear). PA is not available (reserved) for $\overline{CS0}$. PA is set by a hardware reset for $\overline{CS1}$–$\overline{CS5}$.	0 = Brings the chip select output to logic-low 1 = Brings the chip select output to logic-high
CSEN Bit 0	**Chip Select Enable**—Controls the operation of the chip select pin. Except for $\overline{CS0}$, CSEN is cleared by reset, disabling the chip select output pin. When enabled, the PA control bit is ignored. CSEN in the CS0 control register is set at reset to allow $\overline{CS0}$ to select from an external boot ROM. CSEN is set by a hardware reset for $\overline{CS0}$. CSEN is cleared by a hardware reset for $\overline{CS1}$–$\overline{CS5}$.	0 = Chip select function is disabled; attempts to access an address mapped by this chip select results in an error and no assertion of the chip select output When disabled, the pin is a general purpose output controlled by the value of PA control bit. When CSEN in the CS0 control register is clear, $\overline{CS0}$ is inactive. 1 = Chip select is enabled, and is asserted when presented with a valid AHB access.

11.6.3 EIM Configuration Register

Table 11-6. Chip Select Wait State and Burst Delay Encoding

WSC [5:0]	Number of Wait-States					
	WWS = 0		WWS = 1		WWS = 7	
	Read Access	Write Access	Read Access	Write Access	Read Access	Write Access
000000	1	1	1	1	1	7
000001	1	1	1	2	1	8
000010	2	2	2	3	2	9
000011	3	3	3	4	3	10
000100	4	4	4	5	4	11
000101	5	5	5	6	5	12
000110	6	6	6	7	6	13
000111	7	7	7	8	7	14
001000	8	8	8	9	8	15
001001	9	9	9	10	9	16
001010	10	10	10	11	10	17
001011	11	11	11	12	11	18
001100	12	12	12	13	12	19
001101	13	13	13	14	13	20
001110	14	14	14	15	14	21
001111	15	15	15	16	15	22
010000	16	16	16	17	16	23
010001	17	17	17	18	17	24
010010	18	18	18	19	18	25
010011	19	19	19	20	19	26
010100	20	20	20	21	20	27
010101	21	21	21	22	21	28
010110	22	22	22	23	22	29
010111	23	23	23	24	23	30
011000	24	24	24	25	24	31
011001	25	25	25	26	25	32
011010	26	26	26	27	26	33
011011	27	27	27	28	27	34
011100	28	28	28	29	28	35
011101	29	29	29	30	29	36
011110	30	30	30	31	30	37

Table 11-6. Chip Select Wait State and Burst Delay Encoding (continued)

WSC [5:0]	Number of Wait-States					
	WWS = 0		WWS = 1		WWS = 7	
	Read Access	Write Access	Read Access	Write Access	Read Access	Write Access
011111	31	31	31	32	31	38
100000	32	32	32	33	32	39
100001	33	33	33	34	33	40
100010	34	34	34	35	34	41
100011	35	35	35	36	35	42
100100	36	36	36	37	36	43
100101	37	37	37	38	37	44
100110	38	38	38	39	38	45
100111	39	39	39	40	39	46
101000	40	40	40	41	40	47
101001	41	41	41	42	41	48
101010	42	42	42	43	42	49
101011	43	43	43	44	43	50
101100	44	44	44	45	44	51
101101	45	45	45	46	45	52
101110	46	46	46	47	46	53
101111	47	47	47	48	47	54
110000	48	48	48	49	48	55
110001	49	49	49	50	49	56
110010	50	50	50	51	50	57
110011	51	51	51	52	51	58
110100	52	52	52	53	52	59
110101	53	53	53	54	53	60
110110	54	54	54	55	54	61
110111	55	55	55	56	55	62
111000	56	56	56	57	56	63
111001	57	57	57	58	57	63
111010	58	58	58	59	58	63
111011	59	59	59	60	59	63
111100	60	60	60	61	60	63
111101	61	61	61	62	61	63
111110	62	62	62	63	62	63

Table 11-6. Chip Select Wait State and Burst Delay Encoding (continued)

WSC [5:0]	Number of Wait-States					
	WWS = 0		WWS = 1		WWS = 7	
	Read Access	Write Access	Read Access	Write Access	Read Access	Write Access
111111	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

The EIM Configuration Register contains the bit that controls the operation of the burst clock.

EIM EIM Configuration Register **Addr 0x00220030**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														BCM		
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 11-7. EIM Configuration Register Description

Name	Description	Settings
Reserved Bits 31–3	Reserved—These bits are reserved and should read 0.	
BCM Bit 2	**Burst Clock Mode**—Selects the burst clock mode of operation. BCM is cleared by a hardware reset.	0 = The burst clock runs only when accessing a chip select range with the SYNC bit set. When the burst clock is not running, it remains in a logic 0 state; when the burst clock is running, it is configured by the BCD and BCS bits in the chip select control register. 1 = The burst clock runs all the time (independent of chip select accesses).
Reserved Bits 1–0	Reserved—These bits are reserved and should read 0.	

Chapter 12
Phase-Locked Loop and Clock Controller

12.1 Introduction

To produce the wide range of on-chip clock frequencies required by the MC9328MXL, the core clock generator uses a two-stage phase locked loop (PLL). The first stage is a premultiplier PLL. If the input crystal frequency is 32.768 kHz, the premultiplier multiplies it by a factor of 512 to 16.78 MHz. If the input crystal frequency is 32 kHz, the premultiplier multiplies it to 16.384 MHz. The second stage is a digital PLL (DPLL) that produces an output frequency determined by system requirements and used throughout the entire system. The two DPLLs (MCU PLL and System PLL) use digital and mixed analog/digital chips to provide clock generation for wireless communication and other applications. Power management of the MC9328MXL is accomplished by controlling the operation of the premultiplier, MCU PLL and System PLL units, as shown in Figure 12-1 on page 12-2.

12.2 Clock Sources

The distribution of clocks in the MC9328MXL is shown in Figure 12-1 on page 12-2. Clock signal name definitions are provided in Table 12-1 on page 12-2.

The 32kHz clock source can be configured in two ways. The user can place a 32 kHz or 32.768 kHz crystal across EXTAL32K and XTAL32K with appropriate load capacitors. Alternately, an external 32 kHz or 32.768 kHz oscillator can be used; the signal is AC coupled into EXTAL32K and XTAL32K is floated.

The 16 MHz external crystal oscillator is not recommended for use and must be disabled using the CSCR register. Note that the 32 kHz oscillator must be used and satisfies all system requirements.

12.2.1 Low Frequency Clock Source

The MC9328MXL can use either a 32 kHz or a 32.768 kHz crystal as the external low frequency source. Throughout this chapter, the low frequency crystal is referred to as the 32 kHz crystal. The signal from the external 32 kHz crystal is the source of the CLK32 signal that is sent to the real time clock (RTC). The output of the 32 kHz crystal is also input to the premultiplier PLL to produce the 16.78 MHz signal that is input to the MCU PLL (it is 16.78 MHz if a 32.768 kHz crystal is used). The output of the MCU PLL is sent to the prescaler (PRESC) module to produce the fast clock (FCLK) signal for the ARMTDMI core.

The 16.384 MHz output of the premultiplier PLL also can be a source for the System PLL by setting the System_SEL bit in the Clock Source Control Register to produce all of the system clocks from a single 32 kHz crystal oscillator. See Section 12.3.1, “DPLL Phase and Frequency Jitter,” for more detailed information on phase and frequency jitter specifications using this configuration.

12.2.2 High Frequency Clock Source

The System PLL produces the USBPLLCLK signal that is the source for the following clock signals:

- CLK48M—for the USB
- HCLK and BCLK—HCLK is the MC9328MXL system clock and BCLK goes to the ARMTDMI core.
- Peripheral Clocks 1, 2, and 3—The peripheral clocks (PERCLK) provide clock signals to both integrated and external peripherals.

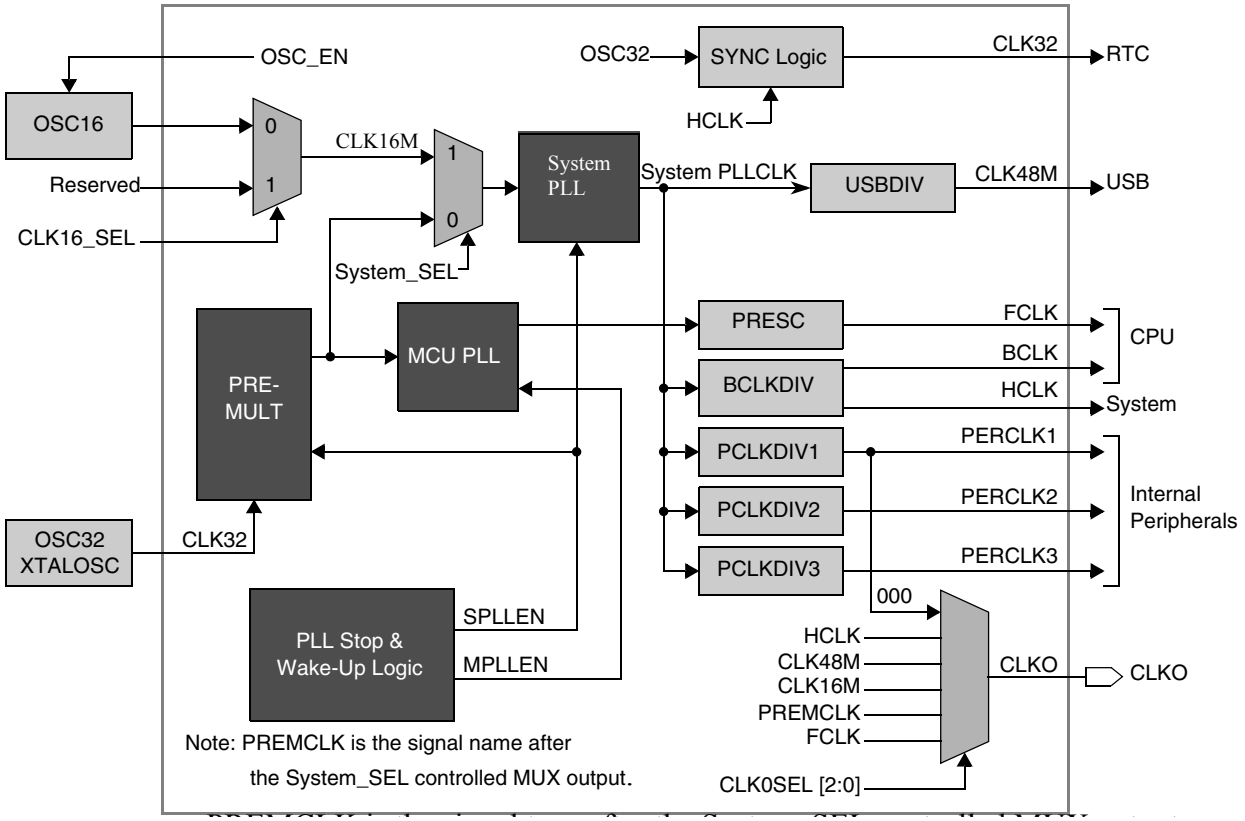


Figure 12-1. Clock Controller Module

Table 12-1. Clock Controller Module Signal Descriptions

Signal Names	I/O	Description	Default
CLK48M	O	Continuous 48 MHz clock output when System PLL is enabled or when external 48 MHz clock is selected.	Run
FCLK	O	Fast clock (FCLK) output to the CPU.	Run
HCLK	O	System clock (HCLK) output to the CPU (as BCLK) and to the system. This is a continuous clock (when the system is not in sleep mode) normally used for bus non-stop system logic (such as bus arbiter or interrupt controller) when the system is running.	Run
CLKO	O	Output internal clock to the CLKO pin.	Run

Table 12-1. Clock Controller Module Signal Descriptions (continued)

Signal Names	I/O	Description	Default
PERCLK1, 2, 3	O	Output clocks used by the peripheral modules.	Run

12.3 DPLL Output Frequency Calculation

The DPLL (both the MCU PLL and System PLL) produce a high frequency clock that exhibits both a low frequency jitter and a low phase jitter. The DPLL output clock frequency (f_{dpll}) is determined by the Equation 12-1:

$$f_{dpll} = 2f_{ref} \bullet \frac{MFI + MFN / (MFD+1)}{PD + 1} \qquad \textit{Eqn. 12-1}$$

where:

- f_{ref} is the reference frequency
- MFI is an integer part of a multiplication factor (MF)
- MFN is the numerator and MFD is the denominator of the MF
- PD is the predivider factor

12.3.1 DPLL Phase and Frequency Jitter

Spectral purity of the DPLL output clock is characterized by both phase and frequency jitter. Phase jitter is a measure of clock phase fluctuations relative to an ideal clock phase. The output clock also can be skewed relative to the reference clock. Frequency jitter is a measure of clock period fluctuations relative to an ideal clock period. Frequency jitter is calculated as a difference of phase jitter values for adjacent clocks.

DPLL jitter requirements vary according to system configuration. For many stand-alone processors and asynchronous multiprocessor applications, only the frequency jitter value is important (slow phase jitter and clock skew do not affect system performance). In these systems, it is not necessary to adjust the output clock phase with an input clock phase. The clock generation mode in which slow phase fluctuations are permissible is called Frequency Only Lock (FOL) mode.

Phase error is sometimes important for synchronous applications and sampling analog-to-digital (A/D) and digital-to-analog (D/A) precision converters. The DPLL mode providing minimum phase jitter and skew elimination is Frequency and Phase Lock (FPL) mode. The DPLL mode is user selectable.

The DPLL communicates with the clock module. This block contains a control register and provides an interface between the DPLL and the ARMTDMI core.

12.4 MC9328MXL Power Management

The operation of the PLL and clock controller at different stages of power management is described in the following sections.

12.4.1 PLL Operation at Power-Up

The crystal oscillator begins oscillating within several hundred milliseconds of initial power-up. While reset remains asserted, the PLL begins the lockup sequence and locks 1 ms after the crystal oscillator becomes stable. After lockup occurs, the system clock is available at the default System PLL output frequency of 96 MHz (when a 32 kHz crystal is used).

12.4.2 PLL Operation at Wake-Up

When the device is awakened from stop mode by a wake-up event, the DPLL locks within 300 µs. The crystal oscillator is always on after initial power-up, so crystal startup time is not a factor. The PLL output clock starts operating as soon as it achieves lock.

12.4.3 ARM920T Processor Low-Power Modes

The MC9328MXL provides two power saving modes, doze and stop:

- In stop mode, the MCU PLL and the System PLL are shut down and only the 32 kHz clock is running.
- In doze mode, the CPU executes a wait for interrupt instruction.

These modes are controlled by the clock control logic and a sequence of CPU instructions. Most of the peripheral modules can enable or disable the incoming clock signal (PERCLK 1, 2, or 3) through clock gating circuitry from the peripheral bus.

12.4.4 SDRAM Power Modes

When the SDRAM controller (SDRAMC) is enabled, the external SDRAM operates in distributed-refresh mode or in self-refresh mode (as shown in Table 12-2). The SDRAM wake-up latency is approximately 20 system clock cycles (HCLK). The SDRAMC can wake up from self-refresh mode when it is in a SDRAM cycle. In doze mode, the SDRAM enters self-refresh mode. When a bus cycle accesses the SDRAM or SyncFlash, the controller wakes up and completes the bus cycle, then returns to self-refresh mode.

Table 12-2. SDRAM/SyncFlash Operation During Power Modes

SDRAM	Run	Doze	Stop
SDRAM	Distributed-refresh	Self-refresh	Self-refresh
SyncFlash	Run	Low-power mode	Deep power-down mode

12.4.5 Power Management in the Clock Controller

Power management in the MC9328MXL is achieved by controlling the duty cycles of the clock system efficiently. The clocking control scheme is shown in Table 12-3.

Table 12-3. Power Management in the Clock Controller

Device/Signal	Shut-Down Conditions	Wake-Up Conditions
System PLL	When 0 is written to the SPEN bit and the PLL shut-down count times out (for details see the SD_CNT settings in Table 12-5 on page 12-6).	When $\overline{IRQ}$ or $\overline{FIQ}$ is asserted.
MCU PLL	When 0 is written to the MPEN bit.	When $\overline{IRQ}$ or $\overline{FIQ}$ is asserted, or 1 is written to the MPEN bit.
Premultiplier	Same as System PLL.	Same as System PLL.
CLK32	Continuously running.	Continuously running.

12.5 Programming Model

The PLL and Clock Controller module includes six user-accessible 32-bit registers. Table 12-4 summarizes these registers and their addresses.

Table 12-4. PLL and Clock Controller Module Register Memory Map

Description	Name	Address
Clock Source Control Register	CSCR	0x0021B000
Peripheral Clock Divider Register	PCDR	0x0021B020
MCU PLL Control Register 0	MPCTL0	0x0021B004
MCU PLL and System Clock Control Register 1	MPCTL1	0x0021B008
System PLL Control Register 0	SPCTL0	0x0021B00C
System PLL Control Register 1	SPCTL1	0x0021B010

12.5.1 Clock Source Control Register

This register controls the various clock sources to the internal modules of the MC9328MXL. It allows the bypass of the 32 kHz derived clock source to the System PLL when the design requires clock signals with greater frequency and phase jitter performance than the internal PLL using the 32 kHz clock source provides.

CSCR Clock Source Control Register **Addr 0x0021B000**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CLKO_SEL			USB_DIV			SD_CNT			SPLL_ RESTART	MPLL_ RESTART			CLK16 _SEL	OSC _EN	System _SEL
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	r	r	rw	rw	rw
RESET	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
	0x0F00															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PRESC		BCLK_DIV												SPEN	MPEN
TYPE	rw	r	rw	rw	rw	rw	r	r	r	r	r	r	r	r	rw	rw
RESET	1	0	1	0	1	1	0	0	0	0	0	0	0	0	1	1
	0xAC03															

Table 12-5. Clock Source Control Register Description

Name	Description	Settings
CLKO_SEL Bits 31–29	**CLKO Select**—Selects the clock signal source that is output on the CLKO pin.	000 = PERCLK1 001 = HCLK 010 = CLK48M 011 = CLK16M 100 = PREMCLK 101 = FCLK
USB_DIV Bits 28–26	**USB Divider**—Contains the integer divider value used to generate the CLK48M signal for the USB modules.	000 = System PLL clock divide by 1 001 = System PLL clock divide by 2 . . . 111 = System PLL clock divide by 8
SD_CNT Bits 25–24	**Shut-Down Control**—Contains the value that sets the duration of System PLL clock output after 0 is written to the SPEN bit. The power controller requests the bus before System PLL shutdown. Any unmasked interrupt event enables the System PLL.	00 = System PLL shuts down after next rising edge of CLK32 is detected and the current bus cycle is completed. A minimum of 16 HCLK cycles is guaranteed from writing ¡°0¡± to SPEN bit. 01 = System PLL shuts down after the second rising edge of CLK32 is detected. 10 = System PLL shuts down after the third rising edge of CLK32 is detected. 11 = System PLL shuts down after forth rising edge of CLK32 is detected.
Reserved Bit 23	Reserved—This bit is reserved and should read 0.	

Table 12-5. Clock Source Control Register Description (continued)

Name	Description	Settings
SPLL_ RESTART Bit 22	**SPLL Restart**—Restarts System PLL at new assigned frequency. SPLL_RESTART self-clears after 1 (min) or 2 (max) cycles of CLK32.	0 = No Effect 1 = Restarts System PLL at new frequency
MPLL_ RESTART Bit 21	**MPLL Restart**—Restarts the MCU PLL at a new assigned frequency. MPLL_RESTART self-clears after 1 (min) or 2 (max) cycles of CLK32.	0 = No Effect 1 = Restarts MCU PLL at new frequency
Reserved Bits 20–19	Reserved—These bits are reserved and should read 0.	
OSC_EN Bit 17	**Oscillator Enable**—Enables the 16 MHz oscillator circuit when set (available when using an external crystal input). When clear, the oscillator circuit control is disabled which bypasses the oscillator circuit when using external clock input. *This oscillator is not recommended for use.*	0 = Disable the external 16 MHz oscillator circuit (Recommended) 1 = Enable the external 16 MHz oscillator circuit
System_SEL Bit 16	**System Select**—Selects the clock source of the System PLL input. When set, the external high frequency clock input is selected.	0 = Clock source is the internal premultiplier 1 = Clock source is the external high frequency clock
PRESC Bit 15	**Prescaler**—Defines the MPU PLL clock prescaler.	0 = Prescaler divides by 1 1 = Prescaler divides by 2
Reserved Bit 14	Reserved—This bit is reserved and should read 0.	
BCLK_DIV Bits 13–10	**BClock Divider**—Contains the 4-bit integer divider values for the generation of the BCLK and HCLK.	0000 = System PLLCLK divided by 1 0001 = System PLLCLK divided by 2 ... 1111 = System PLLCLK divided by 16
Reserved Bits 9–2	Reserved—These bits are reserved and should read 0.	
SPEN Bit 1	**System PLL Enable**—Enables/Disables the System PLL. When software writes 0 to SPEN, the System PLL shuts down after SDCNT times out. SPEN sets automatically when SPLLEN asserts, and on system reset.	0 = System PLL disabled 1 = System PLL enabled
MPEN Bit 0	**MCU PLL Enable**—Enables/Disables the MCU PLL. When cleared, the MCU PLL is disabled. When software writes 0 to MPEN, the PLL shuts down immediately. MPEN sets automatically when MPLLEN asserts, and on system reset.	0 = MCU PLL disabled 1 = MCU PLL enabled

12.5.2 Peripheral Clock Divider Register

Each peripheral module in the MC9328MXL uses clock signals from one of the three clock sources shown in Table 12-6. The three peripheral clock dividers (PCLKDIV1, PCLKDIV2, PCLKDIV3) provide flexible clock configuration capability so that a minimum set of clock frequencies can satisfy a large group of modules to achieve better power efficiency.

Table 12-6. Clock Sources for Peripherals

Clock Source	Peripheral
PERCLK1	UART1, UART2, Timer1, Timer2, PWM
PERCLK2	LCD, SD, SPI 1, SPI 2
PERCLK3	SSI
HCLK	SDRAM, CSI, Memory Stick, I^2C, DMA

PCDR — Peripheral Clock Divider Register

Addr 0x0021B020

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
										PCLK_DIV3						
TYPE	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
	0x000B															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									PCLK_DIV2				PCLK_DIV1			
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	1
	0x00BB															

Table 12-7. Peripheral Clock Divider Register Description

Name	Description	Settings
Reserved Bits 31–23	Reserved—These bits are reserved and should read 0.	
PCLK_DIV3 Bits 22–16	**Peripheral Clock Divider 3**—Contains the 7-bit integer divider that produces the PERCLK3 clock signal for the peripherals. The input to the PCLK_DIV3 divider circuit is System PLLCLK.	0000000 = Divide by 1 0000001 = Divide by 2 … 1111111 = Divide by 128
Reserved Bits 15–8	Reserved—These bits are reserved and should read 0.	
PCLK_DIV2 Bits **7–**4	**Peripheral Clock Divider 2**—Contains the 4-bit integer divider that produces the PERCLK2 clock signal for the peripherals. The input to the PCLK_DIV2 divider circuit is System PLLCLK.	0000 = Divide by 1 0001 = Divide by 2 … 1111 = Divide by 16
PCLK_DIV1 Bits 3–0	**Peripheral Clock Divider 1**—Contains the 4-bit integer divider that produces the PERCLK1 clock signal for the peripherals. The input to the PCLK_DIV1 divider circuit is System PLLCLK.	0000 = Divide by 1 0001 = Divide by 2 … 1111 = Divide by 16

12.5.3 Programming Digital Phase Locked Loops

There are two DPLLs in the MC9328MXL—the MCU PLL and the System PLL. The MCU PLL primarily generates FCLK to the CPU, and the System PLL derives all system clocks to the entire MC9328MXL and generates clocks that produce the programmable frequency range required by modules such as the USB, UARTs, and SSI.

The MCU PLL derives the ARM920T processor's CPU clock FCLK, and the System PLL derives the ARM920T processor's CPU clock BCLK, as well as the system clocks PERCLK 1, 2, and 3, and HCLK. The MCU PLL frequency is determined by the speed requirement of the ARM920T processor. The recommended settings for both MCU PLL and System PLL, which produces the least amount of signal jitter, are shown in Table 12-8.

Table 12-8. Sample Frequency Table

Premultiplier Input Crystal Frequency	PLL Input Frequency (Premultiplier Output)	PD	MFD	MFI	MFN	PLL Output Frequency
32 kHz	16.384 MHz	0	63	5	55	192 MHz

12.5.3.1 MCU PLL Control Register 0

The MCU PLL Control Register 0 (MPCTL0) is a 32-bit register that controls the operation of the MCU PLL. The MPCTL0 control bits are described in the following sections. A delay of 56 FCLK cycles (about 10–30 FCLK cycles for MCU PLL controller plus 2–26 FCLK cycles are necessary to get EDRAM_IDLE and SDRAM_IDLE signals) is required between two write accesses to MPCTL0 register. The following is a procedure for changing the MCU PLL settings:

1. Program the desired values of PD, MFD, MFI, and MFN into the MPCTL0.
2. Set the MPLL_RESTART bit in the CSCR (it will self-clear).
3. New PLL settings will take place.

MPCTL0 MCU PLL Control Register 0 **Addr 0x0021B004**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			PD				MFD									
TYPE	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
								0x003F								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			MFI				MFN									
TYPE	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	1	0	1	0	0	0	0	1	1	0	1	1	1
								0x1437								

Table 12-9. MCU PLL Control Register 0 Description

Name	Description	Settings
Reserved Bits 31–30	Reserved—These bits are reserved and should read 0.	
PD Bits 29–26	**Predivider Factor**—Defines the predivider factor (PD) applied to the PLL input frequency. PD is an integer between 0 and 15 (inclusive). PD is chosen to ensure that the resulting output frequency remains within the specified range. When a new value is written into PD bits, the PLL loses its lock; after a time delay, the PLL re-locks. The output of the MCU PLL is determined by Equation 12-1.	0000 = 0 0001 = 1 … 1111 = 15
MFD Bits 25–16	**Multiplication Factor (Denominator Part)**—Defines the denominator part of the BRM value for the MF. When a new value is written into the MFD bits, the PLL loses its lock; after a time delay, the PLL re-locks.	0x000 = Reserved 0x001 = 1 … 0x3FF = 1023
Reserved Bits 15–14	Reserved—These bits are reserved and should read 0.	
MFI Bits 13–10	**Multiplication Factor (Integer)**—Defines the integer part of the BRM value for the MF. The MFI is encoded so that MFI < 5 results in MFI = 5. When a new value is written into the MFI bits, the PLL loses its lock: after a time delay, the PLL re-locks. The VCO oscillates at a frequency determined by Equation 12-1. Where PD is the division factor of the predivider, MFI is the integer part of the total MF, MFN is the numerator of the fractional part of the MF, and MFD is its denominator part. The MF is chosen to ensure that the resulting VCO output frequency remains within the specified range.	0000–0101 = 5 0110 = 6 ... 1111 = 15
MFN Bits 9–0	**Multiplication Factor (Numerator)**—Defines the numerator of the BRM value for the MF. When a new value is written into the MFN bits, the PLL loses its lock; after a time delay, the PLL re-locks.	0x000 = 0 0x001 = 1 ... 0x3FE = 1022 0x3FF = Reserved

12.5.3.2 MCU PLL and System Clock Control Register 1

The MCU PLL and System Clock Control Register 1 (MPCTL1) is a 32-bit register that directs the operation of the on-chip MCU PLL. The MPCTL1 control bits are described in Table 12-10.

MPCTL1 MCU PLL and System Clock Control Register 1 **Addr 0x0021B008**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										BRMO						
TYPE	r	r	r	r	r	r	r	r	r	rw	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 12-10. MCU PLL and System Clock Control Register 1 Description

Name	Description	Settings
Reserved Bits 31–7	Reserved—These bits are reserved and should read 0.	
BRMO Bit 6	**BRM Order**—Controls the BRM order. The first order BRM is used if a MF fractional part is both more than 1/10 and less than 9/10. In other cases, the second order BRM is used. The BRMO bit is cleared by a hardware reset. A delay of reference cycles is required between two write accesses to BRMO.	0 = BRM contains first order 1 = BRM contains second order
Reserved Bits 5–0	Reserved—These bits are reserved and should read 0.	

12.5.4 Generation of 48 MHz Clocks

The USB interface clock (CLK48M) is used internally by the USB module. Its frequency is set to 48 MHz using the PLL control registers assuming a default input clock frequency 16.384 MHz. This input clock frequency assumes a 32 kHz crystal input.

The predivider/multiplier output depends on the input clock frequency as shown in Table 12-11.

Table 12-11. System PLL Multiplier Factor

Premultiplier Input Crystal Frequency	PLL Input Frequency (Premultiplier Output)	PD	MFD	MFI	MFN	PLL Output Frequency	USBDIV	USB Clock Frequency
32 kHz	16.384 MHz	1	63	5	55	96 MHz	2	48 MHz
The default setting exception is USB_DIV. The user must program this to 001.								

12.5.4.1 System PLL Control Register 0

The System PLL Control Register 0 (SPCTL0) is a 32-bit register that controls the operation of the System PLL. The SPCTL0 control bits are described in the following sections. A delay of 30 System PLLCLK cycles is required between two write accesses to SPCTL0 register. The following is a procedure for changing the System PLL settings:

1. Program the desired values of PD, MFD, MFI, and MFN into the SPCTL0.
2. Set the SPLL_RESTART bit in the CSCR (it will self-clear).
3. New PLL settings will take place.

SPCTL0 System PLL Control Register 0 **Addr 0x0021B00C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			PD				MFD									
TYPE	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	1	0	0	0	0	1	1	1	1	1	1
								0x043F								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			MFI				MFN									
TYPE	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	1	0	1	0	0	0	0	1	1	0	1	1	1
								0x1437								

Table 12-12. System PLL Control Register 0 Description

Name	Description	Settings
Reserved Bits 31–30	Reserved—These bits are reserved and should read 0.	
PD Bits 29–26	**Predivider Factor**—Defines the predivider factor (PD) that is applied to the PLL input frequency. PD is an integer between 0 and 15 (inclusive). The System PLL oscillates at a frequency determined by Equation 12-1. The PD is chosen to ensure that the resulting VCO output frequency remains within the specified range. When a new value is written into the PD bits, the PLL loses its lock: after a time delay, the PLL re-locks.	0000 = 0 0001 = 1 … 1111 = 15

Table 12-12. System PLL Control Register 0 Description (continued)

Name	Description	Settings
MFD Bits 25–16	**Multiplication Factor (Denominator Part)**—Defines the denominator part of the BRM value for the MF. When a new value is written into the MFD9–MFD0 bits, the PLL loses its lock: after a time delay, the PLL re-locks.	0x000 = Reserved 0x001 = 1 … 0x3FF = 1023
Reserved Bits 15–14	Reserved—These bits are reserved and should read 0.	
MFI Bits 13–10	**Multiplication Factor (Integer Part)**—Defines the integer part of the BRM value for the MF. The MFI is decoded so that MFI < 5 results in MFI = 5. The System PLL oscillates at a frequency determined by Equation 12-1. Where PD is the division factor of the predivider, MFI is the integer part of the total MF, MFN is the numerator of the fractional part of the MF, and MFD is the denominator part of the MF. The MF is chosen to ensure that the resulting VCO output frequency remains within the specified range. When a new value is written into the MFI bits, the PLL loses its lock; after a time delay, the PLL re-locks.	0000–0101 = 5 0110 = 6 ... 1111 = 15
MFN Bits 9–0	**Multiplication Factor (Numerator Part)**—Defines the numerator part of the BRM value for the MF. When a new value is written into the MFN bits, the PLL loses its lock; after a time delay, the PLL re-locks.	0x000 = 0 0x001 = 1 ... 0x3FE = 1022 0x3FF = Reserved

12.5.4.2 System PLL Control Register 1

The System PLL control register 1 (SPCTL1) is a 32-bit read/write register in the MCU memory map that directs the operation of the System PLL. The SPCTL1 control bits are described in this section.

SPCTL1 System PLL Control Register 1 **Addr 0x0021B010**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
										0x0000						

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LF									BRMO						
TYPE	rw	r	r	r	r	r	r	r	r	rw	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
										0x0000						

Table 12-13. System PLL Control Register 1 Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
LF Bit 15	**Lock Flag**—Indicates whether the System PLL is locked. When set, the System PLL clock output is valid. When cleared, the System PLL clock output remains at logic high.	0 = System PLL is not locked 1 = System PLL is locked
Reserved Bits 14–7	Reserved—These bits are reserved and should read 0.	
BRMO Bit 6	**BRM Order Bit—**Controls the BRM order. The first order BRM is used if a MF fractional part is both more than 1/10 and less than 9/10. In other cases, the second order BRM is used. The BRMO bit is cleared by a hardware reset.	0 = BRM has first order 1 = BRM has second order
Reserved Bits 5–0	Reserved—These bits are reserved and should read 0.	

Chapter 13
DMA Controller

The Direct Memory Access Controller (DMAC) of the MC9328MXL provides eleven channels supporting linear memory, 2D memory, FIFO and end-of-burst enable FIFO transfers to provide support for a wide variety of DMA operations.

13.1 Features

The MC9328MXL DMAC features are:

- Eleven channels support linear memory, FIFO, and end-of-burst enable FIFO for both source and destination.
- Any one of the eleven channels can be configured to support 2D memory.
- Increment, decrement, and no-change support for source and destination addresses.
- Each channel is configurable to response to any of the 32 DMA request signals.
- Supports 8, 16, or 32-bit FIFO and memory port size data transfers.
- Supports both big and little endian.
- DMA is configurable to a maximum of 16 words, 32 half-words, or 64 bytes for each channel.
- Bus utilization control for the channel that is not trigger by a DMA request.
- Burst time-out errors terminate the DMA cycle when the burst cannot be completed within a programmed time count.
- Buffer overflow errors terminate the DMA cycle when the internal buffer receives more than 64 bytes of data. This is useful when the source mode is set to end-of-burst enable FIFO, in case the $\overline{\text{DMA_EOBI}}$ signal is not detected after 64 bytes of data are received.
- Transfer errors terminate the DMA cycle when a transfer error is detected during a DMA burst.
- DMA request time-out errors are generated for channels that are triggered by DMA requests to interrupt the CPU when a DMA request is not asserted after a programmed time count.
- Interrupts provided to the interrupt controller (and subsequently to the core) on bulk data transfer complete or transfer error.
- Each peripheral supporting DMA transfer generates a $\overline{\text{DMA_REQ}}$ signal to the DMA controller, assuming that each FIFO has a unique system address and generates a dedicated $\overline{\text{DMA_REQ}}$ signal to the DMA controller. For example, an USB device with 8 end-points has 8 DMA request signals to the DMA if they all support DMA transfer.
- The DMA controller provides an acknowledge signal to the peripheral after a DMA burst is complete. This signal is sometimes used by the peripheral to clear some status bits.
- Repeat data transfer function supports automatic USB host–USB device bulk/iso data stream transfer.

13.2 Block Diagram

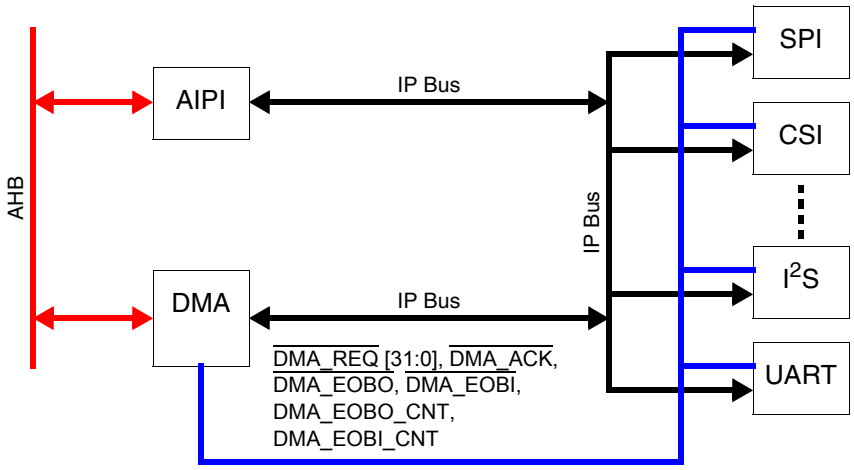


Figure 13-1. DMAC in MC9328MXL

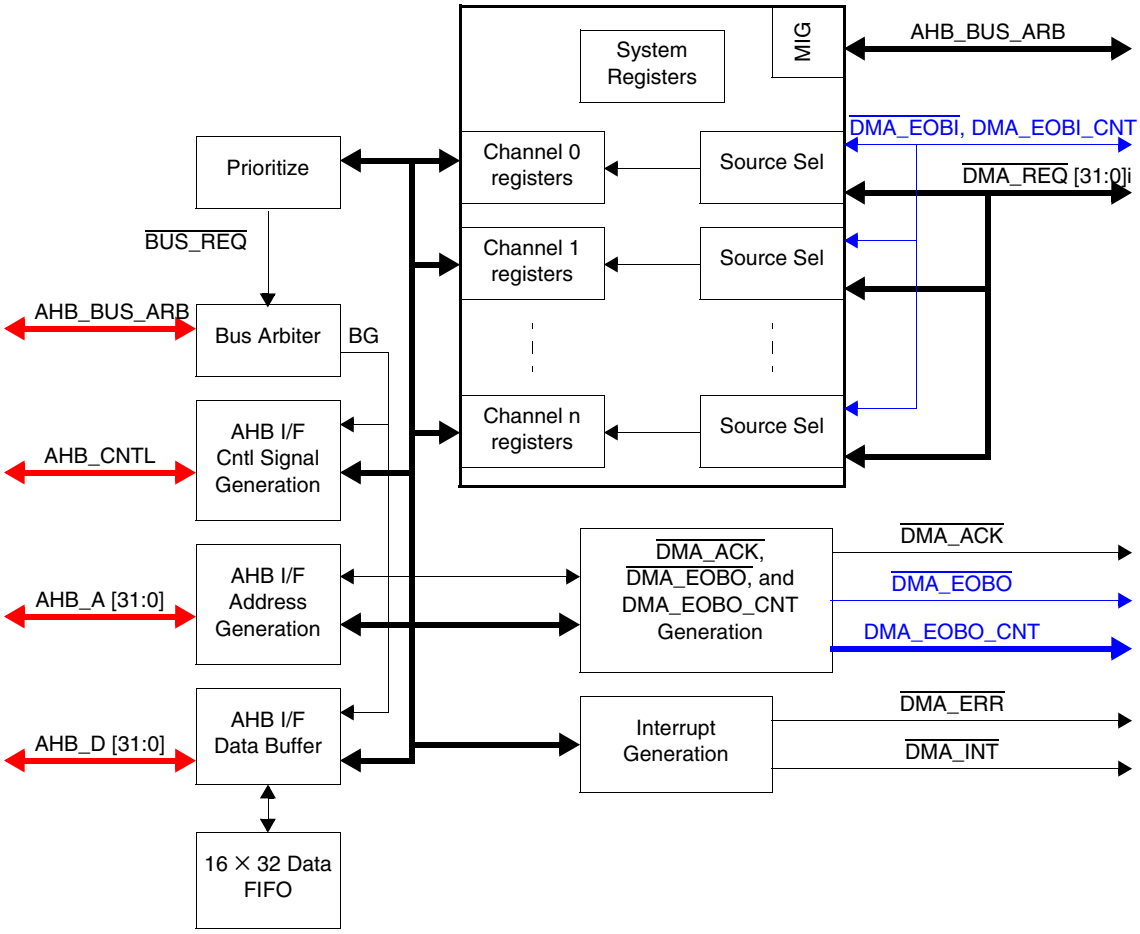


Figure 13-2. DMAC Block Diagram

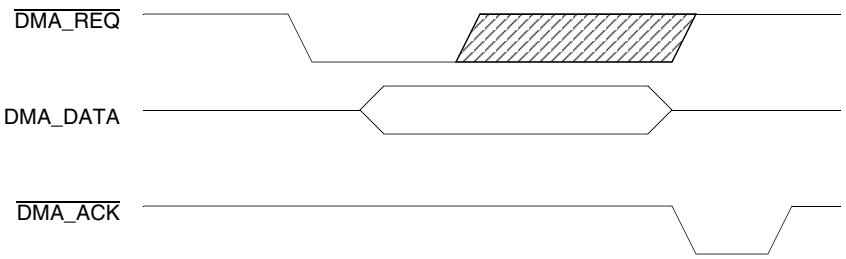


Figure 13-3. DMA Request and Acknowledge Timing Diagram

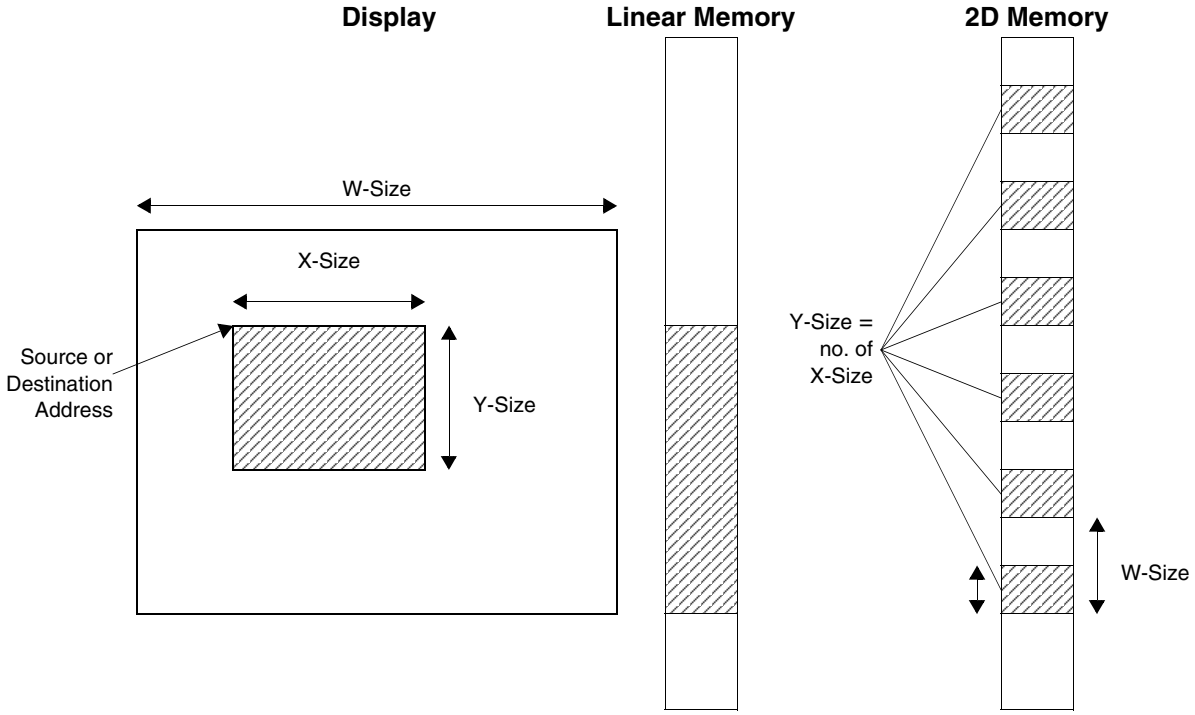


Figure 13-4. 2D Memory Diagram

13.3 Signal Description

The MC9328MXL signal descriptions are identified in Table 13-1.

Table 13-1. Signal Description

Signal	Description
AHB_xxx	AHB bus signals
IP Bus	IP bus signals
$\overline{\text{DMA_REQ}}$	DMA request signal generated by peripherals. One FIFO should generate one $\overline{\text{DMA_REQ}}$ signal. This signal must be negated by the peripheral automatically before the rising edge of $\overline{\text{DMA_ACK}}$. It is usually negated when the FIFO is read.

Table 13-1. Signal Description (continued)

Signal	Description
DMA_ACK	DMA request acknowledge generated by the DMA controller to signal the end of a DMA burst.
DMA_EOBI	This signal is asserted by the USB device when the last data of the burst is read from the FIFO.
DMA_EOBI_CNT	This signal is asserted by the USB device when the last data of the burst is read from the FIFO to indicate the number of valid bytes.
DMA_EOBO	This signal is asserted by the DMA controller when the last data of the burst is written to the FIFO.
DMA_EOBO_CNT	This signal is asserted by the DMA controller when the last data of the burst is written to the FIFO to indicate the number of valid bytes.
DMA_ERR	This signal is asserted by DMA controller when any DMA error is detected.
DMA_INT	This signal is asserted by DMA controller when data transfer is complete—that is, the data count reaches the desired level.

13.3.1 Big Endian and Little Endian

The BIG_ENDIAN signal determines the MC9328MXL memory endian configuration. BIG_ENDIAN is a static pin to the processor and if it is driven logic-high at power on reset the processor's memory system is configured as big endian. If it is driven logic-low at power on reset the processor's memory system is configured as little endian. The pin should not be changed after power on reset (POR) deasserts or during operation.

13.4 Programming Model

The DMA module includes 107 user-accessible 32-bit registers. These registers are divided into three groups by function:

- General registers for all functional blocks (see Section 13.4.1, on page 13-8)
- 2D memory registers to control the display width and the x and y of the window (see Section 13.4.2, on page 13-16)
- Channel registers to control and configure channels 0–10 (see Section 13.4.3, on page 13-18)

Table 13-2 summarizes these registers and their addresses.

Table 13-2. DMA Module Register Memory Map

Description	Name	Address
General Registers		
DMA Control Register	DCR	0x00209000
DMA Interrupt Status Register	DISR	0x00209004
DMA Interrupt Mask Register	DIMR	0x00209008

Table 13-2. DMA Module Register Memory Map (continued)

Description	Name	Address
DMA Burst Time-Out Status Register	DBTOSR	0x0020900C
DMA Request Time-Out Status Register	DRTOSR	0x00209010
DMA Transfer Error Status Register	DSESR	0x00209014
DMA Buffer Overflow Status Register	DBOSR	0x00209018
DMA Burst Time-Out Control Register	DBTOCR	0x0020901C
2D Memory Registers		
W-Size Register A	WSRA	0x00209040
X-Size Register A	XSRA	0x00209044
Y-Size Register A	YSRA	0x00209048
W-Size Register B	WSRB	0x0020904C
X-Size Register B	XSRB	0x00209050
Y-Size Register B	YSRB	0x00209054
Channel Registers		
Channel 0 Source Address Register Channel 0 Destination Address Register Channel 0 Count Register Channel 0 Control Register Channel 0 Request Source Select Register Channel 0 Burst Length Register Channel 0 Request Time-Out Register Channel 0 Bus Utilization Control Register	SAR0 DAR0 CNTR0 CCR0 RSSR0 BLR0 RTOR0 BUCR0	0x00209080 0x00209084 0x00209088 0x0020908C 0x00209090 0x00209094 0x00209098 0x00209098
Channel 1 Source Address Register Channel 1 Destination Address Register Channel 1 Count Register Channel 1 Control Register Channel 1 Request Source Select Register Channel 1 Burst Length Register Channel 1 Request Time-Out Register Channel 1 Bus Utilization Control Register	SAR1 DAR1 CNTR1 CCR1 RSSR1 BLR1 RTOR1 BUCR1	0x002090C0 0x002090C4 0x002090C8 0x002090CC 0x002090D0 0x002090D4 0x002090D8 0x002090D8
Channel 2 Source Address Register Channel 2 Destination Address Register Channel 2 Count Register Channel 2 Control Register Channel 2 Request Source Select Register Channel 2 Burst Length Register Channel 2 Request Time-Out Register Channel 2 Bus Utilization Control Register	SAR2 DAR2 CNTR2 CCR2 RSSR2 BLR2 RTOR2 BUCR2	0x00209100 0x00209104 0x00209108 0x0020910C 0x00209110 0x00209114 0x00209118 0x00209118

Table 13-2. DMA Module Register Memory Map (continued)

Description	Name	Address
Channel 3 Source Address Register	SAR3	0x00209140
Channel 3 Destination Address Register	DAR3	0x00209144
Channel 3 Count Register	CNTR3	0x00209148
Channel 3 Control Register	CCR3	0x0020914C
Channel 3 Request Source Select Register	RSSR3	0x00209150
Channel 3 Burst Length Register	BLR3	0x00209154
Channel 3 Request Time-Out Register	RTOR3	0x00209158
Channel 3 Bus Utilization Control Register	BUCR3	0x00209158
Channel 4 Source Address Register	SAR4	0x00209180
Channel 4 Destination Address Register	DAR4	0x00209184
Channel 4 Count Register	CNTR4	0x00209188
Channel 4 Control Register	CCR4	0x0020918C
Channel 4 Request Source Select Register	RSSR4	0x00209190
Channel 4 Burst Length Register	BLR4	0x00209194
Channel 4 Request Time-Out Register	RTOR4	0x00209198
Channel 4 Bus Utilization Control Register	BUCR4	0x00209198
Channel 5 Source Address Register	SAR5	0x002091C0
Channel 5 Destination Address Register	DAR5	0x002091C4
Channel 5 Count Register	CNTR5	0x002091C8
Channel 5 Control Register	CCR5	0x002091CC
Channel 5 Request Source Select Register	RSSR5	0x002091D0
Channel 5 Burst Length Register	BLR5	0x002091D4
Channel 5 Request Time-Out Register	RTOR5	0x002091D8
Channel 5 Bus Utilization Control Register	BUCR5	0x002091D8
Channel 6 Source Address Register	SAR6	0x00209200
Channel 6 Destination Address Register	DAR6	0x00209204
Channel 6 Count Register	CNTR6	0x00209208
Channel 6 Control Register	CCR6	0x0020920C
Channel 6 Request Source Select Register	RSSR6	0x00209210
Channel 6 Burst Length Register	BLR6	0x00209214
Channel 6 Request Time-Out Register	RTOR6	0x00209218
Channel 6 Bus Utilization Control Register	BUCR6	0x00209218
Channel 7 Source Address Register	SAR7	0x00209240
Channel 7 Destination Address Register	DAR7	0x00209244
Channel 7 Count Register	CNTR7	0x00209248
Channel 7 Control Register	CCR7	0x0020924C
Channel 7 Request Source Select Register	RSSR7	0x00209250
Channel 7 Burst Length Register	BLR7	0x00209254
Channel 7 Request Time-Out Register	RTOR7	0x00209258
Channel 7 Bus Utilization Control Register	BUCR7	0x00209258
Channel 8 Source Address Register	SAR8	0x00209280
Channel 8 Destination Address Register	DAR8	0x00209284
Channel 8 Count Register	CNTR8	0x00209288
Channel 8 Control Register	CCR8	0x0020928C
Channel 8 Request Source Select Register	RSSR8	0x00209290
Channel 8 Burst Length Register	BLR8	0x00209294
Channel 8 Request Time-Out Register	RTOR8	0x00209298
Channel 8 Bus Utilization Control Register	BUCR8	0x00209298

Table 13-2. DMA Module Register Memory Map (continued)

Description	Name	Address
Channel 9 Source Address Register	SAR9	0x002092C0
Channel 9 Destination Address Register	DAR9	0x002092C4
Channel 9 Count Register	CNTR9	0x002092C8
Channel 9 Control Register	CCR9	0x002092CC
Channel 9 Request Source Select Register	RSSR9	0x002092D0
Channel 9 Burst Length Register	BLR9	0x002092D4
Channel 9 Request Time-Out Register	RTOR9	0x002092D8
Channel 9 Bus Utilization Control Register	BUCR9	0x002092D8
Channel 10 Source Address Register	SAR10	0x00209300
Channel 10 Destination Address Register	DAR10	0x00209304
Channel 10 Count Register	CNTR10	0x00209308
Channel 10 Control Register	CCR10	0x0020930C
Channel 10 Request Source Select Register	RSSR10	0x00209310
Channel 10 Burst Length Register	BLR10	0x00209314
Channel 10 Request Time-Out Register	RTOR10	0x00209318
Channel 10 Bus Utilization Control Register	BUCR10	0x00209318

13.4.1 General Registers

This section describes the function of the general registers.

13.4.1.1 DMA Control Register

The DMA Control Register (DCR) controls the input of the system clock and the resetting of the DMA module.

DCR DMA Control Register **Addr 0x00209000**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															DRST	DEN
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	w	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 13-3. DMA Control Register Description

Name	Description	Settings
Reserved Bits 31–2	Reserved—These bits are reserved and should read 0.	
DRST Bit 1	**DMA Reset**—Generates a 3-cycle reset pulse that resets the entire DMA module, bringing the module to its reset condition. DRST always reads 0.	0 = No effect 1 = Generates a 3-cycle reset pulse
DEN Bit 0	**DMA Enable**—Enables/Disables the system clock to the DMA module.	0 = DMA disable 1 = DMA enable

13.4.1.2 DMA Interrupt Status Register

The DMA Interrupt Status Register (DISR) contains the interrupt status of each channel in the DMAC. The status bit is set whenever the corresponding DMA channel data transfer is complete. When any bit in the DMA Interrupt Status Register (DISR) is set and the corresponding bit in the interrupt mask register is cleared, a $\overline{\text{DMA_INT}}$ is asserted to the interrupt controller (AITC). When an interrupt occurs, the interrupt service routine must check the DISR to determine the interrupting channel. Each bit is cleared by writing 1 to it. The DISR bit cannot be cleared when the DMA channel is disabled.

DISR DMA Interrupt Status Register **Addr 0x00209004**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
TYPE	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 13-4. DMA Interrupt Status Register Description

Name	Description	Settings
Reserved Bits 31–11	Reserved—These bits are reserved and should read 0.	
CH10–CH0 Bits 10–0	**Channel 10 to 0 Interrupt Status**—Indicates the interrupt status for each DMA channel.	0 = No interrupt 1 = Interrupt is pending

13.4.1.3 DMA Interrupt Mask Register

The DMA Interrupt Mask Register (DIMR) masks both normal interrupts and error interrupts generated by the corresponding channel. There is one control bit for each channel. When an interrupt is masked, the interrupt controller does not generate an interrupt request to the AITC, however the status of the interrupt can be observed from the interrupt status register, burst time-out status register, request time-out status register, or the transfer error status register. At reset, all the interrupts are masked and all the bits in this register are set to 1.

DIMR DMA Interrupt Mask Register **Addr 0x00209008**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
TYPE	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
								0x07FF								

Table 13-5. DMA Interrupt Mask Register Description

Name	Description	Settings
Reserved Bits 31–11	Reserved—These bits are reserved and should read 0.	
CH10–CH0 Bits 10–0	**Channel 10 to 0**—Controls the interrupts for each DMA channel.	0 = Enables interrupts 1 = Disables interrupts

13.4.1.4 DMA Burst Time-Out Status Register

A burst time-out is set when a DMA burst cannot be completed within the number of clock cycles specified in the DMA Burst Time-Out Control Register (DBTOCR) of the channel. When any bit is set in this register and the corresponding bit in the interrupt mask register is cleared, a $\overline{\text{DMA_ERR}}$ is asserted to the interrupt controller (AITC). The DMA burst time-out status register (DBTOSR) indicates the channel, if any, that is currently being serviced and whether a burst time-out was detected. Each bit is cleared by writing 1 to it.

DBTOSR DMA Burst Time-Out Status Register **Addr 0x0020900C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
TYPE	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 13-6. DMA Burst Time-Out Status Register Description

Name	Description	Settings
Reserved Bits 31–11	Reserved—These bits are reserved and should read 0.	
CH10–CH0 Bits 10–0	**Channel 10 to 0**—Indicates the burst time-out status of each DMA channel.	0 = No burst time-out 1 = Burst time-out

13.4.1.5 DMA Request Time-Out Status Register

A DMA request time-out is set when there is no DMA request from the selected $\overline{\text{DMA_REQ}}$ source within the pre-assigned number of clock cycles specified in the request time-out control register (DBTOCR) for the channel. When any bit is set in this register and the corresponding bit in the interrupt mask register is cleared, a $\overline{\text{DMA_ERR}}$ is asserted to the AITC. The DMA Request Time-Out Status Register (DRTOSR) indicates the enabled channel, if any, that detected a DMA request time-out. Clear each bit by writing 1 to it.

DRTOSR DMA Request Time-Out Status Register **Addr 0x00209010**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
TYPE	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 13-7. DMA Request Time-Out Status Register Description

Name	Description	Settings
Reserved Bits 31–11	Reserved—These bits are reserved and should read 0.	
CH10–CH0 Bits 10–0	**Channel 10 to 0**—Indicates the request time-out status of each DMA channel.	0 = No DMA request time-out 1 = DMA request time-out

13.4.1.6 DMA Transfer Error Status Register

A DMA transfer error is set when the AHB bus signal HRESP [1:0] = ERROR is asserted during a DMA transfer. When any bit is set in this register and the corresponding bit in the interrupt mask register is cleared, a $\overline{\text{DMA_ERR}}$ is asserted to the AITC. The DMA Transfer Error Status Register (DSESR) indicates the channel, if any, detected a transfer error during a DMA burst. Clear each bit by writing 1 to it.

DSESR DMA Transfer Error Status Register **Addr 0x00209014**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
TYPE	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 13-8. DMA Transfer Error Status Register Description

Name	Description	Settings
Reserved Bits 31–11	Reserved—These bits are reserved and should read 0.	
CH10–CH0 Bits 10–0	**Channel 10 to 0**—Indicates the DMA transfer error status of each DMA channel.	0 = No transfer error 1 = Transfer error

13.4.1.7 DMA Buffer Overflow Status Register

The DMA Buffer Overflow Status Register (DBOSR) indicates whether the internal buffer of the DMA Controller overflowed during a data transfer. The channel is not enabled until the corresponding bit is cleared. When any bit is set in this register and the corresponding bit in the interrupt mask register is cleared, a $\overline{\text{DMA_ERR}}$ is asserted to the AITC.

DBOSR DMA Buffer Overflow Status Register **Addr 0x00209018**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
TYPE	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 13-9. DMA Buffer Overflow Status Register Description

Name	Description	Settings
Reserved Bits 31–11	Reserved—These bits are reserved and should read 0.	
CH10–CH0 Bits 10–0	**Channel 10 to 0**—Indicates the buffer overflow error status of each DMA channel.	0 = No buffer overflow occurred 1 = Buffer overflow occurred

13.4.1.8 DMA Burst Time-Out Control Register

This register sets the time-out for DMA transfer cycle for all DMA channels, so that the DMA controller can release the AHB and IP buses on error. An internal counter starts counting when a DMA burst cycle starts, and resets to zero when the burst is completed. When the counter reaches the count value set in the register, it asserts an interrupt and sets the corresponding error bit in the DMA burst time-out error register. The system clock is used as input clock to the counter.

DBTOCR DMA Burst Time-Out Control Register **Addr 0x0020901C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EN	CNT														
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 13-10. DMA Burst Time-Out Control Register Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
EN Bit 15	**Enable**—Enables/Disables the burst time-out.	0 = Disables burst time-out 1 = Enables burst time-out
CNT Bits 14–0	**Count**—Contains the time-out count down value.	

13.4.2 2D Memory Registers (A and B)

The two sets of 2D memory registers allow any one channel of the eleven channels to select any register set to define the respective 2D memory size.

13.4.2.1 W-Size Registers

The W-Size registers (WSRA and WSRB) define the number of bytes that make up the display width. This allows the DMA controller to calculate the next starting address of another row by adding the source/destination address to the contents of the W-Size register.

		Addr
WSRA	W-Size Register A	**0x00209040**
WSRB	W-Size Register B	**0x0020904C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	WS															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 13-11. W-Size Registers Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
WS Bits 15–0	**W-Size**—Contains the number of bytes that make up the display width.

13.4.2.2 X-Size Registers

The X-Size registers (XSRA and XSRB) contain the number of bytes per row of the window. The value of this register is used by the DMA controller to determine when to jump to the next row.

		Addr
XSRA	X-Size Register A	**0x00209044**
XSRB	X-Size Register B	**0x00209050**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	XS															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 13-12. X-Size Registers Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
XS Bits 15–0	**X-Size**—Contains the number of bytes per row that define the X-Size of the 2D memory.	

13.4.2.3 Y-Size Registers

The Y-Size registers (YSRA and YSRB) contain the number of rows in the 2D memory window. This setting is used by the DMA controller to calculate the total size of the transfer.

		Addr
YSRA	Y-Size Register A	**0x00209048**
YSRB	Y-Size Register B	**0x00209054**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	YS															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 13-13. Y-Size Registers Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
YS Bits 15–0	**Y-Size**—Contains the number of rows that make up the 2D memory window.

13.4.3 Channel Registers

Channels 0 to 10 support linear memory, 2D memory, FIFO, and end-of-burst enable FIFO transfer. Only one enabled channel may be configured for 2D memory at any time.

The interrupt request $\overline{\text{DMA_REQ}}$ [31:0] does not have a priority assigned. The only priority available is the priority that is defined for each channel: channel 10 has the highest priority and channel 0 has the lowest priority. Channel priority is implemented only when more than one request occurs at the same time, otherwise, channels are serviced on a first come, first serve basis.

Each channel generates a normal interrupt to the interrupt handler when the data count reaches the selected value and the channel source mode is not set to end-of-burst enable FIFO.

Each channel generates an error interrupt to the interrupt handler when the following conditions exist:

- A DMA request time-out is true
- A DMA burst time-out is true during a burst cycle
- The internal buffer overflows during a burst cycle
- A transfer error acknowledge is asserted during a burst cycle

13.4.3.1 Channel Source Address Register

Each of the channel source address registers contain the source address for the DMA cycle. The value of the register remains unchanged throughout the DMA process. If the memory direction bit (MDIR) in the channel control register (CCR) is clear (indicating a memory address increment), then the channel source address register contains the starting address of the memory block. If MDIR is set (indicating a memory address decrement), then the channel source address register contains the ending address of the memory block.

		Addr
SAR0	Channel 0 Source Address Register	**0x00209080**
SAR1	Channel 1 Source Address Register	**0x002090C0**
SAR2	Channel 2 Source Address Register	**0x00209100**
SAR3	Channel 3 Source Address Register	**0x00209140**
SAR4	Channel 4 Source Address Register	**0x00209180**
SAR5	Channel 5 Source Address Register	**0x002091C0**
SAR6	Channel 6 Source Address Register	**0x00209200**
SAR7	Channel 7 Source Address Register	**0x00209240**
SAR8	Channel 8 Source Address Register	**0x00209280**
SAR9	Channel 9 Source Address Register	**0x002092C0**
SAR10	Channel 10 Source Address Register	**0x00209300**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SA [31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SA [15:2]														SA [1]	SA [0]
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 13-14. Channel Source Address Register Description

Name	Description
SA [31:2] Bits 31–2	**Source Address**—Contains the source address from where data is read during a DMA transfer.
SA [1], SA [0] Bits 1–0	**Source Address [1] and Source Address [0]**—To ensure that all addresses are word-aligned these bits are set internally to 0. These bits will be read/write as any value if and only if running in big endian and source mode set to FIFO. This is to allow FIFO to use offset address during big endian mode.

13.4.3.2 Destination Address Registers

Each of the destination address registers (DARx) contain the destination address for a DMA cycle. The value of the register remains unchanged throughout the DMA process. If the memory direction bit (MDIR) in the channel control register (CCR) is clear (indicating a memory address increment), then the destination address register contains the starting address of the memory block. If MDIR is set (indicating a memory address decrement), then the destination address register contains the ending address of the memory block.

		Addr
DAR0	Channel 0 Destination Address Register	**0x00209084**
DAR1	Channel 1 Destination Address Register	**0x002090C4**
DAR2	Channel 2 Destination Address Register	**0x00209104**
DAR3	Channel 3 Destination Address Register	**0x00209144**
DAR4	Channel 4 Destination Address Register	**0x00209184**
DAR5	Channel 5 Destination Address Register	**0x002091C4**
DAR6	Channel 6 Destination Address Register	**0x00209204**
DAR7	Channel 7 Destination Address Register	**0x00209244**
DAR8	Channel 8 Destination Address Register	**0x00209284**
DAR9	Channel 9 Destination Address Register	**0x002092C4**
DAR10	Channel 10 Destination Address Register	**0x00209304**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DA [31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DA [15:2]														DA [1]	DA [0]
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 13-15. Channel Destination Address Registers Description

Name	Description
DA [31:2] Bits 31–2	**Destination Address**—Contains the destination address to which data is written to during a DMA transfer.
DA [1], DA [0] Bits 1–0	**Destination Address [1] and Destination Address [0]**—To ensure that all addresses are word-aligned, these bits are set internally to 0.

13.4.3.3 Channel Count Registers

Each of the channel count registers (CNTRx) contain the number of bytes of data to be transferred. There is an internal counter that counts up (number of bytes—4 for word, 2 for halfword and 1 for byte) for every DMA transfer. The internal counter is compared with the register after every transfer. When the counter value matches with the register value, the channel is disabled until the CEN bit is cleared and reset, or the RPT bit in the associated channel control register is set to 1. The internal counter is reset to 0 when the channel is enabled again.

The length of the last DMA burst can be shorter than the regular burst length specified in the burst length register. However, when data is transferred out from an I/O FIFO and the last burst is less than BL, the I/O device must generate a DMA request for the last transfer. When data is transferred to an I/O FIFO and the last burst is less than BL, only the remaining number of data is transferred.

When the source mode is set to end-of-burst enable FIFO, this register becomes a read only register and the value of the register is the number of bytes being transferred.

		Addr
CNTR0	Channel 0 Count Register	**0x00209088**
CNTR1	Channel 1 Count Register	**0x002090C8**
CNTR2	Channel 2 Count Register	**0x00209108**
CNTR3	Channel 3 Count Register	**0x00209148**
CNTR4	Channel 4 Count Register	**0x00209188**
CNTR5	Channel 5 Count Register	**0x002091C8**
CNTR6	Channel 6 Count Register	**0x00209208**
CNTR7	Channel 7 Count Register	**0x00209248**
CNTR8	Channel 8 Count Register	**0x00209288**
CNTR9	Channel 9 Count Register	**0x002092C8**
CNTR10	Channel 10 Count Register	**0x00209308**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
									CNT							
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CNT															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 13-16. Channel Count Registers Description

Name	Description
Reserved Bits 31–24	Reserved—These bits are reserved and should read 0.
CNT Bits 23–0	**Count**—Contains the number of bytes of data to be transferred during a DMA cycle.

13.4.3.4 Channel Control Registers

Each of the channel control registers (CCRx) controls and displays the status of a DMA channel operation.

NOTE:

While any one of the eleven channels may be configured for 2D memory, only one enabled channel may be configured for 2D memory at any time, This constraint does not apply to configuring the DMA channels for linear memory, FIFO, and end-of-burst enable FIFO.

		Addr
CCR0	Channel 0 Control Register	**0x0020908C**
CCR1	Channel 1 Control Register	**0x002090CC**
CCR2	Channel 2 Control Register	**0x0020910C**
CCR3	Channel 3 Control Register	**0x0020914C**
CCR4	Channel 4 Control Register	**0x0020918C**
CCR5	Channel 5 Control Register	**0x002091CC**
CCR6	Channel 6 Control Register	**0x0020920C**
CCR7	Channel 7 Control Register	**0x0020924C**
CCR8	Channel 8 Control Register	**0x0020928C**
CCR9	Channel 9 Control Register	**0x002092CC**
CCR10	Channel 10 Control Register	**0x0020930C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			DMOD		SMOD		MDIR	MSEL	DSIZ		SSIZ		REN	RPT	FRC	CEN
TYPE	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 13-17. Channel Control Registers Description

Name	Description	Settings
Reserved Bits 31–14	Reserved—These bits are reserved and should read 0.	
DMOD Bits 13–12	**Destination Mode**—Selects the destination transfer mode.	00 = Linear memory 01 = 2D memory 10 = FIFO 11 = End-of-burst enable FIFO

Table 13-17. Channel Control Registers Description (continued)

Name	Description	Settings
SMOD Bits 11–10	**Source Mode**—Selects the source transfer mode.	00 = Linear memory 01 = 2D memory 10 = FIFO 11 = End-of-burst enable FIFO
MDIR Bit 9	**Memory Direction**—Selects the memory address direction.	0 = Memory address increment 1 = Memory address decrement
MSEL Bit 8	**Memory Select**—Selects the 2D memory register set when either source and/or destination is programmed to 2D memory mode.	0 = 2D memory register set A selected 1 = 2D memory register set B selected
DSIZ Bits 7–6	**Destination Size**—Selects the destination size of a data transfer. **Note:** DSIZ1:DSIZ0 always reads/writes 00 when destination mode is programmed as end-of-burst enable FIFO, because end-of-burst operation only works for 32-bit FIFO.	00 = 32-bit destination port 01 = 8-bit destination port 10 = 16-bit destination port 11 = Reserved
SSIZ Bits 5–4	**Source Size**—Selects the source size of data transfer. **Note:** SSIZ1:SSIZ0 always reads/writes 00 when destination mode is programmed as end-of-burst enable FIFO, because end of burst operation only works for 32-bit FIFO.	00 = 32-bit source port 01 = 8-bit source port 10 = 16-bit source port 11 = Reserved
REN Bit 3	**Request Enable**—Enables/Disables the DMA request signal. When REN is set, the DMA burst is initiated by the $\overline{\text{DMA_REQ}}$ signal from the I/O FIFO. When REN is cleared, DMA transfer is initiated by CEN.	0 = Disables the DMA request signal (when the peripheral asserts a DMA request, no DMA transfer is triggered); DMA transfer is initiated by CEN only 1 = Enables the DMA request signal (when the peripheral asserts a DMA request, a DMA transfer is triggered)
RPT Bit 2	**Repeat**—Enables/Disables the data transfer repeat function. When enabled and when the counter reaches the value set in Count Register, the Count Register is reset to its zero, an interrupt is asserted, and the corresponding channel bit in the Interrupt Mask Register is cleared. The address is reloaded from the source and destination address register for the next DMA burst. Data transfer is carried out continuously until the channel is disabled or it completes the last cycle after RPT is cleared. If enabling the repeat function, do not change source and destination addresses on the fly. If it is necessary to change source and destination addresses, do it after a complete DMA cycle finishes and then re-start the channel again. **Note:** To correctly terminate a repeat enabled channel[x], user is required to first set RSSR[x] to 0, then set CCR[x]-REN to 1, and finally CCR[x]-CEN to 0.	0 = Disables repeat function 1 = Enables repeat function
FRC Bit 1	**Force a DMA Cycle**—Forces a DMA cycle to occur. FRC always reads 0.	0 = No effect 1 = Force DMA cycle

Table 13-17. Channel Control Registers Description (continued)

Name	Description	Settings
CEN Bit 0	**DMA Channel Enable**—Enables/Disables the DMA channel. **Note:** 1. Program all of the channel settings before enabling the channel. 2. To restart a channel, clear CEN, and then set CEN to 1.	0 = Disables the DMA channel 1 = Enables the DMA channel

When the source mode is set to end-of-burst enable FIFO, the burst length is determined by the input signals $\overline{\text{DMA_EOBI}}$ and DMA_EOBI_CNT, and the DMA burst (from peripheral to memory) can be terminated only by disabling the channel (clearing the corresponding CEN bit in channel control register). The count register (CNTR0-CNTR10) becomes read-only and indicates the number of bytes being transferred. This setting is typically used when the channel is configured to transfer data from an endpoint FIFO of a USB device to an endpoint data packet buffer in system memory.

When the destination mode is set to end-of-burst enable FIFO, the channel operates the same as in normal FIFO mode, the only difference is that at the end of each burst, the DMA controller generates a $\overline{\text{DMA_EOBO}}$ and DMA_EOBO_CNT signal to the peripheral. This setting is typically used when the I/O channel is configured to transfer data from an endpoint data packet buffer in system memory to an endpoint FIFO of a USB device.

Table 13-18. DMA_EOBO_CNT and DMA_EOBI_CNT Settings

DMA_EOBI_CNT [1:0] or DMA_EOBO_CNT [1:0]	Number of Bytes Per Transfer
00	4
01	1
10	2
11	3

13.4.3.5 Channel Request Source Select Registers

Each of the 32-bit channel request source select registers (RSSRx) selects one of the 32 DMA request signals ($\overline{\text{DMA_REQ}}$ [31:0]) to initiate a DMA transfer for the corresponding channel.

		Addr
RSSR0	Channel 0 Request Source Select Register	**0x00209090**
RSSR1	Channel 1 Request Source Select Register	**0x002090D0**
RSSR2	Channel 2 Request Source Select Register	**0x00209110**
RSSR3	Channel 3 Request Source Select Register	**0x00209150**
RSSR4	Channel 4 Request Source Select Register	**0x00209190**
RSSR5	Channel 5 Request Source Select Register	**0x002091D0**
RSSR6	Channel 6 Request Source Select Register	**0x00209210**
RSSR7	Channel 7 Request Source Select Register	**0x00209250**
RSSR8	Channel 8 Request Source Select Register	**0x00209290**
RSSR9	Channel 9 Request Source Select Register	**0x002092D0**
RSSR10	Channel 10 Request Source Select Register	**0x00209310**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												RSS				
TYPE	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 13-19. Channel Request Source Select Registers Description

Name	Description	Settings
Reserved Bits 31–5	Reserved—These bits are reserved and should read 0.	
RSS Bits 4–0	**Request Source Select**—Selects one of the 32 $\overline{\text{DMA_REQ}}$ signals that initiates a DMA transfer cycle for the channel.	00000 = select $\overline{\text{DMA_REQ}}$ [0] 00001 = select $\overline{\text{DMA_REQ}}$ [1] ... 11111 = select $\overline{\text{DMA_REQ}}$ [31]

13.4.3.6 Channel Burst Length Registers

The Channel Burst Length registers (BLRx) control the burst length of a DMA cycle. For a FIFO channel setting, the burst length is normally assigned according to the FIFO size of the selected I/O device, or by the FIFO level at which its $\overline{\text{DMA_REQ}}$ signal is asserted.

For example, when the UART RxD FIFO is 12 × 8 and it asserts $\overline{\text{DMA_REQ}}$ when it receives more than 8 bytes of data, BL is 8. When the memory port size also is 8-bit, the DMA burst is 8-byte reads followed by 8-byte writes.

When the memory port size is smaller than the I/O port size, the burst length of the byte writes is doubled. For example, the I/O port is 32-bit, the memory port is 16-bit, and the burst length is set to 32. In this configuration, the DMA performs 8 word burst reads and 16 halfword burst writes for I/O to memory transfer.

BLR0	Channel 0 Burst Length Register	**0x00209094**
BLR1	Channel 1 Burst Length Register	**0x002090D4**
BLR2	Channel 2 Burst Length Register	**0x00209114**
BLR3	Channel 3 Burst Length Register	**0x00209154**
BLR4	Channel 4 Burst Length Register	**0x00209194**
BLR5	Channel 5 Burst Length Register	**0x002091D4**
BLR6	Channel 6 Burst Length Register	**0x00209214**
BLR7	Channel 7 Burst Length Register	**0x00209254**
BLR8	Channel 8 Burst Length Register	**0x00209294**
BLR9	Channel 9 Burst Length Register	**0x002092D4**
BLR10	Channel 10 Burst Length Register	**0x00209314**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											BL					
TYPE	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 13-20. Channel Burst Length Registers Description

Name	Description	Settings
Reserved Bits 31–6	Reserved—These bits are reserved and should read 0.	
BL Bits 5–0	**Burst Length**—Contains the number of data bytes that are transferred in a DMA burst.	000000 = 64 bytes read follow 64 bytes write 000001 = 1byte read follow 1 byte write 000010 = 2 bytes read follow 2 bytes write .... 111111 = 63 bytes read follow 63 bytes write

13.4.3.7 Channel Request Time-Out Registers

The channel request time-out registers (RTOx) set the time-out for $\overline{\text{DMA_REQ}}$ from the selected request source of the channel, which detects any discontinuity of data transfer. The request time-out takes effect only when the corresponding request enable (REN) bit in the channel control register (CCR) is set. An internal counter starts counting when a DMA channel is enabled, the burst is completed, and the counter is reset to zero when a DMA request is detected. When the counter reaches the count value set in the register, it asserts an interrupt and sets its error bit in the DMA request time-out status register. The input clock of the counter is selectable from either the system clock (HCLK) or input crystal (CLK32K).

NOTE:

This register shares the same address as the bus utilization control register.

RTOR0	Channel 0 Request Time-Out Register	**0x00209098**
RTOR1	Channel 1 Request Time-Out Register	**0x002090D8**
RTOR2	Channel 2 Request Time-Out Register	**0x00209118**
RTOR3	Channel 3 Request Time-Out Register	**0x00209158**
RTOR4	Channel 4 Request Time-Out Register	**0x00209198**
RTOR5	Channel 5 Request Time-Out Register	**0x002091D8**
RTOR6	Channel 6 Request Time-Out Register	**0x00209218**
RTOR7	Channel 7 Request Time-Out Register	**0x00209258**
RTOR8	Channel 8 Request Time-Out Register	**0x00209298**
RTOR9	Channel 9 Request Time-Out Register	**0x002092D8**
RTOR10	Channel 10 Request Time-Out Register	**0x00209318**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EN	CLK	PSC	CNT												
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 13-21. Channel Request Time-Out Registers Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
EN Bit 15	**Enable**—Enables/Disables the DMA request time-out.	0 = Disables DMA request time-out 1 = Enables DMA request time-out

Table 13-21. Channel Request Time-Out Registers Description (continued)

Name	Description	Settings
CLK Bit 14	**Clock Source**—Selects the counter of input clock source.	0 = HCLK 1 = 32.768 kHz
PSC Bit 13	**Prescaler Count**—Sets the prescaler of the input clock.	0 = Divide by 1 1 = Divide by 256
CNT Bits 12–0	**Request Time-Out Count**—Contains the time-out count down value for the internal counter. This value remains unchanged through out the DMA process.	

13.4.3.8 Channel 0 Bus Utilization Control Register

The Bus Utilization Control register (BUCRx) controls the bus utilization of an enabled channel when the request enable (REN) bit in channel control register (CCR) is cleared. The channel does not request a DMA transfer until the counter reaches the count value set in the register except for the very first burst. This counter is cleared when the channel burst is started. When the count value is set to zero, the DMA carries on burst transfers one after another until it reaches the value set in count register. In this case, the user must be careful not to violate the maximum bus request latency of other devices.

NOTE:

This register shares the same address of request time-out register.

BUCR0	Channel 0 Bus Utilization Control Register	**0x00209098**
BUCR1	Channel 1 Bus Utilization Control Register	**0x002090D8**
BUCR2	Channel 2 Bus Utilization Control Register	**0x00209118**
BUCR3	Channel 3 Bus Utilization Control Register	**0x00209158**
BUCR4	Channel 4 Bus Utilization Control Register	**0x00209198**
BUCR5	Channel 5 Bus Utilization Control Register	**0x002091D8**
BUCR6	Channel 6 Bus Utilization Control Register	**0x00209218**
BUCR7	Channel 7 Bus Utilization Control Register	**0x00209258**
BUCR8	Channel 8 Bus Utilization Control Register	**0x00209298**
BUCR9	Channel 9 Bus Utilization Control Register	**0x002092D8**
BUCR10	Channel 10 Bus Utilization Control Register	**0x00209318**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CCNT															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 13-22. Channel 0 Bus Utilization Control Registers Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
CCNT Bits 15–0	**Clock Count**—Sets the number of system clocks that must occur before the memory channel releases the AHB, before the next DMA request for the channel.

13.5 DMA Request Table

Table 13-23 identifies the dedicated DMA request signal and its associated peripheral.

Table 13-23. DMA Request Table

DMA Request	Peripheral
$\overline{\text{DMA_REQ}}$ [31]	UART 1 Receive DMA Request
$\overline{\text{DMA_REQ}}$ [30]	UART 1 Transmit DMA Request
$\overline{\text{DMA_REQ}}$ [29]	UART 2 Receive DMA Request
$\overline{\text{DMA_REQ}}$ [28]	UART 2 Transmit DMA Request
$\overline{\text{DMA_REQ}}$ [27]	SPI 2 Transmit DMA RequestMSHC DMA Request
$\overline{\text{DMA_REQ}}$ [26]	SPI 2 Receive DMA RequestMSHC DMA Request
$\overline{\text{DMA_REQ}}$ [25]	USB Device End Point 5 DMA Request
$\overline{\text{DMA_REQ}}$ [24]	USB Device End Point 4 DMA Request
$\overline{\text{DMA_REQ}}$ [23]	USB Device End Point 3 DMA Request
$\overline{\text{DMA_REQ}}$ [22]	USB Device End Point 2 DMA Request
$\overline{\text{DMA_REQ}}$ [21]	USB Device End Point 1 DMA Request
$\overline{\text{DMA_REQ}}$ [20]	USB Device End Point 0 DMA Request
$\overline{\text{DMA_REQ}}$ [19]	Reserved
$\overline{\text{DMA_REQ}}$ [18]	Reserved
$\overline{\text{DMA_REQ}}$ [17]	SSI Receive DMA Request
$\overline{\text{DMA_REQ}}$ [16]	SSI Transmit DMA Request
$\overline{\text{DMA_REQ}}$ [15]	SPI 1 Transmit DMA Request
$\overline{\text{DMA_REQ}}$ [14]	SPI 1 Receive DMA Request
$\overline{\text{DMA_REQ}}$ [13]	SDHC DMA Request
$\overline{\text{DMA_REQ}}$ [12]	Ext_DMA
$\overline{\text{DMA_REQ}}$ [11]	DSPA MAC DMA Request
$\overline{\text{DMA_REQ}}$ [10]	DSPA DCT DIN DMA Request
$\overline{\text{DMA_REQ}}$ [9]	DSPA DCT DOUT DMA Request
$\overline{\text{DMA_REQ}}$ [8]	MSHC DMA Request
$\overline{\text{DMA_REQ}}$ [7]	CSI Receive FIFO DMA Request
$\overline{\text{DMA_REQ}}$ [6]	CSI Statistic FIFO DMA Request
$\overline{\text{DMA_REQ}}$ [5]	Reserved
$\overline{\text{DMA_REQ}}$ [4]	Reserved
$\overline{\text{DMA_REQ}}$ [3]	Reserved
$\overline{\text{DMA_REQ}}$ [2]	Reserved
$\overline{\text{DMA_REQ}}$ [1]	Reserved
$\overline{\text{DMA_REQ}}$ [0]	Reserved

Chapter 14
Watchdog Timer Module

14.1 General Overview

The watchdog timer module of the MC9328MXL protects against system failures by providing a method of escaping from unexpected events or programming errors. Once activated, the timer must be serviced by software on a periodic basis. If servicing does not take place, the timer times out. Upon a time-out, the watchdog timer module either asserts a system reset signal $\overline{\text{WDT_RST}}$ or a interrupt request signal $\overline{\text{WDT_INT}}$ depending on software configuration. Table 14-1 on page 14-5 shows the watchdog timer module's input and output signals. A state machine that demonstrates the time-out operation of the counter operation is shown in Figure 14-2 on page 14-4.

14.2 Watchdog Timer Operation

The following sections describe the operation and programming of the watchdog timer module.

14.2.1 Timing Specifications

The watchdog timer provides time-out periods from 0.5 seconds up to 64 seconds with a time resolution of 0.5 seconds. As shown in Figure 14-1, the watchdog timer uses the CLK2HZ clock (from RTC module) as an input to achieve the resolution of 0.5 seconds and a frequency of 2 Hz. This clock is connected to the input of a 7-bit counter to obtain a range of 0.5 to 64 seconds. The user can determine the time-out period by writing to the watchdog time-out field (WT[6:0]) in the Watchdog Control Register (WCR).

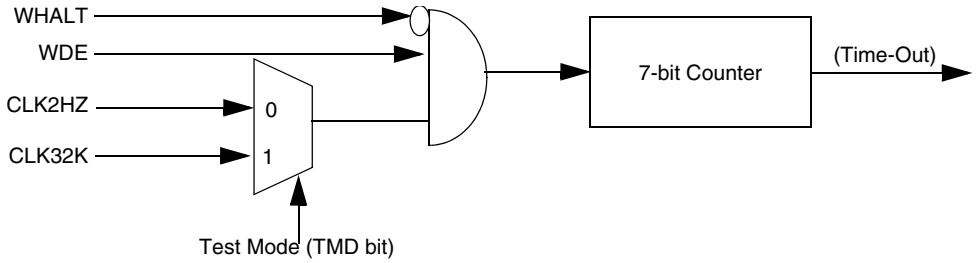


Figure 14-1. Watchdog Timer Functional Block Diagram

14.2.2 Watchdog During Reset

14.2.2.1 Power-On Reset

During a power-on reset (POR) all registers are reset to their reset values and the counter is placed in the idle state until the watchdog is enabled. The Watchdog Status Register (WSTR) contains the source of the reset event and the interrupt status bit TINT is reset to 0.

14.2.2.2 Software Reset

When software reset occurs, the software reset (SWR) bit in Watchdog Control Register (WCR) is set to 1, all registers of the Watchdog module are reset to their reset values and the counter is placed in the idle state until the watchdog is enabled.

14.3 Watchdog After Reset

After reset, watchdog timer operation can be divided into four states: initial load, countdown, reload, and time-out. The following sections define each of the watchdog timer states after reset.

14.3.1 Initial Load

The Watchdog Control Register (WCR) bits WT[6:0] must be written to before the watchdog is enabled. The watchdog is then enabled by setting the one-time writable watchdog enable (WDE) bit in the WCR. The time-out value is loaded into the counter after the service sequence is written to the Watchdog Service Register (WSR) or after the watchdog is enabled. The service sequence is described in Section 14.3.3, “Reload.” The counter state machine is shown in Figure 14-2 on page 14-4.

14.3.2 Countdown

The counter is activated after the Watchdog is enabled and begins to count down from its initial programmed value. If any system errors have occurred which prevents the software from servicing the Watchdog Service Register (WSR), the timer will time-out when the counter reaches zero. If the WSR is serviced prior to the counter reaching zero, the watchdog reloads its counter to the time-out value indicated by bits WT[6:0] of the WCR and re-start the countdown. A reset will reset the counter and place it in the idle state at any time during the countdown. The counter state machine is shown in Figure 14-2 on page 14-4.

14.3.3 Reload

The recommended service sequence is to write a $5555 followed by a $AAAA to the WSR. To reload the counter, the writes must take place within the time-out value indicated by bits WT[6:0] of the WCR. Any number of instructions can be executed between the two writes. This service sequence is also used to activate the counter during the initial load. See Section 14.3.1, “Initial Load.”

If the WSR is not loaded with a $5555 prior to a write of $AAAA to the WSR, the counter will not be reloaded. If any value other than $AAAA is written to the WSR after $5555, the counter will not be reloaded.

14.3.4 Time-Out

If the counter reaches zero, the TOUT bit in WSTR (Watchdog Status Register) is set to 1 indicating that watchdog has timed out. Reading the TOUT bit will clear it.

If the counter reaches zero, the watchdog asserts either a system reset signal $\overline{\text{WDT_RST}}$ or an interrupt request signal $\overline{\text{WDT_INT}}$ depending on the state of the WIE bit in the WCR. A 1 written to WIE configures the watchdog to generate a interrupt request signal to the interrupt handler. When a watchdog time-out interrupt is asserted, the TINT bit in WSTR (Watchdog Status Register) is set to 1 to indicate that an interrupt request is generated and the reading of this bit clears the interrupt and this bit. A 0 written to the WIE bit configures the watchdog to generate a $\overline{\text{WDT_RST}}$ signal to reset the module. The counter state machine is shown in Figure 14-2 on page 14-4.

14.3.5 Halting the Counter

The watchdog counting can be halted at any time by setting the WHALT bit (WCR[15]) to 1. The counter immediately stops counting and the counter value is held at the last value. The WHALT bit can be cleared by writing 0 to it or it can be automatically cleared by the occurrence of any of three system events, fast interrupt, slow interrupt, or system reset. The counter resumes counting from the stopped value. No other configurations are affected.

14.4 Watchdog Control

14.4.1 Interrupt Control

The watchdog timer generates interrupt request signal $\overline{\text{WDT_INT}}$ as a result of a WDOG time-out when WIE bit of WCR set to 1. The TINT bit of WSTR is set to 1 to indicate that the interrupt request has been generated. Reading the TINT bit clears the interrupt and this status bit.

14.4.2 Reset Sources

The watchdog timer generates reset signal $\overline{\text{WDT_RST}}$ as a result of a WDOG time-out. This signal is an output to the Reset Module for system reset generation.

14.5 State Machine

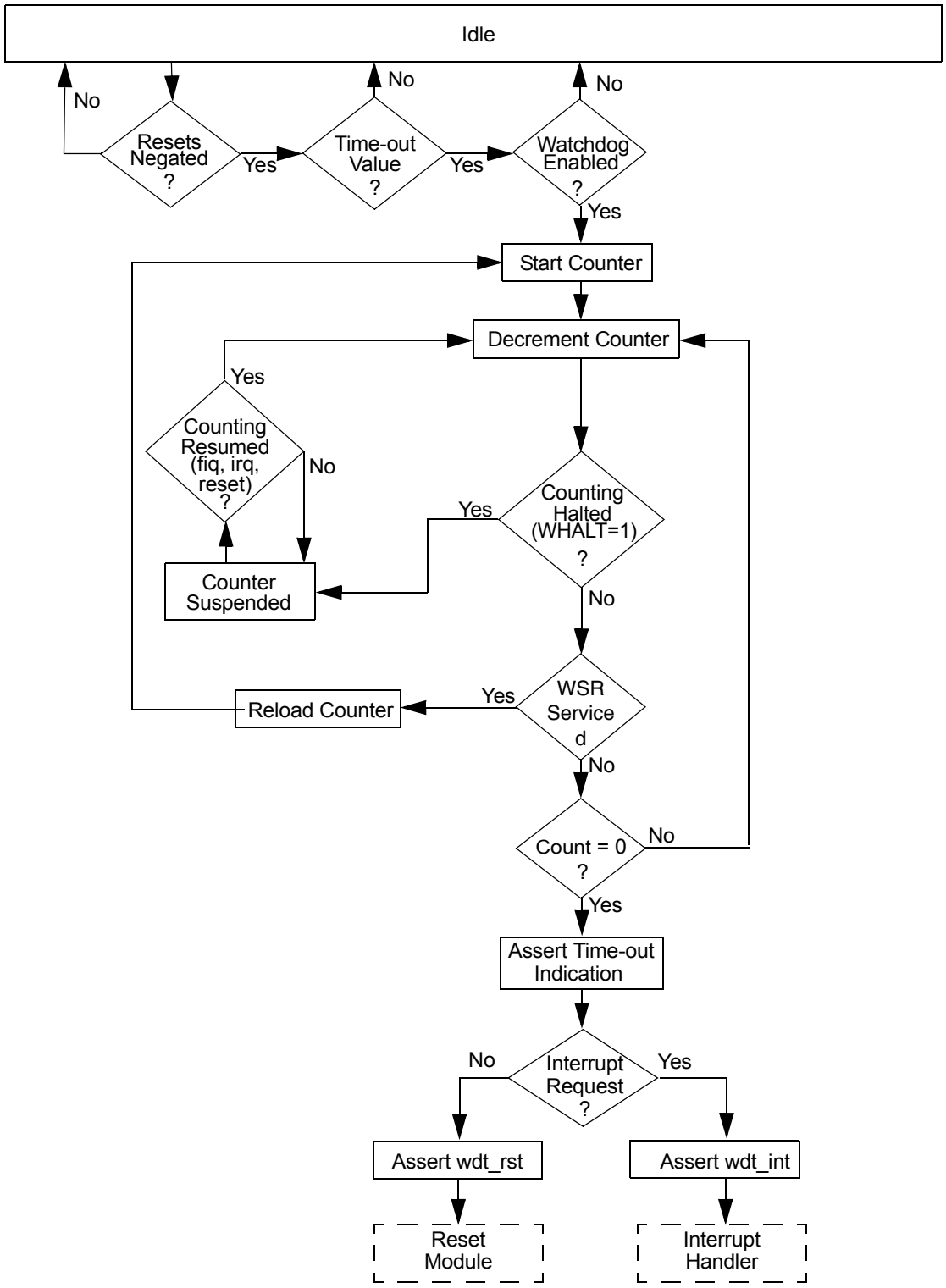


Figure 14-2. Counter State Machine

14.6 Watchdog Timer I/O Signals

Table 14-1shows the watchdog timer module input and output signals.

Table 14-1. Watchdog Timer I/O Signals

Signal Name	I/O	Description
$\overline{\text{FIQ}}$	I	Fast Interrupt
$\overline{\text{IRQ}}$	I	Normal Interrupt
$\overline{\text{IPS_HARD_ASYNC_RESET}}$	I	WDOG global reset from reset module
IPS_CONT_CLK	I	96 MHz system clock
$\overline{\text{IPS_CONT_CLK}}$	I	96 MHz system clock inverted
IPS_GATED_CLK	I	Bus clock
$\overline{\text{IPS_GATED_CLK}}$	I	Bus clock inverted
CLK2HZ	I	2 Hz clock input from RTC module output
CLK32K	I	in test mode, counter clock becomes 32 kHz clock
IPS_MODULE_EN	I	Watchdog module enable
IPS_BYTE_15_8	I	Bit 15 to 8 enable
IPS_BYTE_7_0	I	Bit 7 to 0 enable
IPS_MRW	I	Module read/write signal
IPS_ADDR[11:2]	I	Module address bus
IPS_WDATA[31:0]	I	Module write data bus
SCAN_MODE	I	Indicates scan mode selection
$\overline{\text{SCAN_RESET}}$	I	Indicates scan reset
IPS_CONT_CLK_EN	O	ips_cont_clk enable
IPS_XFR_ERR	O	Transfer error acknowledge
IPS_XFR_WAIT	O	Transfer wait acknowledge
IPS_RDATA[31:0]	O	Module read data bus

14.7 Programming Model

The watchdog timer has three registers in its programming model: Watchdog Control Register (WCR), Watchdog Service Register (WSR), and Watchdog Status Register (WSTR).

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WCR ($00201000)	R	WHALT	WT[6:0]							0	0	0	WIE	TMD	SWR	WDEC	WDE
	W																
WSR ($00201004)	R	WSR[15:0]															
	W																
WSTR ($00201008)	R	0	0	0	0	0	0	0	TINT	0	0	0	0	0	0	0	TOUT
	W																

14.7.1 Watchdog Control Register

The WCR is a 32-bit read/write (byte writable) register. It controls the Watchdog operation. See Table 14-2 on page 14-6 for bit descriptions and settings.

WCR Watchdog Control Register **Addr 0x00201000**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	WHALT	WT										WIE	TMD	SWR	WDEC	WDE
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 14-2. Watchdog Control Register Description

Name	Description	Settings
Reserved Bits 31–15	Reserved—These bits are reserved and should read 0.	
WHALT Bit 15	**Watchdog Halt**—When set, the watchdog counter immediately stops counting and the counter value is held at the last value. The WHALT bit can be cleared by writing 0 to it or it can be automatically cleared by the occurrence of any of three system events, fast interrupt, slow interrupt, or system reset.	0 = Counter is not halted 1 = Counter is halted

Table 14-2. Watchdog Control Register Description (continued)

Name	Description	Settings
WT Bits 14–8	**Watchdog Time-Out Field**—This 7-bit field contains the time-out value and is loaded into the Watchdog counter after the service routine has been performed. After reset, WT[6:0] must be written before enabling the Watchdog.	Set to desired time-out value.
Reserved Bits 7–5	Reserved—These bits are reserved and should read 0.	
WIE Bit 4	**Watchdog Interrupt Enable**—Determines if the $\overline{\text{WDT_RST}}$ is asserted or $\overline{\text{WDT_INT}}$ is asserted upon a watchdog time-out.	1 = Assert $\overline{\text{WDT_INT}}$ 0 = Assert $\overline{\text{WDT_RST}}$
TMD Bit 3	**Test Mode Enable**—Determines if WDOG timer is in test mode. **Note:** This bit is used only for test purposes	0 = Use 2 Hz clock as counter clock 1 = Use CLK32K as counter clock
SWR Bit 2	**Software Reset Enable**—Determines if a software reset is enabled.	0 = Software reset is not enabled 1 = Software reset is enabled
WDEC Bit 1	**Watchdog Enable Control**—Controls the write access of the WDE bit.	0 = WDE bit is write once only 1 = WDE bit is write multiple
WDE Bit 0	**Watchdog Enable**—Enables or disables the watchdog module. Write once-only if WDEC bit is low. Write multiple if WDEC bit is high	0 = Disable Watchdog 1 = Enable Watchdog

14.7.2 Watchdog Service Register

The Watchdog Service register contains the watchdog service sequence. When Watchdog is enabled, the Watchdog requires that a service sequence be written to the Watchdog Service Register (WSR) as described in Table 14-3.

WSR Watchdog Service Register **Addr 0x00201004**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	WSR															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 14-3. Watchdog Service Register Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	

Table 14-3. Watchdog Service Register Description (continued)

Name	Description	Settings
WSR Bits 15–0	**Watchdog Service Register**—This 15-bit field contains the watchdog service sequence. Both writes must occur in the order listed prior to the time-out, however any number of instructions can be executed between the two writes.	The service sequence must be performed as follows: a) Write $5555 to the Watchdog Service Register (WSR). b) Write $AAAA to the Watchdog Service Register (WSR)

14.7.3 Watchdog Status Register

The WSTR is a read-only register which records the source of the RESET_OUT event and interrupt status. It is cleared by reset. It records the source of the RESET_OUT event and interrupt status. RESET_OUT can be generated by the following sources which are listed in priority from highest to lowest: Power-on reset, External reset (RESET_IN), and Watchdog Time-out.

WSTR Watchdog Status Register **Addr 0x00201008**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								TINT								TOUT
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 14-4. Watchdog Status Register Description

Name	Description	Settings
Reserved Bits 31–9	Reserved—These bits are reserved and should read 0.	
TINT Bit 8	**Time-Out Interrupt**—Indicates whether the time-out interrupt generated	0 = No time-out interrupt generated 1 = Time-out interrupt generated
Reserved Bits 7–1	Reserved—These bits are reserved and should read 0.	
TOUT Bit 0	**Time-Out**—Indicates whether the watchdog timer times out.	0 = Watchdog timer does not time-out. 1 = Watchdog timer times out.

Chapter 15
Multimedia Accelerator (MMA)

15.1 Introduction

Many digital signal processing algorithms require iterative operations that can be closely pipelined, however they require irregular addressing for data access. These algorithms include FIR filtering, correlation, and FFT operations. In many system implementations, these operations account for a large percentage of the total processing cycles.

The multimedia accelerator (MMA) provides the with digital signal processing capability while maintaining efficient utilization of system and bus resources. The MMA in conjunction with the ARM9 processor core (ARM920T processor), form a hybrid operating environment that combines the efficiency and simplicity of a RISC processor with the powerful, number crunching, iterative operations of a digital signal processor. The RISC processor implements the algorithms and processes, assisted by the MMA in crucial digital signal processing operations. Applications include MPEG or MP3 encoding/decoding and speech compression/decompression such as G.723.1, CELP, or RPE-LTP for GSM.

15.2 MMA Operation

The MMA module consists of two major blocks—a multiply-accumulate (MAC) block and a discrete cosine transform (DCT) block. Each of these blocks has its own set of control registers. The control registers are accessed by the ARM920T processor for configuration as well as data input and result access.

Output data is stored in the internal FIFO of the MMA. If the FIFO is not cleared, MMA processes halt so that no output data is overwritten or lost.

15.2.1 Memory Access

The MMA supports only 32–bit access to its registers because the bus interface to the system bus, referred to as the Advanced High-performance Bus (AHB), is 32 bits wide. Because the MMA processes data that is 24 bits wide, access to memory is always in 32–bit words. The MMA supports both big endian and little endian access.

Figure 15-1 on page 15-3 shows the data access to the by the MMA and the ARM920T processor.

15.2.2 MAC

The MAC block provides the with fast multiply-accumulate capability. It can perform 1-D × 1-D, 1-D × 2-D, 2-D × 1-D and 2-D × 2-D matrix multiplication to support applications such as MPEG audio encoder subband filtering, decoder subband synthesis, and MP3 IMDCT.

15.2.2.1 Basic MAC Operation

Two circular data addressing units in the MMA provide the control to fetch data for two operands. All memory access is in 32–bit words. The MAC can perform 24–bit × 24–bit signed, unsigned, or alternating sign multiplication. The 48–bit multiplier output is added to a 56–bit accumulator, allowing for 8–bit overflow. After a user-defined number of MAC iterations, the accumulator value is stored in a 32 × 32–bit FIFO and the accumulator is cleared. The user can select which 32–bit subset of the 56–bit accumulator result is stored in the FIFO.

15.2.2.2 Data Access

The two operands for the multiplier are supplied by the X and Y registers. The data for these two registers is loaded from memory by the data access controller. The MMA maintains two circular buffers in the, one each for the X and Y operands.

To limit how long the bus is held when the MMA accesses memory, the MMA_MAC_BURST register sets the number of burst cycles permitted for each access, after which the is released. The MMA resumes operation if there are no other access requests pending.

Circular buffer operation for the X registers is shown in Figure 15-1 on page 15-3.

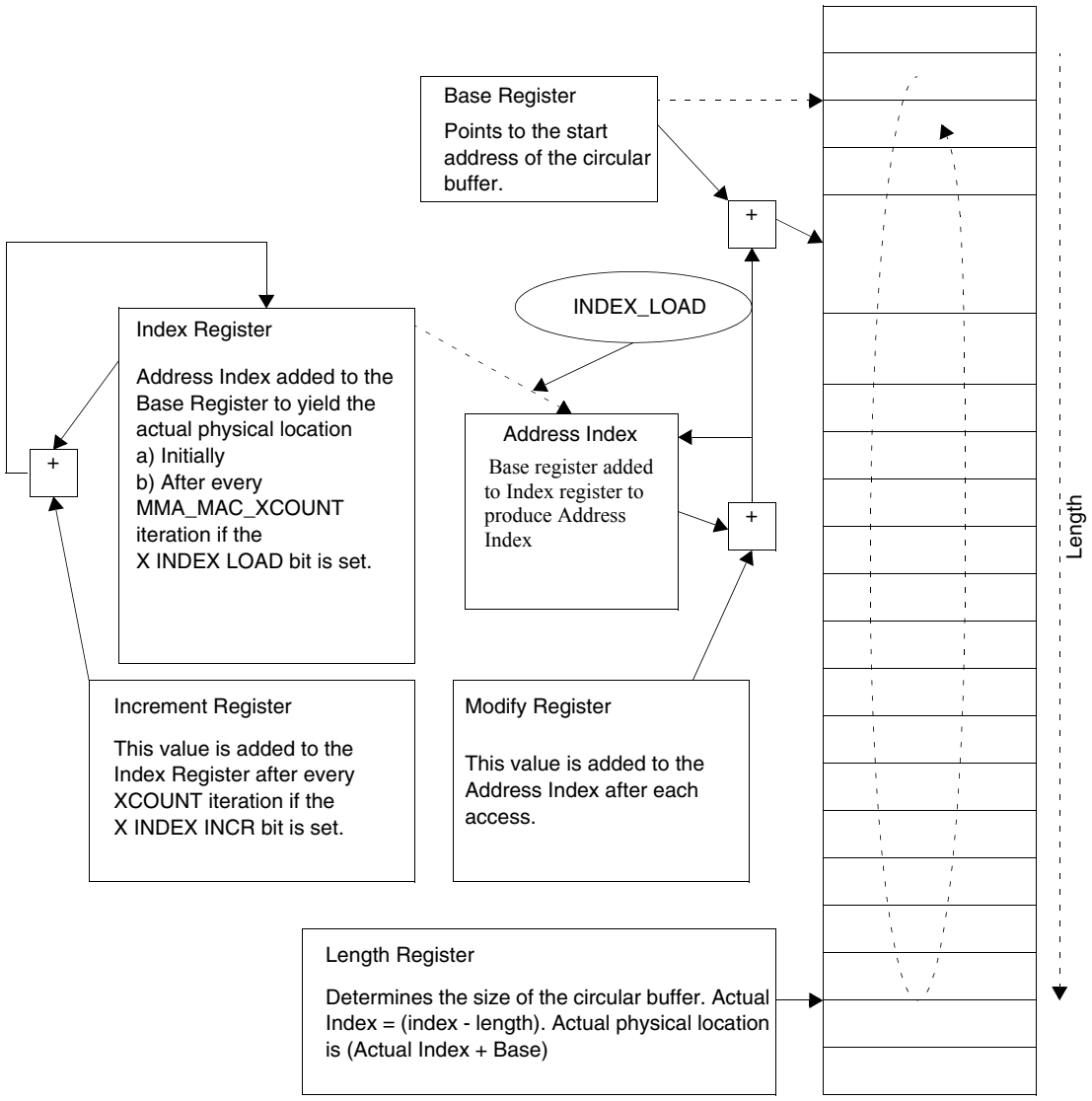


Figure 15-1. Circular Buffering Operation

15.2.2.3 Cache

The X operand access has an associated cache and cache controller. Initially, the cache is cleared and the X operand data is accessed from the and stored in the cache. Subsequent accesses to the same address cause a cache hit and the data is accessed from the cache.

The cache is a memory block of 512 24–bit words. Each word also has an associated valid bit to indicate data validity. The cache can be enabled or disabled. To fully use the cache, the base address of the operand must be on a 2K boundary. When a data access matches an address in the cache and the valid bit for that word is set, data is fetched from the cache. When the valid bit is cleared, data is accessed from the or from external memory and stored in the cache, and the valid bit is set. The cache is cleared only by writing 1 to the CACHE CLR bit. This action also registers the base address of the 2K boundary as the valid cache block address. The user must program the MMA_MAC_XBASE register and the MMA_MAC_XINDEX register before clearing the cache.

15.2.3 DCT/iDCT

The DCT/iDCT block in the MMA performs 2-D 8 × 8 discrete cosine transforms and inverse discrete cosine transforms on 8 × 8 blocks of pixel data. The design is based on a distributed arithmetic processor that computes two bits at a time. Latency is approximately 170 clock cycles after filling the input FIFO.

Figure 15-2 is a block diagram of the DCT/iDCT.

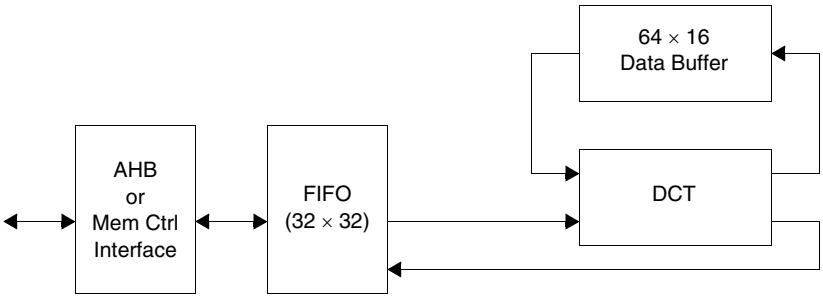


Figure 15-2. DCT/iDCT Architecture

The DCT/iDCT can be programmed to access data through the AHB bus or through the memory controller. When the memory controller is used, addresses are generated automatically—the user programs the start and destination addresses, the number of blocks in the X-direction (XCOUNT) and Y-direction (YCOUNT), and the address offsets. The source address and destination address can be same.

Input data is loaded into a 32 × 32 FIFO. Each word in the FIFO represents two 16–bit pixels. The accuracy of the input data is 9 bits for a DCT, so the 7 least significant bits (LSBs) must be zero-filled. For an iDCT, the accuracy is 12 bits, so the 4 LSBs must be zero-filled.

DCT Format

D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0	0	0

iDCT Format

D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0

Figure 15-3. Data Formatting for DCT and iDCT

The DCT/iDCT is enabled by writing 1 to the DCT ENA bit in the MMA_DCTCTRL register. After the bit is set, the DCT/iDCT is performed automatically. When the process is complete, an interrupt is generated and the DCT ENA bit is cleared. In this way, a DCT/iDCT can be run for an entire frame of data.

15.3 Programming Model

The MMA module includes 33 user-accessible 32-bit registers. Table 15-1 on page 15-5 summarizes these registers and their addresses.

Table 15-1. MMA Module Register Memory Map

Description	Name	Address
MMA MAC Control Registers		
MMA MAC Module Register	MMA_MAC_MOD	0x00222000
MMA MAC Control Register	MMA_MAC_CTRL	0x00222004
MMA MAC Multiply Counter Register	MMA_MAC_MULT	0x00222008
MMA MAC Accumulate Counter Register	MMA_MAC_ACCU	0x0022200C
MMA MAC Interrupt Register	MMA_MAC_INTR	0x00222010
MMA MAC Interrupt Mask Register	MMA_MAC_INTR_MASK	0x00222014
MMA MAC FIFO Register	MMA_MAC_FIFO	0x00222018
MMA MAC FIFO Status Register	MMA_MAC_FIFO_STAT	0x0022201C
MMA MAC Burst Count Register	MMA_MAC_BURST	0x00222020
MMA MAC Bit Select Register	MMA_MAC_BITSEL	0x00222024
MMA MAC X Register Control Registers		
MMA MAC X Base Address Register	MMA_MAC_XBASE	0x00222200
MMA MAC X Index Register	MMA_MAC_XINDEX	0x00222204
MMA MAC X Length Register	MMA_MAC_XLENGTH	0x00222208
MMA MAC X Modify Register	MMA_MAC_XMODIFY	0x0022220C
MMA MAC X Increment Register	MMA_MAC_XINCR	0x00222210
MMA MAC X Count Register	MMA_MAC_XCOUNT	0x00222214
MMA MAC Y Register Control Registers		
MMA MAC Y Base Address Register	MMA_MAC_YBASE	0x00222300
MMA MAC Y Index Register	MMA_MAC_YINDEX	0x00222304
MMA MAC Y Length Register	MMA_MAC_YLENGTH	0x00222308
MMA MAC Y Modify Register	MMA_MAC_YMODIFY	0x0022230C
MMA MAC Y Increment Register	MMA_MAC_YINCR	0x00222310
MMA MAC Y Count Register	MMA_MAC_YCOUNT	0x00222314

Table 15-1. MMA Module Register Memory Map (continued)

Description	Name	Address
MMA DCT/iDCT Registers		
DCT/iDCT Control Register	MMA_DCTCTRL	0x00222400
DCT/iDCT Version Register	MMA_DCTVERSION	0x00222404
DCT/iDCT IRQ Enable Register	MMA_DCTIRQENA	0x00222408
DCT/iDCT IRQ Status Register	MMA_DCTIRQSTAT	0x0022240C
DCT/iDCT Source Data Address	MMA_DCTSRCDATA	0x00222410
DCT/iDCT Destination Data Address	MMA_DCTDESDATA	0x00222414
DCT/iDCT X-Offset Address	MMA_DCTXOFF	0x00222418
DCT/iDCT Y-Offset Address	MMA_DCTOFF	0x0022241C
DCT/iDCT XY Count	MMA_DCTXYCNT	0x00222420
DCT/iDCT Skip Address	MMA_DCTSKIP	0x00222424
DCT/iDCT Data FIFO	MMA_DCTFIFO	0x00222500

15.3.1 MMA MAC Control Registers

There are 10 registers that control general MMA MAC operation.

15.3.1.1 MMA MAC Module Register

MMA_MAC_MOD MMA MAC Module Register **Addr 0x00222000**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RST															
TYPE	rw	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																MOD ENAB
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-2. MMA MAC Module Register Description

Name	Description	Settings
RST Bit 31	**Software Reset for the MAC**—indicates whether the reset sequence is complete.	0 = Reset is complete 1 = Reset is in progress
Reserved Bits 30–1	Reserved—These bits are reserved and should read 0.	
MOD ENAB Bit 0	**Module Enable**—Enables or disables the MAC.	0 = Disable the MAC 1 = Enable the MAC

15.3.1.2 MMA MAC Control Register

MMA_MAC_CTRL MMA MAC Control Register **Addr 0x00222004**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	X MODIFY PRESET	X INDEX CLR	X INDEX INCR	X INDEX LOAD	X_DATA_SEL	X SIGN ALT	X SIGN INI	X SIGNED	Y MODIFY PRESET	Y INDEX CLR	Y INDEX INCR	Y INDEX LOAD	Y_DATA_SEL	Y SIGN ALT	Y SIGN INI	Y SIGNED
TYPE	w	w	rw	rw	rw	rw	rw	rw	w	w	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														CACHE CLR	CACHE EN	OP EN
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	w	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-3. MMA MAC Control Register Description

Name	Description	Settings
X MODIFY PRESET Bit 31	**X Modify Preset**—Presets MMA_MAC_XMODIFY register to value 0x00000004.	0 = MMA_MAC_XMODIFY Register is not preset 1 = MMA_MAC_XMODIFY Register is preset to value 0x00000004
X INDEX CLR Bit 30	**X Index Clear**—Clears MMA_MAC_XINDEX register to value of 0x00000000.	0 = MMA_MAC_XINDEX Register is not reset. 1 = MMA_MAC_XINDEX Register is reset to value 0x00000000

Table 15-3. MMA MAC Control Register Description (continued)

Name	Description	Settings
X INDEX INCR Bit 29	**X Index Increment**—Determines whether the MMA_MAC_XINDEX register in the XDAC module is incremented by the value in the MMA_MAC_XINCR register for every (MMA_MAC_XCOUNT + 1) iteration. X INDEX INCR is used with X INDEX LOAD.	0 = MMA_MAC_XINDEX is not incremented 1 = MMA_MAC_XINDEX register is incremented
X INDEX LOAD Bit 28	**X Index Load**—Determines whether the index in the XDAC module is loaded with the values in the MMA_MAC_XINDEX register for every (MMA_MAC_XCOUNT + 1) iteration. When set, the Index is reloaded with the MMA_MAC_XINDEX register value. When cleared, the index retains the last stored value.	0 = XDAC index register is not reloaded 1 = XDAC index register is reloaded
X_DATA_SEL Bit 27	**X Data Select**—Selects which bits in the 32-bit data bus are used to create the 24-bit operand. **Note:** When the cache is enabled, X_DATA_SEL must be set to one.	0 = Bits [23:0] used as operand 1 = Bits [31:8] used as operand
X SIGN ALT Bit 26	**X Operand Sign Alternate**—Determines whether the X operand alternates its operand sign of the operation.	0 = X operand sign is not alternated 1 = X operand sign is alternated
X SIGN INI Bit 25	**X Operand Initial Sign**—Determines the X operand initial sign of the operation with x.	0 = +(x) 1 = -(x)
X SIGNED Bit 24	**X Operand Signed**—Determines whether the X operand is signed or unsigned. When set, the X operand is a signed value. When cleared, the X operand is unsigned.	0 = X operand is unsigned 1 = X operand is signed
Y MODIFY PRESET Bit 23	**Y Modify Preset**—Presets MMA_MAC_YMODIFY register to value 0x00000004.	0 = MMA_MAC_YMODIFY Register is not preset 1 = MMA_MAC_YMODIFY Register is preset to value 0x00000004
Y INDEX CLR Bit 22	**Y Index Clear**—Setting this bits clears the MMA_MAC_YINDEX register to value 0x00000000.	0 = MMA_MAC_YINDEX is not reset 1 = MMA_MAC_YINDEX is reset to value of 0x000 0000
Y INDEX INCR Bit 21	**Y Index Increment**—Determines whether the MMA_MAC_YINDEX register in the YDAC module is incremented by the value in the MMA_MAC_YINCR register for every (MMA_MAC_YCOUNT + 1) iteration. Y INDEX INCR is used with Y INDEX LOAD.	0 = MMA_MAC_YINDEX is not incremented 1 = MMA_MAC_YINDEX is incremented
Y INDEX LOAD Bit 20	**Y Index Load**—Determines whether the index in the YDAC module is loaded with the values in the MMA_MAC_YINDEX register for every (MMA_MAC_YCOUNT + 1) iteration. When set, the index is reloaded with the MMA_MAC_YINDEX register value. When cleared, the index retains the last stored value.	0 = Index register is not reloaded 1 = Index register is reloaded
Y_DATA_SEL Bit 19	**Y Data Select**—Selects which bits in the 32-bit data bus are used to create the 24-bit operand.	0 = Bits [23:0] used as operand 1 = Bits [31:8] used as operand

Table 15-3. MMA MAC Control Register Description (continued)

Name	Description	Settings
Y SIGN ALT Bit 18	**Y Operand Sign Alternate**—Determines whether the Y operand alternates its operand sign of the operation.	0 = Y operand sign is not alternated 1 = Y operand sign is alternated
Y SIGN INI Bit 17	**Y Operand Initial Sign**—Determines whether the Y operand initial sign of the operation with y.	0 = +(y) 1 = -(y)
Y SIGNED BIT 16	**Y Operand Signed**—Determines whether the Y operand is signed or unsigned. When set, the Y operand is a signed value. When cleared, the Y operand is unsigned.	0 = Y operand is unsigned 1 = Y operand is signed
Reserved Bits 15–3	Reserved—These bits are reserved and should read 0.	
CACHE CLR Bit 2	**Cache Clear**—Clears the X operand cache and writes the (base + index) register value into the cache block register.	0 = No effect 1 = Clear X operand cache and write (base+index) register value into the cache block register
CACHE EN Bit 1	**Cache Enable**—Enables or disables the X operand cache.	0 = X operand cache is disabled 1 = X operand cache is enabled
OP EN Bit 0	**Operation Enable**—Initiates MAC operation and indicates if the operation is complete. The operation does not start if MMA_MAC_MULT register contains 0.	0 = MAC operation is complete 1 = Initiate MAC operation or MAC operation is not complete

15.3.1.3 MMA MAC Multiply Counter Register

MMA_MAC_MULT MMA MAC Multiply Counter Register **Addr 0x00222008**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0X0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MULT COUNTER															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0X0000															

Table 15-4. MMA MAC Multiply Counter Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
MULT COUNTER Bits 15–0	**Multiply Counter**—Determines the number of multiply operations that the MAC module performs. For proper operation, this value must be an integer multiple of the (MMA_MAC_ACCU + 1) value.

15.3.1.4 MMA MAC Accumulate Counter Register

MMA_MAC_ACCU MMA MAC Accumulate Counter Register **Addr 0x0022200C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0X0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ACCU COUNTER															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0X0000															

Table 15-5. MMA MAC Accumulate Counter Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
ACCU COUNTER Bits 15–0	**Accumulate Counter**—Determines the number of accumulate operations that the MAC module performs before writing the accumulated result to the FIFO. The value written to this register is the actual value minus 1 (0x0003 for four accumulate operations).

15.3.1.5 MMA MAC Interrupt Register

MMA_MAC_INTR MMA MAC Interrupt Register **Addr 0x00222010**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												OP ERROR	OP END	FIFO EMPT	FIFO HALF	FIFO FULL
TYPE	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	0x0004															

Table 15-6. MMA MAC Interrupt Register Description

Name	Description	Settings
Reserved Bits 31–5	Reserved—These bits are reserved and should read 0.	
OP ERROR Bit 4	**OP ERROR Interrupt**—Sets when there is a memory access error from the bus. Write 1 to clear	0 = No error in operation. 1 = Error in operation.
OP END Bit 3	**Operation End Interrupt**—Sets when the MAC operation ends. Write a 1 to clear.	0 = MAC operation is not complete 1 = MAC operation is complete
FIFO EMPT Bit 2	**FIFO Empty Interrupt**—Sets when the FIFO is empty. Write a 1 to clear.	0 = FIFO is not empty 1 = FIFO is empty
FIFO HALF Bit 1	**FIFO Half Full Interrupt**—Sets when the FIFO is half full. Write a 1 to clear.	0 = FIFO is not half full 1 = FIFO is half full
FIFO FULL Bit 0	**FIFO Full Interrupt**—Sets when the FIFO is full. Write a 1 to clear.	0 = FIFO is not full 1 = FIFO is full

15.3.1.6 MMA MAC Interrupt Mask Register

MMA_MAC_INTR_MASK MMA MAC Interrupt Mask Register **Addr 0x00222014**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												OP ERROR	OP END	FIFO EMPT	FIFO HALF	FIFO FULL
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
	0x001F															

Table 15-7. MMA MAC Interrupt Mask Register Description

Name	Description	Settings
Reserved Bits 31–5	Reserved—These bits are reserved and should read 0.	
OP ERROR Bit 4	**OP ERROR Mask**—Masks the OP ERROR Interrupt.	0 = Mask on/enable interrupt 1 = Mask off/disable interrupt.
OP END Bit 3	**Operation End Interrupt Mask**—Masks the OP END interrupt.	0 = Mask on/enable interrupt 1 = Mask off/disable interrupt
FIFO EMPT Bit 2	**FIFO Empty Interrupt Mask**—Masks the FIFO EMPT interrupt.	0 = Mask on/enable interrupt 1 = Mask off/disable interrupt
FIFO HALF Bit 1	**FIFO Half Full Interrupt Mask**—Masks the FIFO HALF interrupt.	0 = Mask on/enable interrupt 1 = Mask off/disable interrupt
FIFO FULL Bit 0	**FIFO Full Interrupt Mask**—Masks the FIFO FULL interrupt.	0 = Mask on/enable interrupt 1 = Mask off/disable interrupt

15.3.1.7 MMA MAC FIFO Register

MMA_MAC_FIFO MMA MAC FIFO Register **Addr 0x00222018**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	FIFO REGISTER															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FIFO REGISTER															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-8. MMA MAC FIFO Register Description

Name	Description
FIFO REGISTER Bits 31–0	**FIFO Read Register**—Returns FIFO output.

15.3.1.8 MMA MAC FIFO Status Register

MMA_MAC_FIFO_STAT MMA MAC FIFO Status Register **Addr 0x0022201C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
												FIFO COUNT				
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														FIFO EMPT	FIFO HALF	FIFO FULL
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	0x0004															

Table 15-9. MMA MAC FIFO Status Register Description

Name	Description	Settings
Reserved Bits 31–21	Reserved—These bits are reserved and should read 0.	
FIFO COUNT Bits 20–16	**FIFO Data Count**—Indicates the number of data in the FIFO.	See description
Reserved Bits 15–3	Reserved—These bits are reserved and should read 0.	
FIFO EMPT Bit 2	**FIFO Empty Status**—Indicates the status of the FIFO EMPT interrupt.	See description
FIFO HALF Bit 1	**FIFO Half Full Status**—Indicates the status of the FIFO HALF interrupt.	See description
FIFO FULL Bit 0	**FIFO Full Status**—Indicates the status of the FIFO FULL interrupt.	See description

15.3.1.9 MMA MAC Burst Count Register

MMA_MAC_BURST MMA MAC Burst Count Register **Addr 0x00222020**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									BURST COUNT							
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-10. MMA MAC Burst Count Register Description

Name	Description
Reserved Bits 31–8	Reserved—These bits are reserved and should read 0.
BURST COUNT Bits 7–0	**Memory Access Burst Count**—Determines the maximum number of read accesses to memory allowed in one burst. This feature ensures that the MMA does not hold the memory bus for too long.

15.3.1.10 MMA MAC Bit Select Register

MMA_MAC_BITSEL MMA MAC Bit Select Register **Addr 0x00222024**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														BITSEL		
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 15-11. MMA MAC Bit Select Register Description

Name	Description	Settings
Reserved Bits 31–3	Reserved—These bits are reserved and should read 0.	
BITSEL Bits 2–0	**Accumulator Output Bit Select**—Selects which 32–bit subset of the 56–bit accumulator is stored in the FIFO.	000 = bits 31:0 001 = bits 35:4 010 = bits 39:8 011 = bits 43:12 100 = bits 47:16 101 = bits 51:20 110 = bits 55:24

15.3.2 MMA MAC XY Count Accumulate Register

MMA_MAC_XYACCU MMA MAC XY Count Accumulate Register **Addr 0x00222040**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	MMA_MAC_XY_COUNT															
TYPE	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MMA_MAC_XY_COUNT															
TYPE	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

15.3.3 MMA MAC X Register Control Registers

There are 6 registers that reside in the X operand Data Access Controller (XDAC). The XDAC has an associated cache. The initial access by the XDAC is a cache miss, so the operand is fetched from memory and stored in the cache. Subsequent accesses to the same location cause cache hits, so the data is loaded from the cache instead of from memory.

15.3.3.1 MMA MAC X Base Address Register

MMA_MAC_XBASE MMA MAC X Base Address Register **Addr 0x00222200**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	XBASE															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	XBASE															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-12. MMA MAC X Base Address Register Description

Name	Description
XBASE Bits 31–0	**X Base Address**—Determines the base/start address of the X data buffer. Writing 0 to this register will force the module to use the data stored in the cache when Cache_En bit is set.

15.3.3.2 MMA MAC X Index Register

MMA_MAC_XINDEX MMA MAC X Index Register **Addr 0x00222204**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	XINDEX															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-13. MMA MAC X Index Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
XINDEX Bits 15–0	**MAC X Index**—Determines part of the offset from the base address. The X Address Index (actual access address) is equal to MMA_MAC_XBASE + contents of MMA_MAC_XINDEX + [contents of MMA_MAC_XMODIFY + the number of iterations since initialization, the index last wrapped, or the last MMA_MAC_XCOUNT iterations (when the X INDEX LOAD bit is set)].

15.3.3.3 MMA MAC X Length Register

MMA_MAC_XLENGTH MMA MAC X Length Register **Addr 0x00222208**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	COLUMN															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LENGTH															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-14. MMA MAC X Length Register Description

Name	Description
COLUMN Bits 31–16	**Column Size**—MMA_MAC_XINDEX wraps around to the base of the buffer when MMA_MAC_XINDEX + MMA_MAC_XINCR is greater than COLUMN. Writing 0 to this register will disable the wrapping of address. **Note:** Note: If the current MMA_MAC_XINDEX is 12 and the COLUMN is 16, MMA_MAC_XINCR is 8, then the next XINDEX will be (12+8)% 16 == 4. The physical address is 4 + MMA_MAC_XBASE.
LENGTH Bits 15–0	**Circular Length**—Circular Length - Determines the size of the X data buffer. Address_Index wraps back to base of buffer when Address_Index + MMA_MAC_XMODIFY is greater than LENGTH. Writing 0 to this register will disable the wrapping of address. **Note:** Note: If the current Address_Index is 12 and the LENGTH is 16, MMA_MAC_XMODIFY is 8, then the next Address_Index will be (12+8)% 16 == 4. The physical address is 4 + MMA_MAC_XBASE.

15.3.3.4 MMA MAC X Modify Register

MMA_MAC_XMODIFY MMA MAC X Modify Register **Addr 0x0022220C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

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

Table 15-15. MMA MAC X Modify Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
XMODIFY Bits 15–0	**X Increment**—Determines the size of the increment to the X Address Index after each iteration.

15.3.3.5 MMA MAC X Increment Register

MMA_MAC_XINCR MMA MAC X Increment Register **Addr 0x00222210**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	XINCR															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-16. MMA MAC X Increment Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
XINCR Bits 15–0	**X Increment**—Determines the size of the increment to the X Address Index after each (MMA_MAC_XCOUNT +1) iteration.

15.3.3.6 MMA MAC X Count Register

This 16–bit read/write register sets the MAC Count value for the increment of the Index register when the X INDEX INCR bit in the MMA_MAC_CTRL register is set.

MMA_MAC_XCOUNT MMA MAC X Count Register **Addr 0x00222214**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	XCOUNT															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	0x0004															

Table 15-17. MMA MAC X Count Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
XCOUNT Bits 15–0	**X Count**—Determines the number of iterations required to: • reload the XDAC Address Index with [MMA_MAC_XBASE + MMA_MAC_XINDEX] (when the X INDEX LOAD bit is set) or, • increment the MMA_MAC_XINDEX by the value in MMA_MAC_XINCR (when the X INDEX INCR bit is set). The value written to this register is the actual value minus 1 (0x0003 for four iterations).

15.3.4 MMA MAC Y Register Control Registers

There are six registers that reside in the Y operand Data Access Controller (YDAC). The YDAC does not have a cache, so it always fetches data from memory.

15.3.4.1 MMA MAC Y Base Address Register

MMA_MAC_YBASE MMA MAC Y Base Address Register **Addr 0x00222300**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	YBASE															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	YBASE															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-18. MMA MAC Y Base Address Register Description

Name	Description
YBASE Bits 31–0	**Y Base Address**—Determines the base/start address of the Y data buffer.

15.3.4.2 MMA MAC Y Index Register

MMA_MAC_YINDEX MMA MAC Y Index Register **Addr 0x00222304**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	YINDEX															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-19. MMA MAC Y Index Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
YINDEX Bits 15–0	**Y Index**—Determines part of the offset from the base address. The Y Address Index (actual access address) is equal to MMA_MAC_YBASE + MMA_MAC_YINDEX + [MMA_MAC_YMODIFYr × the number of iterations since initialization, the index last wrapped, or the last MMA_MAC_YCOUNT iterations (if the Y INDEX LOAD bit is set)].

15.3.4.3 MMA MAC Y Length Register

MMA_MAC_YLENGTH MMA MAC Y Length Register **Addr 0x00222308**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	COLUMN															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LENGTH															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-20. MMA MAC Y Length Register Description

Name	Description
COLUMN Bits 31–16	**Column Size**—MMA_MAC_YINDEX wraps around to the base of the buffer when MMA_MAC_YINDEX + MMA_MAC_YINCR is greater than COLUMN. Writing 0 to this register will disable the wrapping of address. **Note:** Note: If the current MMA_MAC_YINDEX is 12 and the COLUMN is 16, MMA_MAC_YINCR is 8, then the next YINDEX will be (12+8)% 16 == 4. The physical address is 4 + MMA_MAC_YBASE.
LENGTH Bits 15–0	**Circular Length**—Circular Length - Determines the size of the Y data buffer. Address_Index wraps back to base of buffer when Address_Index + MMA_MAC_YMODIFY is greater than LENGTH. Writing 0 to this register will disable the wrapping of address. **Note:** Note: If the current Address_Index is 12 and the LENGTH is 16, MMA_MAC_YMODIFY is 8, then the next Address_Index will be (12+8)% 16 == 4. The physical address is 4 + MMA_MAC_YBASE.

15.3.4.4 MMA MAC Y Modify Register

MMA_MAC_YMODIFY MMA MAC Y Modify Register **Addr 0x0022230C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	YMODIFY															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	0x0004															

Table 15-21. MMA MAC Y Modify Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
YMODIFY Bits 15–0	**Y Increment**—Determines the size of the increment to the Y Address Index after each iteration.

15.3.4.5 MMA MAC Y Increment Register

MMA_MAC_YINCR MMA MAC Y Increment Register **Addr 0x00222310**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	YINCR															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-22. MMA MAC Y Increment Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
YINCR Bits 15–0	**Y Increment**—Determines the size of the increment to the MMA_MAC_YINDEX register after each (MMA_MAC_YCOUNT+1) iteration.

15.3.4.6 MMA MAC Y Count Register

MMA_MAC_YCOUNT MMA MAC Y Count Register **Addr 0x00222314**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	YCOUNT															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-23. MMA MAC Y Count Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
YCOUNT Bits 15–0	**Y Count**—Determines the number of iterations required to: • reload the YDAC Address Index with [MMA_MAC_YBASE + MMA_MAC_YINDEX] (when the Y INDEX LOAD bit is set) or, • increment MMA_MAC_YINDEX by the value in MMA_MAC_YINCR (when the Y INDEX INCR bit is set). The value written to this register is the actual value minus 1 (0x0003 for four iterations).

15.3.5 MMA DCT/iDCT Registers

There are 11 registers that control DCT/iDCT functions.

15.3.5.1 DCT/iDCT Control Register

This register controls the function of DCT/iDCT and the interface module. This is a read/write register.

MMA_DCTCTRL DCT/iDCT Control Register **Addr 0x00222400**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			DCT_HWORD_SWAP	DATAINSHIFT				SEMA EN	DCT XPOSE	DCT CLK EN	SW RST	DCT BY PASS	ARMMCM SEL		DCT/ IDCT	DCT ENA
TYPE	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 15-24. DCT/iDCT Control Register Description

Name	Description	Setting
Reserved Bits 31–14	Reserved—These bits are reserved and should read 0.	
DCT_HWORD_SWAP Bit 13	**DCT Half-Word Swap**—This bit determines if the data is swapped in half-word (16-bit) or word (32-bit).	0 = Not swap data in half word, result in data{data[31:0]} 1 = Swap data in half word data{data[15:0], data[31:16]
DATAINSHIFT Bits 12–9	**DATAINSHIFT**—Reserved for future use and should be written with 0 for normal operation.	
SEMAEN Bit 8	**SEMAEN**—Reserved for future use and must be written with 0 for normal operation.	
DCTXPOSE Bit 7	**DCT Transpose**—Enables/Disables the DCT output to be transposed.	0 = Disable transpose 1 = Enable transpose
DCTCLKEN Bit 6	**DCT Clock Enable**—Enables/Disables the DCT clock to save power when DCT is not in use.	0 = Disable clock 1 = Enable clock
SWRST Bit 5	**Software Reset**—Resets the DCT Module.	0 = No effect 1 = Resets DCT module

Table 15-24. DCT/iDCT Control Register Description (continued)

Name	Description	Setting
DCTBYPASS Bit 4	**DCT Bypass**—Enables DCT/iDCT input data to be bypassed to the output without being transformed.	0 = Perform transform 1 = Bypass
ARMMCMSEL Bits 3–2	**ARM/Memory Controller Select**—Controls DCT/iDCT module data input and output.	00 = Data in and out through memory controller 01 = Data in through memory controller and out through ARM9 core. 10 = Data in through ARM9 core and out through memory controller. 11 = Data in and out through ARM9 core.
DCT/IDCT Bit 1	**DCT/IDCT**—Selects either DCT or iDCT.	0 = iDCT 1 = DCT
DCT ENA Bit 0	**DCT Enable**—Enables/Disables the DCT/iDCT. If data is accessed through the memory controller, DCT ENA is reset to zero after an 8 × 8 transform is completed.	0 = DCT disabled 1 = DCT enabled

15.3.5.2 DCT/iDCT Version Register

MMA_DCTVERSION DCT/iDCT Version Register **Addr 0x00222404**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	VERSION NUMBER															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	VERSION NUMBER															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0x0000															

Table 15-25. DCT/iDCT Version Register Description

Name	Description
VERSION NUMBER Bits 31–0	**Version Number**—Contains the version number of the DCT/iDCT block.

15.3.5.3 DCT/iDCT IRQ Enable Register

MMA_DCTIRQENA DCT/iDCT IRQ Enable Register **Addr 0x00222408**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											ERR INTR EN	DODEN	DIDEN	DOIEN	DIIEN	DCT COMP
TYPE	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-26. DCT/iDCT IRQ Enable Register Description

Name	Description	Settings
Reserved Bits 31–6	Reserved—These bits are reserved and should read 0.	
ERRINTREN Bit 5	**Error Interrupt Enable**—Enables/Disables interrupt generation when an error occurs while accessing memory through the memory controller.	0 = Interrupt disabled 1 = Interrupt enabled
DODEN Bit 4	**DMA Output Data Enable**—Enables/Disables interrupt generation when the DMA data out request signal is asserted.	0 = Interrupt disabled 1 = Interrupt enabled
DIDEN Bit 3	**DMA Input Data Enable**—Enables/Disables interrupt generation when the DMA data in request signal is asserted.	0 = Interrupt disabled 1 = Interrupt enabled
DOIEN Bit 2	**Data Output Interrupt Enable**—Enables/Disables interrupt generation when the data out signal is asserted.	0 = Interrupt disabled 1 = Interrupt enabled
DIIEN Bit 1	**Data Input Interrupt Enable**—Enables/Disables interrupt generation when the data in signal is asserted.	0 = Interrupt disabled 1 = Interrupt enabled
DCTCOMP Bit 0	**DCT Complete**—Enables/Disables interrupt generation when the DCT block completes a set of 8 × 8 transforms.	0 = Interrupt disabled 1 = Interrupt enabled

15.3.5.4 DCT/iDCT IRQ Status Register

MMA_DCTIRQSTAT DCT/iDCT IRQ Status Register **Addr 0x0022240C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											FIFO FULL	FIFO EMP	ERR INTR	DOUT INTR	DIN INTR	DCT COMP
TYPE	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-27. DCT/iDCT IRQ Status Register Description

Name	Description	Settings
Reserved Bits 31–6	Reserved—These bits are reserved and should read 0.	
FIFO FULL Bit 5	**FIFO Full**—Indicates whether the FIFO is full or not. Remains high until all data is read from the FIFO. Write a 1 to clear.	0 = Output FIFO is not filled 1 = Output FIFO is filled, some data left unread
FIFO EMP Bit 4	**FIFO Empty**—Indicates whether the FIFO is filled with input data or not.The bit is cleared automatically when the input FIFO becomes full. Write a 1 to clear.	0 = Input FIFO is not filled 1 = Input FIFO is full
ERR INTR Bit 3	**Error Interrupt**—Indicates whether an error has occurred while accessing memory through the memory controller. Write a 1 to clear.	0 = No error has occurred 1 = An error has occurred
DOUTINTR Bit 2	**Data Out Interrupt**—Determines when the data out interrupt is asserted. Write a 1 to clear.	0 = Data out interrupt has occurred 1 =Data out interrupt has not occurred
DININTR Bit 1	**Data In Interrupt**—Determines when the data in interrupt is asserted. Write a 1 to clear.	0 = Data In interrupt has occurred 1 =Data In interrupt has not occurred
DCTCOMP Bit 0	**Transform Complete**—Indicates whether a transform has completed. Write a 1 to clear.	0 = Transform is not complete 1 = Transform is complete

15.3.5.5 DCT/iDCT Source Data Address

MMA_DCTSRCDATA DCT/iDCT Source Data Address **Addr 0x00222410**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DCT_SRC_ADDR															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DCT_SRC_ADDR															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-28. DCT/iDCT Source Data Address Register Description

Name	Description
DCT_SRC_ADDR Bits 31–0	**DCT Source Address**—Determines the source address of the data to be transformed.

15.3.5.6 DCT/iDCT Destination Data Address

MMA_DCTDESDATA DCT/iDCT Destination Data Address **Addr 0x00222414**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DCT_DES_ADDR															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DCT_DES_ADDR															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-29. DCT/iDCT Destination Data Address Register Description

Name	Description
DCT_DES_ADDR Bits 31–0	**DCT Destination Address**—Determines the destination address for the transformed data.

15.3.5.7 DCT/iDCT X-Offset Address

MMA_DCTXOFF DCT/iDCT X-Offset Address **Addr 0x00222418**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	X-OFFSET															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-30. DCT/iDCT X-Offset Address Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
X-OFFSET Bits 15–0	**X Offset**—Determines the offset address along the X-direction from the last transformed block. For the first block, the start address is the same as MMA_DCTSRCDATA or MMA_DCTDESDATA. For the following blocks the start address is [MMA_DCTSRCDATA or MMA_DCTDESDATA] + (X-OFFSET × N) where N = 1, …(X-COUNT – 1) along the X-direction.

15.3.5.8 DCT/iDCT Y-Offset Address

MMA_DCTOFF DCT/iDCT Y-Offset Address **Addr 0x0022241C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Y-OFFSET															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-31. DCT/iDCT Y-Offset Address Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
Y-OFFSET Bits 15–0	**Y Offset**—Determines the number of bytes to skip *in the Y direction* when accessing each successive row in a block of data.

15.3.5.9 DCT/iDCT XY Count

MMA_DCTXYCNT DCT/iDCT XY Count **Addr 0x00222420**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		Y-COUNT								X-COUNT						
TYPE	r	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-32. DCT/iDCT XY Count Register Description

Name	Description
Reserved Bits 31–15	Reserved—These bits are reserved and should read 0.
Y-COUNT Bits 14–8	**Y Count**—Controls the number of blocks to be transformed in the Y direction.
Reserved Bit 7	Reserved—This bit is reserved and should read 0.
X-COUNT Bits 6–0	**X Count**—Controls the number of blocks to be transformed in the X direction.

15.3.5.10 DCT/iDCT Skip Address

MMA_DCTSKIP DCT/iDCT Skip Address **Addr 0x00222424**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SKIP_ADDR															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-33. DCT/iDCT Skip Address Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
SKIP_ADDR Bits 15–0	**SKIP_ADDR**—Determines the number of bytes to skip *in the X direction* when accessing each successive row in a block of data.

15.3.5.11 DCT/iDCT Data FIFO

MMA_DCTFIFO DCT/iDCT Data FIFO **Addr 0x00222500**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DATA															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 15-34. DCT/iDCT Data FIFO Register Description

Name	Description
DATA Bits 31–0	**Data**—Stores input data to be transformed and the outputs the data after transformation. Writing to this register stores input data in the FIFO. Reading this register retrieves the results of the transformation from the FIFO. The FIFO is 32 × 32–bits.

Chapter 16
Serial Peripheral Interface Modules (SPI 1 and SPI 2)

The MC9328MXL contains two identical serial peripheral interface modules (SPI 1 and SPI 2). Due to pin availability, the SPI 2 module operates in master mode only and does not support the SPI Ready ($\overline{\text{SPI_RDY}}$) control signal function.

SPI 1 signals are multiplexed with GPIO ports as primary functions. See Chapter 2, “Signal Descriptions and Pin Assignments,” for detailed pin assignments. SPI 2 signals are connected to the “IN” and “OUT” ports of the GPIO modules. The user must configure the corresponding GPIO registers to make SPI 1 and SPI 2 signals available at the pins. See Section 16.2.3, “Pin Configuration for SPI 1 and SPI 2,” for more information about selecting SPI 1 and SPI 2 signals.

Table 16-1. SPI 1 and SPI 2 Signal Multiplexing

SPI Signal Names	Connect to GPIO Signal
SPI1_$\overline{\text{SPI_RDY}}$	Primary function of GPIO port C [13]
SPI1_SCLK	Primary function of GPIO port C [14]
SPI1_$\overline{\text{SS}}$	Primary function of GPIO port C [15]
SPI1_MISO	Primary function of GPIO port C [16]
SPI1_MOSI	Primary function of GPIO port C [17]
SPI2_SCLK	AIN of GPIO port A [0] or AIN of port D [7]
SPI2_$\overline{\text{SS}}$	AIN of GPIO port A [17] or AIN of port D [8]
SPI2_RXD	AOUT of GPIO port A [1] or AOUT of port D [9]
SPI2_TXD	BIN of GPIO port D [31] or AIN of port D [10]

16.1 SPI Block Diagram

This section describes how the SPI modules communicate with external devices.

Each SPI module has one 8 × 16-bit receive buffer (RXFIFO) and one 8 × 16-bit transmit buffer (TXFIFO). The SPI ready ($\overline{\text{SPI_RDY}}$) and slave select ($\overline{\text{SS}}$) control signals enable fast data communication with fewer software interrupts. The block diagram shown in Figure 16-1 on page 16-2is the same for each SPI module, except that the SPI 2 module does not support the $\overline{\text{SPI_RDY}}$ control signal function.

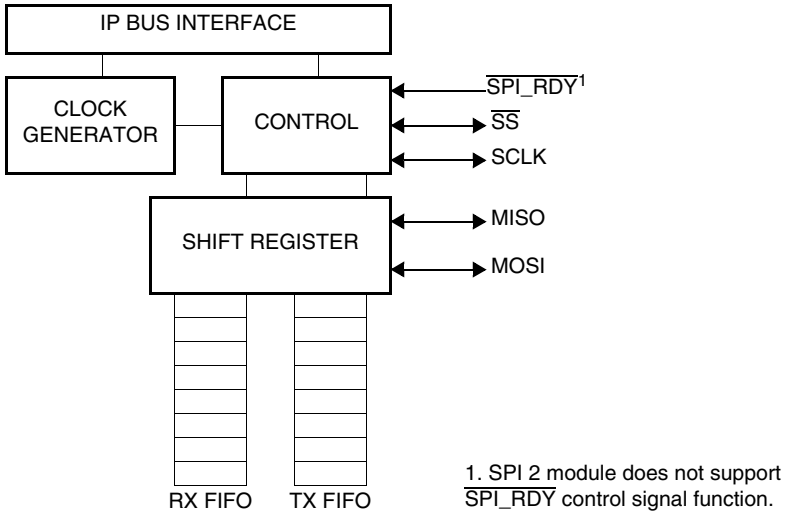


Figure 16-1. SPI Module Block Diagram

16.2 Operation

To use the internal transmit (TX) and receive (RX) data FIFOs when the SPI 1 module is configured as a master, two control signals are used for data transfer rate control: the $\overline{SS}$ signal (output) and the $\overline{SPI_RDY}$ signal (input). The SPI 1 Sample Period Control Register (PERIODREG1) and the SPI 2 Sample Period Control Register (PERIODREG2) can also be programmed to a fixed data transfer rate for either SPI 1 or SPI 2.

When the SPI 1 module is configured as a slave, the user can configure the SPI 1 Control Register (CONTROLREG1) to match the external SPI master's timing. In this configuration, $\overline{SS}$ becomes an input signal, and is used to latch data into or load data out to the internal data shift registers, as well as to increment the data FIFO. Figure 16-2 on page 16-3 shows the generic SPI timing.

16.2.1 Phase and Polarity Configurations

The SPI master uses the SCLK signal to transfer data in and out of the shift register. Data is clocked by one of four programmable clock phase and polarity combinations, selected through the phase (PHA) and polarity (POL) bits in the CONTROLREG1 and CONTROLREG2 registers.

In Phase 0 operation (PHA=0) and SCLK Polarity active low (POL=0), output data changes on falling edges of the SCLK signal and input data is shifted in on rising edges. The most significant bit (MSB) is output when the CPU loads the transmitted data.

In Phase 0 operation (PHA=0) and SCLK Polarity active high (POL=1), output data changes on rising edges of the SCLK signal and input data is shifted in on falling edges. The most significant bit (MSB) is output on the first rising edge of the SCLK signal.

In Phase 1 operation (PHA=1) and SCLK Polarity active low (POL=0), output data changes on rising edges of the SCLK signal and input data is shifted in on falling edges. The MSB is output on the first rising edge of the SCLK signal.

In Phase 1 operation (PHA=1) and SCLK Polarity active high (POL=1), output data changes on falling edges of the SCLK signal and input data is shifted in on rising edges. The MSB is output when the CPU loads the transmitted data.

The polarity of SCLK may be configured (to invert the SCLK signal), however it does not change the edge-triggered events that are internal to the SPI 1and 2. This flexibility allows the SPI modules to operate with most currently available serial peripheral devices. Figure 16-2 shows the relationship of the polarity and phase settings.

16.2.2 Signals

The following signals are used to control the serial peripheral interface master:

- Master Out Slave In (MOSI)—In master mode, this bidirectional signal is a TX output signal from the data shift register. In slave mode, it is an RX input.
- Master In Slave Out (MISO)—In master mode, this bidirectional signal is a RX input signal to the data shift register. In slave mode, it is a TX output.
- SPI Clock (SCLK)—In master mode, this bidirectional signal is an SPI clock output. In slave mode, it is an input.
- Slave Select ($\overline{SS}$)—In master mode, this bidirectional signal is an output. In slave mode, it is an input.
- SPI Ready ($\overline{SPI_RDY}$)—Used only in master mode to edge- or level-trigger an SPI burst. The SPI 2 module does not support this signal.

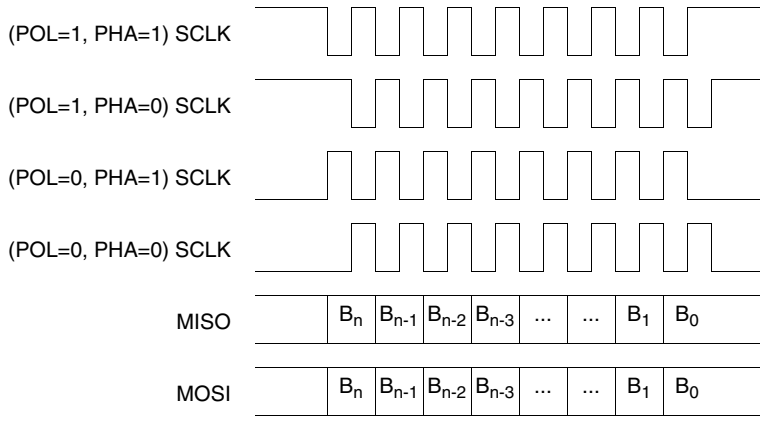


Figure 16-2. SPI Generic Timing

16.2.3 Pin Configuration for SPI 1 and SPI 2

Table 16-1 lists the pins used for the SPI 1 and SPI 2 modules. These pins are multiplexed with other functions on the device, and must be configured for SPI operation.

NOTE:

The user must ensure that the data direction bits in the GPIO are set to the correct direction for proper operation. See Section 29.5.1, “Data Direction Registers,” on page 29-8 for details.

Table 16-2. SPI Pin Configuration

Pin	Setting[1]	Configuration Procedure
SPI1_$\overline{\text{SPI_RDY}}$	Primary function of GPIO port C [13]	1. Clear bit 13 of Port C GPIO In Use Register (GIUS_C) 2. Clear bit 13 of Port C General Purpose Register (GPR_C)
SPI1_SCLK	Primary function of GPIO port C [14]	1. Clear bit 14 of Port C GPIO In Use Register (GIUS_C) 2. Clear bit 14 of Port C General Purpose Register (GPR_C)
SPI1_$\overline{\text{SS}}$	Primary function of GPIO port C [15]	1. Clear bit 15 of Port C GPIO In Use Register (GIUS_C) 2. Clear bit 15 of Port C General Purpose Register (GPR_C)
SPI1_MISO	Primary function of GPIO port C [16]	1. Clear bit 16 of Port C GPIO In Use Register (GIUS_C) 2. Clear bit 16 of Port C General Purpose Register (GPR_C)
SPI1_MOSI	Primary function of GPIO port C [17]	1. Clear bit 17 of Port C GPIO In Use Register (GIUS_C) 2. Clear bit 17 of Port C General Purpose Register (GPR_C)
SPI2_SCLK[2]	AIN of GPIO port A [0]	1. Set bit 0 of port A GPIO In Use Register (GIUS_A) 2. Clear bits 1 and 0 of port A Output Configuration Register 1 (OCR1_A) 3. Set bit 0 of port A Data Direction Register (DDIR_A)
	AIN of port D [7]	1. Set bit 7 of port D GPIO In Use Register (GIUS_D) 2. Clear bits 15 and 14 of port D Output Configuration Register 1 (OCR1_D) 3. Set bit 7 of port D Data Direction Register (DDIR_D)
SPI2_$\overline{\text{SS}}$[2]	AIN of GPIO port A [17]	1. Set bit 17 of port A GPIO In Use Register (GIUS_A) 2. Clear bits 3 and 2 of port A Output Configuration Register 2 (OCR2_A) 3. Set bit 17 of port A Data Direction Register (DDIR_A)
	AIN of port D [8]	1. Set bit 8 of port D GPIO In Use Register (GIUS_D) 2. Clear bits 17 and 16 of port D Output Configuration Register 1 (OCR1_D) 3. Set bit 8 of port D Data Direction Register (DDIR_D)
SPI2_RXD[2]	AOUT of GPIO port A [1]	1. Set bit 1 of port A GPIO In Use Register (GIUS_A) 2. Clear bits 3 and 2 of port A Input Configuration Register A1 (ICONFA1_A) 3. Clear bit 1 of port A Data Direction Register (DDIR_A) 4. Clear bit 8 in the FMCR register.
	AOUT of GPIO port D [9]	1. Set bit 9 of Port D GPIO In Use Register (GIUS_D) 2. Clear bits 19 and 18 of Port D Input Configuration Register D1 (ICONFA1_D) 3. Clear bit 9 of Port D Data Direction Register (DDIR_D) 4. Set bit 8 in the FMCR register
SPI2_TXD[2]	BIN of GPIO port D [31]	1. Set bit 31 of port D GPIO In Use Register (GIUS_D) 2. Clear bit 31 and set bit 30 of port D Output Configuration Register 2 (OCR2_D) 3. Set bit 31 of port D Data Direction Register (DDIR_D)
	AIN of port D [10]	1. Set bit 10 of port D GPIO In Use Register (GIUS_D) 2. Clear bits 21 and 20 of port D Output Configuration Register 1 (OCR1_D) 3. Set bit 10 of port D Data Direction Register (DDIR_D)

1. SPI 1 pins must only be configured if SPI 1 is being used. SPI 2 pins must only be configured if SPI 2 is being used.
2. Only one of the two pins must be set-up for the SPI 2 signal.

16.3 Programming Model

Each SPI module includes eight 32-bit registers. Table 16-3 summarizes these registers and their addresses.

Table 16-3. SPI Module Register Memory Map

Description	Name	Address
SPI 1 Rx Data Register	RXDATAREG1	0x00213000
SPI 1 Tx Data Register	TXDATAREG1	0x00213004
SPI 1 Control Register	CONTROLREG1	0x00213008
SPI 1 Interrupt Control/Status Register	INTREG1	0x0021300C
SPI 1 Test Register	TESTREG1	0x00213010
SPI 1 Sample Period Control Register	PERIODREG1	0x00213014
SPI 1 DMA Control Register	DMAREG1	0x00213018
SPI 1 Soft Reset Register	RESETREG1	0x0021301C
SPI 2 Rx Data Register	RXDATAREG2	0x00219000
SPI 2 Tx Data Register	TXDATAREG2	0x00219004
SPI 2 Control Register	CONTROLREG2	0x00219008
SPI 2 Interrupt Control/Status Register	INTREG2	0x0021900C
SPI 2 Test Register	TESTREG2	0x00219010
SPI 2 Sample Period Control Register	PERIODREG2	0x00219014
SPI 2 DMA Control Register	DMAREG2	0x00219018
SPI 2 Soft Reset Register	RESETREG2	0x0021901C

16.3.1 Receive (RX) Data Registers

The SPI receive data registers are read-only registers that form the top word of the 8 × 16 RXFIFOs for each SPI module. These registers hold data received from an external SPI device during a data transaction.

		Addr
RXDATAREG1	SPI 1 Rx Data Register	**0x00213000**
RXDATAREG2	SPI 2 Rx Data Register	**0x00219000**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 16-4. SPI 1 Rx Data Register and SPI 2 Rx Data Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
DATA Bits 15–0	**DATA**—Holds the top word of data received into the FIFO. Not valid when the Receive Data Ready (RR) bit in the corresponding Interrupt Control/Status Register (INTREG1 or INTREG2) is cleared.

16.3.2 Transmit (TX) Data Registers

The SPI transmit data registers are write-only data registers that form the top word of the 8 × 16 TXFIFOs for each SPI module. The TXFIFO can be written to as long as it is not full, even when the exchange (XCH) bit in the corresponding SPI control register (CONTROLREG1 or CONTROLREG2) is set. This allows user write access to the TXFIFO during an SPI data exchange process. Writes to these registers are ignored when the SPI module is disabled (SPIEN bit of the corresponding SPI control register is cleared).

		Addr
TXDATAREG1	SPI 1 Tx Data Register	**0x00213004**
TXDATAREG2	SPI 2 Tx Data Register	**0x00219004**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA															
TYPE	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 16-5. SPI 1 Tx Data Register and SPI 2 Tx Data Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
DATA Bits 15–0	**DATA**—Holds the top word of data loaded into the FIFO. Data written to this register can be 8 or 16 bits. The number of bits actually transmitted is determined by the BIT_COUNT field of the corresponding SPI control register. If this field contains more bits than the number specified by BIT_COUNT, the extra bits are ignored. For example, to transfer 10 bits of data, a 16-bit word must be written to this register. Bits 9-0 are shifted out and bits 15-10 are ignored. When the SPI module is operating in slave mode, ‘0’s are shifted out when the FIFO is not full.

16.3.3 Control Registers

The SPI control registers allow the user to enable the SPI modules, select the operating modes, specify the divider value, phase, and polarity of the clock, configure the $\overline{SS}$ and $\overline{SPI_RDY}$ control signals (for SPI 1), and define the transfer length.

		Addr
CONTROLREG1	SPI 1 Control Register	**0x00213008**
CONTROLREG2	SPI 2 Control Register	**0x00219008**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATARATE			DRCTL		MODE	SPIEN	XCH	SSPOL	SSCTL	PHA	POL	BIT_COUNT			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0/1[1]	0	0	0	0	0	0	0	0	0	0
	0x0000 / 0x0400[1]															

1. In CONTROLREG2, the MODE bit is set to 1.

Table 16-6. SPI 1 Control Register and SPI 2 Control Register Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
DATARATE Bits 15–13	**Data Rate**—Selects the divide value of SCLK from the PERCLK2 in the PLL and Clock Control Module.	000 = Divide by 4 001 = Divide by 8 010 = Divide by 16 011 = Divide by 32 100 = Divide by 64 101 = Divide by 128 110 = Divide by 256 111 = Divide by 512
DRCTL Bits 12–11	**$\overline{SPI_RDY}$ Control**—Selects the waveform of the $\overline{SPI_RDY}$ input signal when the SPI 1 module operates in master mode. In slave mode, DRCTL is ignored. Because the SPI 2 module does not support the $\overline{SPI_RDY}$ control signal function, DRCTL must be written with 00 in CONTROLREG2.	00 = Ignore $\overline{SPI_RDY}$ 01 = Falling edge triggers input 10 = Active low level triggers input 11 = Reserved
MODE Bit 10	**SPI Mode Select**—Selects the mode for the SPI 1 module. In CONTROLREG2, MODE is set by the hardware.	0 = Slave mode 1 = Master mode

Table 16-6. SPI 1 Control Register and SPI 2 Control Register Description (continued)

Name	Description	Settings
SPIEN Bit 9	**SPI Module Enable**—Enables/Disables the serial peripheral interface. SPIEN must be asserted before an exchange is initiated. Writing 0 to SPIEN flushes the receive and transmit FIFOs.	0 = Disable the SPI 1 = Enable the SPI
XCH Bit 8	**Exchange**—Initiates a data exchange in master mode. XCH remains set while the exchange is in progress, or while the SPI module is waiting for an active $\overline{\text{SPI_RDY}}$ control signal input. XCH is automatically cleared when all data in the TXFIFO and shift register are shifted out. In slave mode, XCH must be cleared.	0 = Idle 1 = Initiates exchange (write) or busy (read)
SSPOL Bit 7	**$\overline{\text{SS}}$ Polarity Select**—Selects the polarity of the $\overline{\text{SS}}$ signal (in both master and slave mode).	0 = Active low 1 = Active high
SSCTL Bit 6	**$\overline{\text{SS}}$ Wave Form Select**—Selects the output wave form for the $\overline{\text{SS}}$ signal when in master mode. Controls RXFIFO advancement when in slave mode for SPI 1 only.	*In master mode:* 0 = SS stays low between SPI bursts 1 = Insert pulse between SPI bursts *In slave mode:* 0 = RXFIFO advanced by BIT_COUNT 1 = RXFIFO advanced by $\overline{\text{SS}}$ rising edge
PHA Bit 5	**Phase**—Controls the clock/data phase relationship (see Figure 16-2 on page 16-3).	0 = Phase 0 operation 1 = Phase 1 operation
POL Bit 4	**Polarity**—Controls the polarity of the SCLK signal (see Figure 16-2 on page 16-3).	0 = Active high polarity (0 = idle) 1 = Active low polarity (1 = idle)
BIT_COUNT Bits 3–0	**Bit Count**—This field selects the length of the transfer. A maximum of 16bits can be transferred. *In master mode*, a 16-bit data word is loaded from TxFIFO to shift register and only the least n bits (n=BIT COUNT) are shifted out. The next 16-bit word is then loaded to shift register. *In slave mode* and when SSCTL bit is 0 this field controls the number of bits received as a data word loaded to RxFIFO. When SSCTL bit is 1 and SS rising edge is faster than BIT COUNT is treated, then this field is don't care.	0000 = 1–bit transfer 0001 = 2–bit transfer ... 1111 = 16–bit transfer

16.3.4 Interrupt Control/Status Registers

The SPI interrupt control status registers allow the user to enable various interrupt signals and monitor the status of those interrupts.

		Addr
INTREG1	SPI 1 Interrupt Control/Status Register	**0x0021300C**
INTREG2	SPI 2 Interrupt Control/Status Register	**0x0021900C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	BOEN	ROEN	RFEN	RHEN	RREN	TFEN	THEN	TEEN	BO	RO	RF	RH	RR	TF	TH	TE
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 16-7. SPI 1 Interrupt Control/Status Register and SPI 2 Interrupt Control/Status Register Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
BOEN Bit 15	**Bit Count Overflow Interrupt Enable**—Enables/Disables the Bit Count Overflow Interrupt.	0 = Disable interrupt 1 = Enable interrupt
ROEN BIT 14	**RXFIFO Overflow Interrupt Enable**—Enables/Disables the RXFIFO Overflow Interrupt.	0 = Disable interrupt 1 = Enable interrupt
RFEN Bit 13	**RXFIFO Full Interrupt Enable**—Enables/Disables the RXFIFO Full Interrupt.	0 = Disable interrupt 1 = Enable interrupt
RHEN Bit 12	**RXFIFO Half Interrupt Enable**—Enables/Disables the RXFIFO Half-Full Interrupt.	0 = Disable interrupt 1 = Enable interrupt
RREN Bit 11	**RXFIFO Data Ready Interrupt Enable**—Enables/Disables the RXFIFO Data Ready Interrupt.	0 = Disable interrupt 1 = Enable interrupt
TFEN Bit 10	**TXFIFO Full Interrupt Enable**—Enables/Disables the TXFIFO Full Interrupt.	0 = Disable interrupt 1 = Enable interrupt
THEN Bit 9	**TXFIFO Half Interrupt Enable**—Enables/Disables the TXFIFO Half-Empty Interrupt.	0 = Disable interrupt 1 = Enable interrupt

Table 16-7. SPI 1 Interrupt Control/Status Register and SPI 2 Interrupt Control/Status Register Description (continued)

Name	Description	Settings
TEEN Bit 8	**TXFIFO Empty Interrupt Enable**—Enables/Disables the TXFIFO Empty Interrupt.	0 = Disable interrupt 1 = Enable interrupt
BO Bit 7	**Bit Count Overflow**—Indicates that a bit count overflow has occurred. BO is applicable for the SPI 1 module only when the bits of CONTROLREG1 are set so that MODE = 0 and SSCTL = 1. The overflow occurs when the slave receives more than 16 bits in one burst. BO is cleared after a data read from the RXDATAREG1 register. There is no way to determine which data word overflowed, so the bad data word can still be in the FIFO if it is not empty.	0 = No bit count overflow error 1 = At least one data word in RXFIFO has a bit count overflow error
RO Bit 6	**RXFIFO Overflow**—Indicates that the RXFIFO has overflowed. At least one newly written data word has been lost. The RO flag is automatically cleared after a data read.	0 = No RXFIFO overflow error 1 = At least one data word in the RXFIFO has been overwritten
RF Bit 5	**RXFIFO Full Status**—Indicates that the RXFIFO is full.	0 = Less than 8 data words are in the RXFIFO 1 = 8 data words are in the RXFIFO
RH Bit 4	**RXFIFO Half Status**—Indicates that the RXFIFO is at least half-full.	0 = Less than 4 data words are in the RXFIFO 1 = At least 4 data words are in the RXFIFO
RR Bit 3	**RXFIFO Data Ready Status**—Indicates that the RXFIFO is empty.	0 = The RXFIFO is empty 1 = At least one data word is in the RXFIFO
TF Bit 2	**TXFIFO Full Status**—Indicates that the TXFIFO is full.	0 = Less than 8 data words are in the TXFIFO 1 = 8 data words are in the TXFIFO
TH Bit 1	**TXFIFO Half Status**—Indicates that the TXFIFO is at least half-empty.	0 = Less than 4 empty slots are in the TXFIFO 1 = At least 4 empty slots are in the TXFIFO
TE Bit 0	**TXFIFO Empty Status**—Indicates that the TXFIFO is empty.	0 = At least one data word is in the TXFIFO 1 = The TXFIFO is empty, however data shifting may still be on-going. To be sure no data transaction is on-going, check the XCH bit(s) in the Control Register(s).

16.3.5 Test Registers

The SPI test registers allow the user to internally connect the receive and transmit sections, display the status of the state machine, and monitor the contents of the receive and transmit FIFOs.

		Addr
TESTREG1	SPI 1 Test Register	**0x00213010**
TESTREG2	SPI 2 Test Register	**0x00219010**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		LBC			SSTATUS				RXCNT				TXCNT			
TYPE	r	rw	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 16-8. SPI 1 Test Register and SPI 2 Test Register Description

Name	Description	Settings
Reserved Bits 31–15	Reserved—These bits are reserved and should read 0.	
LBC Bit 14	**Loop Back Control**—Internally connects the receive and transmit sections internally for test purposes.	0 = RX and TX sections are not internally connected 1 = RX and TX sections are internally connected
Reserved Bits 13–12	Reserved—These bits are reserved and should read 0.	
SSTATUS Bits 11–8	**State Machine Status**—Indicates the state machine status (used for test purposes only).	
RXCNT Bits 7–4	**RXFIFO Counter**—Indicates the number of data words in the RXFIFO.	0000 = RXFIFO is empty 0001 = 1 data word in RXFIFO 0010 = 2 data words in RXFIFO 0011 = 3 data words in RXFIFO 0100 = 4 data words in RXFIFO 0101 = 5 data words in RXFIFO 0110 = 6 data words in RXFIFO 0111 = 7 data words in RXFIFO 1000 = 8 data words in RXFIFO

Table 16-8. SPI 1 Test Register and SPI 2 Test Register Description (continued)

Name	Description	Settings
TXCNT Bits 3–0	**TXFIFO Counter**—Indicates the number of data words in the TXFIFO.	0000 = TXFIFO is empty 0001 = 1 data word in TXFIFO 0010 = 2 data words in TXFIFO 0011 = 3 data words in TXFIFO 0100 = 4 data words in TXFIFO 0101 = 5 data words in TXFIFO 0110 = 6 data words in TXFIFO 0111 = 7 data words in TXFIFO 1000 = 8 data words in TXFIFO

16.3.6 Sample Period Control Registers

The SPI sample period control registers allow the user to select the clock source for the counter and to set the wait between data transactions. The wait is only applicable when the SPI module is operating in master mode.

		Addr
PERIODREG1	SPI 1 Sample Period Control Register	**0x00213014**
PERIODREG2	SPI 2 Sample Period Control Register	**0x00219014**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CSRC	WAIT														
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 16-9. SPI 1 Sample Period Control Register and SPI 2 Sample Period Control Register Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
CSRC Bit 15	**Clock Source**—Selects the clock source for the counter.	0 = Bit clock 1 = 32.768 kHz or 32 kHz clock

Table 16-9. SPI 1 Sample Period Control Register and SPI 2 Sample Period Control Register Description (continued)

Name	Description	Settings
WAIT Bits 14–0	**Wait**—Determines the number of clocks inserted between data transactions (when operating in master mode).	0x0000 = 0 clock 0x0001 = 1 clock 0x0002 = 2 clocks … 0x7FFF = 32,767 clocks

16.3.7 DMA Control Registers

The SPI DMA control registers allow the user to enable DMA requests when the FIFOs are full, empty, or half-full. These registers also contain status bits for FIFO full, empty, and half-empty conditions.

		Addr
DMAREG1	SPI 1 DMA Control Register	**0x00213018**
DMAREG2	SPI 2 DMA Control Register	**0x00219018**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	THDEN	TEDEN	RFDEN	RHDEN					THDMA	TEDMA	RFDMA	RHDMA				
TYPE	rw	rw	rw	rw	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 16-10. SPI 1 DMA Control Register and SPI 2 DMA Control Register Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
THDEN Bit 15	**THDEN**—Enables/Disables the TXFIFO Half DMA Request.	0 = Disabled 1 = Enabled
TEDEN Bit 14	**TEDEN**—Enables/Disables the TXFIFO Empty DMA Request.	0 = Disabled 1 = Enabled
RFDEN Bit 13	**RFDEN**—Enables/Disables the RXFIFO Full DMA Request.	0 = Disabled 1 = Enabled

Table 16-10. SPI 1 DMA Control Register and SPI 2 DMA Control Register Description (continued)

Name	Description	Settings
RHDEN Bit 12	**RHDEN**—Enables/Disables the RXFIFO Half DMA Request.	0 = Disabled 1 = Enabled
Reserved Bits 11–8	Reserved—These bits are reserved and should read 0.	
THDMA Bit 7	**TXFIFO Half Status**—Indicates when the transmit FIFO is half-empty.	0 = There are less than 4 empty slots in the TXFIFO 1 = There are at least 4 empty slots in the TXFIFO
TEDMA Bit 6	**TXFIFO Empty Status**—Indicates when the transmit FIFO is empty.	0 = There is at least one data word in the TXFIFO 1 = The TXFIFO is empty, however data shifting may still be on-going. To be sure no data transaction is on-going, read the XCH bit in the Control Registers.
RFDMA Bit 5	**RXFIFO Full Status**—Indicates when the receive FIFO is full.	0 = There are less than 8 data words in the RXFIFO 1 = There are 8 data words in the RXFIFO
RHDMA Bit 4	**RXFIFO Half Status**—Indicates when the receive FIFO is half-full.	0 = There are less than 4 data words in the RXFIFO 1 = There are at least 4 data words in the RXFIFO
Reserved Bits 3–0	Reserved—These bits are reserved and should read 0.	

16.3.8 Soft Reset Registers

The SPI soft reset registers allow the user to reset the SPI module.

		Addr
RESETREG1	SPI 1 Soft Reset Register	**0x0021301C**
RESETREG2	SPI 2 Soft Reset Register	**0x0021901C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																START
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 16-11. SPI 1 Soft Reset Register and SPI 2 Soft Reset Register Description

Name	Description	Settings
Reserved Bits 31–1	Reserved—These bits are reserved and should read 0.	
START Bit 0	**Start**—Executes soft reset.	0 = No soft reset 1 = Soft reset

Chapter 17
LCD Controller

17.1 Introduction

The Liquid Crystal Display Controller (LCDC) provides display data for external gray-scale or color LCD panels. The LCDC is capable of supporting black-and-white, gray-scale, passive-matrix color, and active-matrix color LCD panels.

17.2 Features

- Support for single (non-split) screen monochrome/color LCD panels and self-refresh type LCD panels
- 16 simultaneous gray-scale levels from a palette of 16 for monochrome display
- Support for:
  - 4/8/12 bits per pixel (bpp) for passive color panel
  - 4/8/12/16 bpp for TFT panel
  - Up to 256 colors out of a palette of 4096 colors for an 8 bpp display and 4096 colors for a 12 bpp display
  - True 64K colors for 16 bpp
  - Additional support details are shown in Table 17-1

Table 17-1. Supported Panel Characteristics

Panel Type	Bit/Pixel	Panel Interface (Bits)	Number of Gray Level/Color
Monochrome	1	1, 2, 4, 8	Black-and-white
	2	1, 2, 4, 8	4
	4	1, 2, 4, 8	16
CSTN	4,8,12	8	16, 256, 4096
TFT	4, 8	16	16, 256
	12, 16	12, 16	4096, 64K

- Standard panel interface for common LCD drivers
- Panel interface of 16-, 12-, 8-, 4-, 2-, and 1-bit-wide LCD panel data bus for monochrome or color panels
- For 4 bpp and 8 bpp a palette table is used for re-mapping of data from memory, independent of type of panel used. For the 1 bpp, 2 bpp, 12 bpp and 16 bpp the palette table is by-passed.

- Direct interface to active color panels (TFT) such as NEC and Sharp
- Dedicated signals facilitate interface to Sharp HR-TFT panels such as 320 × 240
- Hardware-generated cursor with blink, color, and size programmability
- Logical operation between color hardware cursor and background
- Hardware panning (soft horizontal scrolling)
- 8-bit pulse-width modulator for software contrast control

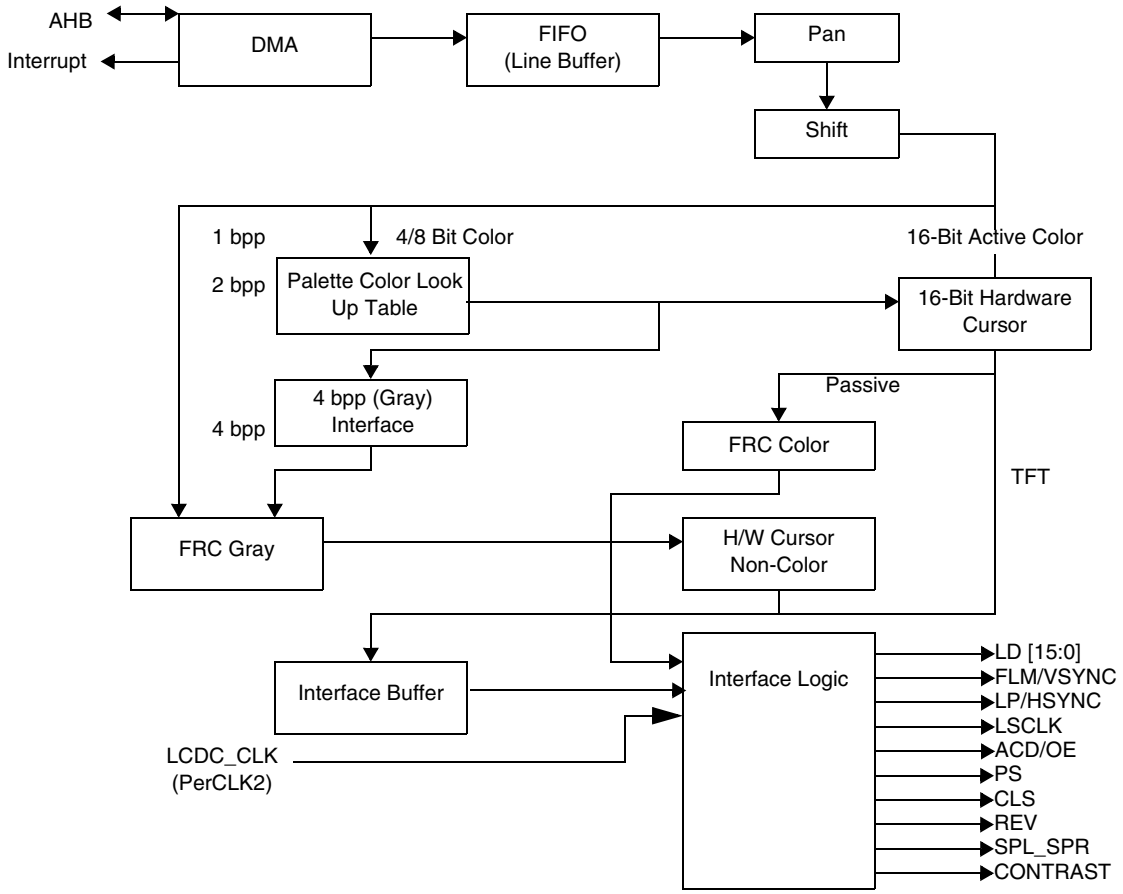


Figure 17-1. LCDC Block Diagram

17.3 LCDC Operation

17.3.1 LCD Screen Format

The number of pixels forming the screen width and screen height of the LCD panel are software programmable. Figure 17-2 shows the relationship between the screen size and memory window.

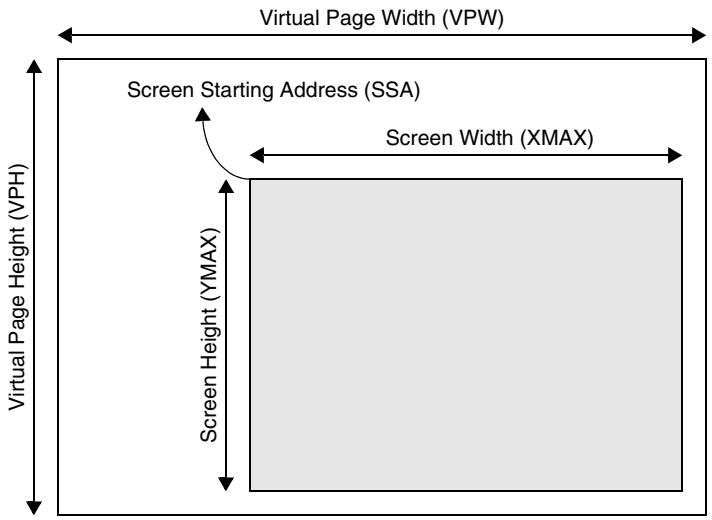


Figure 17-2. LCD Screen Format

The Screen Width (XMAX) and Screen Height (YMAX) parameters specify the LCD panel size. The LCDC will start scanning the display memory at the location pointed to by the Screen Starting Address (SSA) register, represented by the shaded area in Figure 17-2, for display on the LCD panel.

The maximum page width is specified by the Virtual Page Width (VPW) parameter. Virtual Page Height (VPH) does not affect the LCDC and is limited only by memory size. By changing the SSA register, a screen-sized window can be vertically or horizontally scrolled (panned) anywhere inside the virtual page boundaries. The software must control the starting address in the SSA properly so that the scanning logic's System Memory Pointer (SMP) stays within the VPW and VPH limits to prevent the display of strange artifacts on the screen.

VPH is used by the programmer only for boundary checks. There is no VPH parameter internal to the LCDC.

VPW is used in calculating the RAM starting address representing the beginning of each displayed line. SSA sets the address of data for the first line of a frame. For each subsequent line, VPW is added to an accumulation initialized by the SSA to yield the starting address of that line.

17.3.2 Panning

Panning Offset (POS) is expressed in bits, not pixels, so when operating in any mode other than 1 bpp, only even pixel boundaries are valid. In 12 bpp mode, the pixels are aligned to 16-bit boundaries, and POS also must align to these boundaries.

SSA and POS are located in isolated registers and are double buffered because they are dynamic parameters likely to change while the LCDC is running. New values of SSA and POS do not take effect until the beginning of the next frame. A typical panning algorithm includes an interrupt at the beginning of the frame. In the interrupt service routine, POS and/or SSA are updated (the old values are internally latched). The updates take effect on the next frame.

17.3.3 Display Data Mapping

The LCDC supports 1/2/4 bpp in monochrome mode and 4/8/12/16 bpp in color mode. System memory data mapping in 2/4/8/12/16 bpp modes is shown in Figure 17-4 and in Figure 17-5.

NOTE:

In 12 bpp mode, 16 bits of memory are used for each set of 12 bits, leaving 4 bits unused. In 16 bpp mode, all 16 bits are used. Refer to Figure 17-5 and Table 17-7.

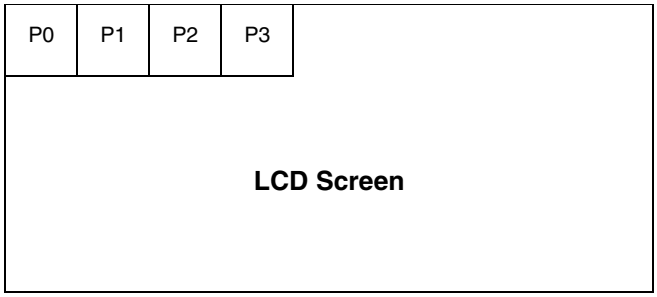


Figure 17-3. Pixel Location on Display Screen

1 bpp Mode, Little Endian

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
	P24	P25	P26	P27	P28	P29	P30	P31
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	P16	P17	P18	P19	P20	P21	P22	P23
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	P8	P9	P10	P11	P12	P13	P14	P15
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	P0	P1	P2	P3	P4	P5	P6	P7

1 bpp Mode, Big Endian

Byte Address	Sample Bit-to-Pixel Mapping							
0	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
	P0	P1	P2	P3	P4	P5	P6	P7
1	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	P8	P9	P10	P11	P12	P13	P14	P15
2	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	P16	P17	P18	P19	P20	P21	P22	P23
3	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	P24	P25	P26	P27	P28	P29	P30	P31

2 bpp Mode, Little Endian

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
	P12		P13		P14		P15	
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	P8		P9		P10		P11	
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	P4		P5		P6		P7	
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	P0		P1		P2		P3	

2 bpp Mode, Big Endian

Byte Address	Sample Bit-to-Pixel Mapping							
0	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
	P0		P1		P2		P3	
1	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	P4		P5		P6		P7	
2	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	P8		P9		P10		P11	
3	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	P12		P13		P14		P15	

4 bpp Mode, Little Endian

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
	P6				P7			
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	P4				P5			
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	P2				P3			
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	P0				P1			

4 bpp Mode, Big Endian

Byte Address	Sample Bit-to-Pixel Mapping							
0	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
	P0				P1			
1	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	P2				P3			
2	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	P4				P5			
3	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	P6				P7			

8 bpp Mode, Little Endian

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
	P3							
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	P2							
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	P1							
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	P0							

8 bpp Mode, Big Endian

Byte Address	Sample Bit-to-Pixel Mapping							
0	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
	P0							
1	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	P1							
2	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	P2							
3	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	P3							

P0 = Red0Green0Blue0 **P1 = Red1Green1Blue1**

Figure 17-4. Display Data Mapping, 1/2/4/8 bpp Modes

Table 17-2. Display Mapping in 12 bpp, CSTN Panel, Little Endian

16 bpp Mode, Little Endian

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
	Red1 [4]	Red1 [3]	Red1 [2]	Red1 [1]	Red1 [0]	Green1 [5]	Green1 [4]	Green1 [3]
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	Green1 [2]	Green1 [1]	Green1 [0]	Blue1 [4]	Blue1 [3]	Blue1 [2]	Blue1 [1]	Blue1 [0]
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	Red0 [4]	Red0 [3]	Red0 [2]	Red0 [1]	Red0 [0]	Green0 [5]	Green0 [4]	Green0 [3]
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Green0 [2]	Green0 [1]	Green0 [0]	Blue0 [4]	Blue0 [3]	Blue0 [2]	Blue0 [1]	Blue0 [0]

16 bpp Mode, Big Endian

Byte Address	Sample Bit-to-Pixel Mapping							
0	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
	Red0 [4]	Red0 [3]	Red0 [2]	Red0 [1]	Red0 [0]	Green0 [5]	Green0 [4]	Green0 [3]
1	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	Green0 [2]	Green0 [1]	Green0 [0]	Blue0 [4]	Blue0 [3]	Blue0 [2]	Blue0 [1]	Blue0 [0]
2	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	Red1 [4]	Red1 [3]	Red1 [2]	Red1 [1]	Red1 [0]	Green1 [5]	Green1 [4]	Green1 [3]
3	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Green1 [2]	Green1 [1]	Green1 [0]	Blue1 [4]	Blue1 [3]	Blue1 [2]	Blue1 [1]	Blue1 [0]

Table 17-3. Display Mapping in 12 bpp, CSTN Panel, Little Endian

Byte Address	Bit-to-Pixel Mapping							
3	B31	B30	B29	B28	B27	B26	B25	B24
	–	–	–	–	$R1^3$	$R1^2$	$R1^1$	RI^0
2	B23	B22	B21	B20	B19	B18	B17	B16
	$G1^3$	$G1^2$	$G1^1$	GI^0	$B1^3$	$B1^2$	$B1^1$	BI^0
1	B15	B14	B13	B12	B11	B10	B9	B8
	–	–	–	–	$R0^3$	$R0^2$	$R0^1$	RO^0
0	B7	B6	B5	B4	B3	B2	B1	BO
	GO3	$G0^2$	$G0^1$	GO^0	BO3	$B0^2$	$B0^1$	BO^0

PO = RGBo P1= RGB_1

Table 17-4. Display Mapping in 12 bpp, CSTN Panel, Big Endian

Byte Address	Bit-to-Pixel Mapping							
0	B31	B30	B29	B28	B27	B26	B25	B24
	–	–	–	–	$R0^3$	$R0^2$	$R0^1$	RO^0
1	B23	B22	B21	B20	B19	B18	B17	B16
	GO3	$G0^2$	$G0^1$	GO^0	BO3	$B0^2$	$B0^1$	BO^0
2	B15	B14	B13	B12	B11	B10	B9	B8
	–	–	–	–	$R1^3$	$R1^2$	$R1^1$	RI^0
3	B7	B6	B5	B4	B3	B2	B1	BO
	$G1^3$	$G1^2$	$G1^1$	GI^0	$B1^3$	$B1^2$	$B1^1$	BI^0

PO = RGBo P1= RGB_1

Figure 17-5. Display Data Mapping, 16 bpp Mode

17.3.4 Black-and-White Operation

The 1 bpp mode is also known as black-and-white mode because each pixel is always either fully on or fully off.

17.3.5 Gray-Scale Operation

The LCDC generates a maximum of 16 gray levels. These gray levels are defined by 2 or 4 bits of display data for each pixel. Using 2 bpp, the LCDC displays 4 shades of gray, and using 4 bpp, the LCDC displays all 16 shades. The shades of gray are obtained by controlling the number of frames in which the pixel is “on” over a period of 16 frames. This method is known as Frame Rate Control (FRC). For more information on FRC, see Section 17.3.7, “Frame Rate Modulation Control (FRC).”

The use of the mapping RAM is shown in Figure 17-6. When using 2 bpp, the 2-bit code is mapped to one of 4 gray levels, and when using 4 bpp, the 4-bit code is mapped to one of 16 gray levels. Because crystal formulations and driving voltages vary, the visual gray effect may or may not be linearly related to the frame rate. A logarithmic scale such as 0, 1/4, 1/2 and 1 might be more pleasing than a linearly spaced scale such as 0, 5/16, 11/16 and 1 for certain graphics.

Figure 17-6 illustrates gray-scale pixel generation. The flexible mapping scheme allows the user to optimize the visual effect for a specific panel or application.

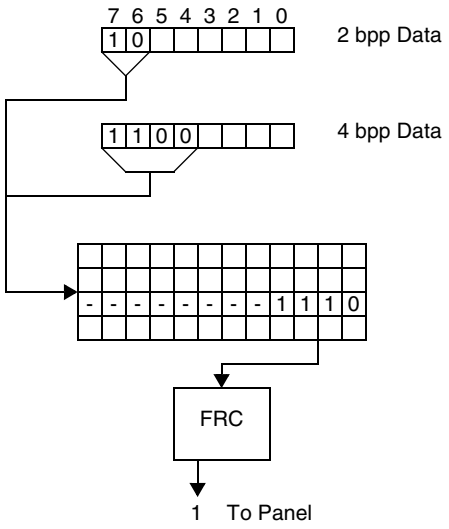


Figure 17-6. Gray-Scale Pixel Generation

17.3.6 Color Generation

The value corresponding to each color pixel on the screen is represented by a 4-, 8-, 12- or 16-bit code in the display memory.

For the 4- and 8- bit modes use the LCDC's color mapping RAM to map the data to a 12-bit RGB code. For passive matrix color displays, 4-bit and 8-bit mode, the 12-bit RGB code from the mapping RAM is output to the FRC blocks that independently process the code corresponding to the red, green, and blue components of each pixel to generate the required shade and intensity.

For active matrix display, the 12-bit output from the mapping RAM is output to the panel.

For 12-bit mode for passive matrix color display, the mapping RAM is by-passed and output directly to the FRC block.

In 16-bit mode, pixel data is simply moved from display memory to the 16-bit LCDC output bus.

For active matrix displays, the 16-bit RGB code from the mapping RAM is output to the panel. For passive color display, the maximum color depth is 12-bit and 16-bit color is not supported.

Figure 17-7 and Figure 17-8 on page 17-10 illustrate passive matrix and active matrix color pixel generation.

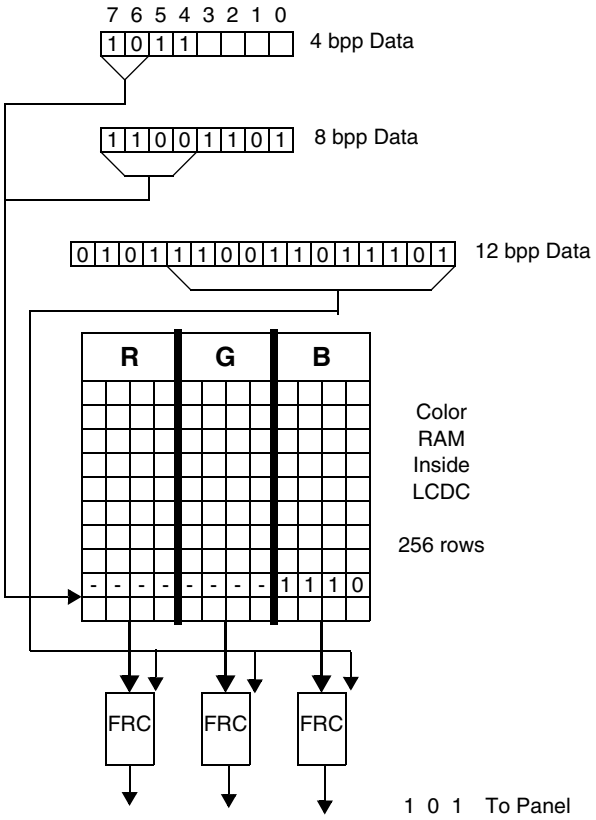


Figure 17-7. Passive Matrix Color Pixel Generation

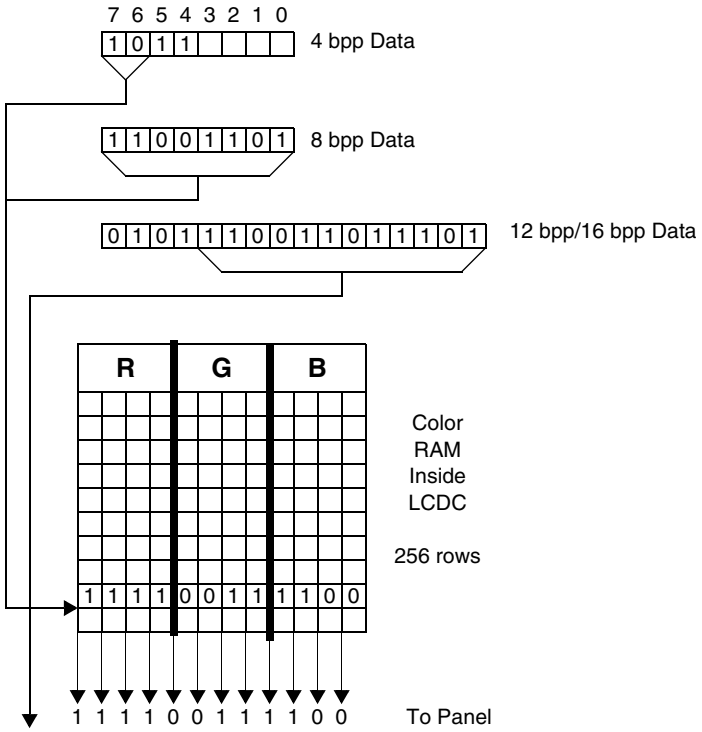


Figure 17-8. Active Matrix Color Pixel Generation

17.3.7 Frame Rate Modulation Control (FRC)

Circuitry inside the LCDC generates intermediate gray-scale colors on the panel by adjusting the density of zeroes and ones that appear over the frames. The LCDC can generate 16 simultaneous gray-scale levels.

Table 17-5. Gray Palette Density

Gray Code (Hexadecimal)	Density	Density (Decimal)
0	0	0
1	1/8	0.125
2	1/5	0.2
3	1/4	0.25
4	1/3	$0.\overline{333}$
5	2/5	0.4
6	4/9	$0.\overline{444}$
7	1/2	0.5
8	5/9	$0.\overline{555}$
9	3/5	0.6
A	2/3	$0.\overline{666}$

Table 17-5. Gray Palette Density (continued)

Gray Code (Hexadecimal)	Density	Density (Decimal)
B	3/4	0.75
C	4/5	0.8
D	7/8	0.875
E	14/15	$0.9\overline{33}$
F	1	1
Note: Overbars indicate repeating decimal numbers.		

17.3.8 Panel Interface Signals and Timing

The LCDC continuously provides pixel data to the LCD panel via the LCD panel interface. Panel interface signals are illustrated in Figure 17-9 on page 17-11.

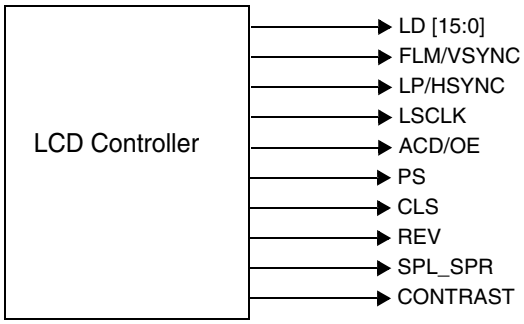


Figure 17-9. LCDC Interface Signals

The format, timing, and polarity of the panel interface signals are programmable. There are two basic modes, passive and active, selected by the TFT register bit. The user must also select either grayscale mode or color mode.

SPL_SPR, PS, CLS, and REV are additional interface signals required for Sharp HR-TFT panels.

17.3.8.1 Pin Configuration for LCDC

Figure 17-9 shows the signals used for the LCDC. These pins are multiplexed with other functions on the device, and must be configured for LCDC operation before they can be used.

NOTE:

The user must ensure that the data direction bits in the GPIO are set to the correct direction for proper operation. See Section 29.5.1, “Data Direction Registers,” on page 29-8 for details.

Table 17-6. Pin Configuration

Pin	Setting	Configuration Procedure
LD [15:0]	Primary function of GPIO Port D [30:15]	1. Clear bits [30:15] of Port D GPIO In Use Register (GIUS_D) 2. Clear bits [30:15] of Port D General Purpose Register (GPR_D)
FLM/VSYNC	Primary function of GPIO Port D [14]	1. Clear bit 14 of Port D GPIO In Use Register (GIUS_D) 2. Clear bit 14 of Port D General Purpose Register (GPR_D)
LP/HSYNC	Primary function of GPIO Port D [13]	1. Clear bit 13 of Port D GPIO In Use Register (GIUS_D) 2. Clear bit 13 of Port D General Purpose Register (GPR_D)
LSCLK	Primary function of GPIO Port D [6]	1. Clear bit 6 of Port D GPIO In Use Register (GIUS_D) 2. Clear bit 6 of Port D General Purpose Register (GPR_D)
ACD/OE	Primary function of GPIO Port D [12]	1. Clear bit 12 of Port D GPIO In Use Register (GIUS_D) 2. Clear bit 12 of Port D General Purpose Register (GPR_D)
CONTRAST	Primary function of GPIO Port D [11]	1. Clear bit 11 of Port D GPIO In Use Register (GIUS_D) 2. Clear bit 11 of Port D General Purpose Register (GPR_D)
SPL_SPR	Primary function of GPIO Port D [10]	1. Clear bit 10 of Port D GPIO In Use Register (GIUS_D) 2. Clear bit 10 of Port D General Purpose Register (GPR_D)
PS	Primary function of GPIO Port D [9]	1. Clear bit 9 of Port D GPIO In Use Register (GIUS_D) 2. Clear bit 9 of Port D General Purpose Register (GPR_D)
CLS	Primary function of GPIO Port D [8]	1. Clear bit 8 of Port D GPIO In Use Register (GIUS_D) 2. Clear bit 8 of Port D General Purpose Register (GPR_D)
REV	Primary function of GPIO Port D [7]	1. Clear bit 7 of Port D GPIO In Use Register (GIUS_D) 2. Clear bit 7 of Port D General Purpose Register (GPR_D)

17.3.8.2 Passive Matrix Panel Interface Signals

Figure 17-10 shows the LCD interface timing for monochrome panels and Figure 17-11 shows the LCD interface timing for passive matrix color panels. Signal polarities are shown positive, however it can be reversed by clearing the bits in the Panel Configuration Register (PCR). The data bus timing for passive panels is controlled by the shift clock (LSCLK), line pulse (LP), first line marker (FLM), alternate crystal direction (ACD), and line data (LD) signals. Operation of the panel interface is accomplished in the following steps:

1. LSCLK clocks the pixel data into the display driver's internal shift register.
2. LP signifies the end of the current line of serial data and latches the shifted pixel data into a wide latch.
3. FLM marks the first line of the displayed page. The LD (and the associated LP), enclosed by the FLM signal, marks the first line of the current frame.
4. ACD toggles after a pre-programmed number of FLM pulses. This signal refreshes the LCD panel.

NOTE:

The LD bus width is programmable to 1, 2, 4, or 8 bits in monochrome mode (the COLOR bit in the Panel Configuration register is set to 0). Data is justified to the least significant bits of the LD [15:0] bus. Passive color displays use a fixed 2-2/3 pixels of data per 8-bit vector as shown in Figure 17-11 on page 17-13.

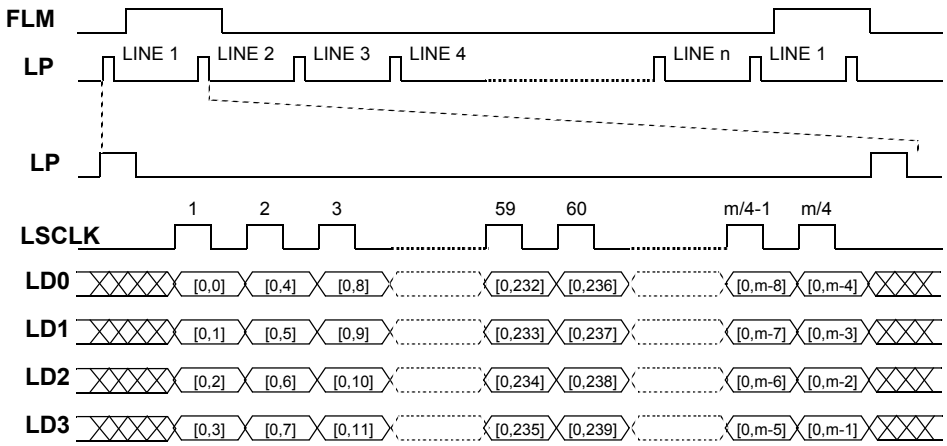


Figure 17-10. LCDC Interface Timing for 4-bit Data Width Gray-Scale Panels

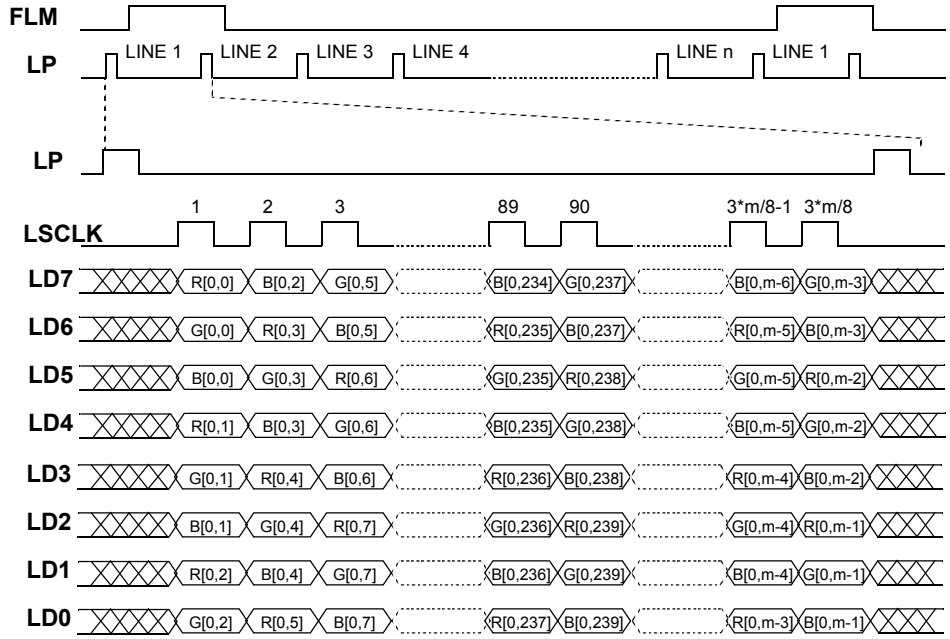


Figure 17-11. LCDC Interface Timing for 8-Bit Data Passive Matrix Color Panels

17.3.8.3 Passive Panel Interface Timing

Figure 17-12 on page 17-14 shows the horizontal timing (timing of one line), including both the line pulse (LP) and the data. The width of LP and delays both before and after LP are programmable. The parameters used for panel interface timing are:

- XMAX (X size) defines the number of pixels per line. XMAX is the total number of pixels per line.
- H_WAIT_1 defines the delay from the end of data output to the beginning of LP.
- H_WIDTH (horizontal sync pulse width) defines the width of the FLM pulse, and H_WIDTH must be at least 1.
- H_WAIT_2 defines the delay from the end of LP to the beginning of data output.

NOTE:

All parameters are defined in unit of pixel clock period, unless stated otherwise.

17.3.9 8 bpp Mode Color STN Panel

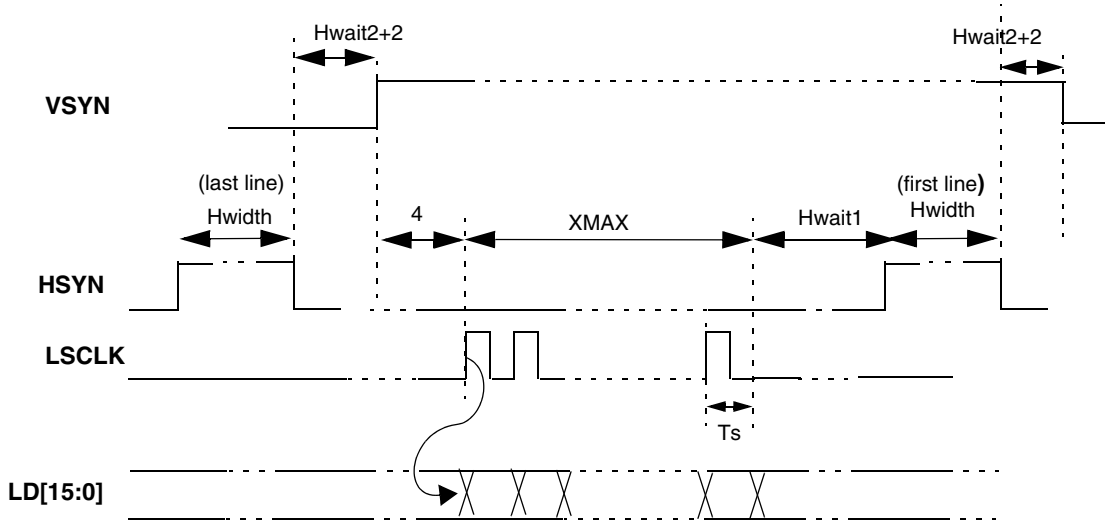


Figure 17-12. Horizontal Sync Pulse Timing in Passive Mode

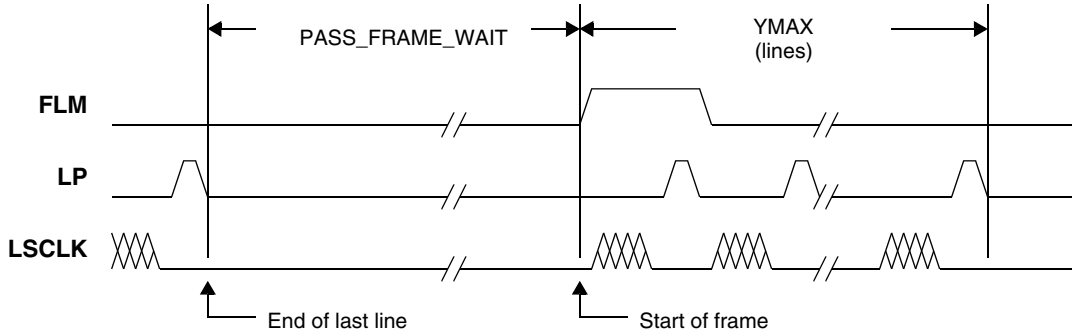


Figure 17-13. Vertical Sync Pulse Timing Passive, Color, (Non-TFT) Mode

17.3.9.1 Active Matrix Panel Interface Signals

Figure 17-14 on page 17-16 shows the LCD interface timing for an active matrix color TFT panel. In this figure signals are shown with negative polarity (FLMPOL=1, LPPOL=1, CLKPOL=0, OEPOL=1). In TFT mode, the LSCLK is automatically inverted. The panel interface timing for active matrix panels is sometimes referred to as a “digital CRT” and is controlled by the shift clock (LSCLK), horizontal sync pulse (HSYNC, the LP pin in passive mode), vertical sync pulse (VSYNC, the FLM pin in passive mode), output enable (OE, the ACD pin in passive mode), and line data (LD) signals. The sequence of events for active matrix interface timing is:

1. LSCLK latches data into the panel on its negative edge (when positive polarity is selected). In active mode, LSCLK runs continuously.

2. HSYNC causes the panel to start a new line.
3. VSYNC causes the panel to start a new frame. It always encompasses at least one HSYNC pulse.
4. OE functions as an output enable signal to the CRT. This output enable signal is similar to the blanking output in a CRT and enables the data to be shifted onto the display. When disabled, the data is invalid and the trace is off.

In 4- and 8-bit mode, the LD [15:12] bits define red, the LD [10:7] bits define green, and the LD [4:1] bits define blue. In 16-bit mode, the LD [15:11] bits define red, the LD [10:5] bits define green, and the LD [4:0] bits define blue.

The actual TFT color channel assignments are shown in Table 17-7. In 4 bpp and 8 bpp, bits LD11, LD6, LD5 and LD0 are fixed at 0.

Table 17-7. TFT Color Channel Assignments

	LD 15	LD 14	LD 13	LD 12	LD 11	LD 10	LD 9	LD 8	LD 7	LD 6	LD 5	LD 4	LD 3	LD 2	LD 1	LD 0
4 bpp	R3	R2	R1	R0	–	G3	G2	G1	G0	–	–	B3	B2	B1	B0	–
8 bpp	R3	R2	R1	R0	–	G3	G2	G1	G0	–	–	B3	B2	B1	B0	–
12 bpp	R3	R2	R1	R0	–	G3	G2	G1	G0	–	–	B3	B2	B1	B0	–
16 bpp	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0

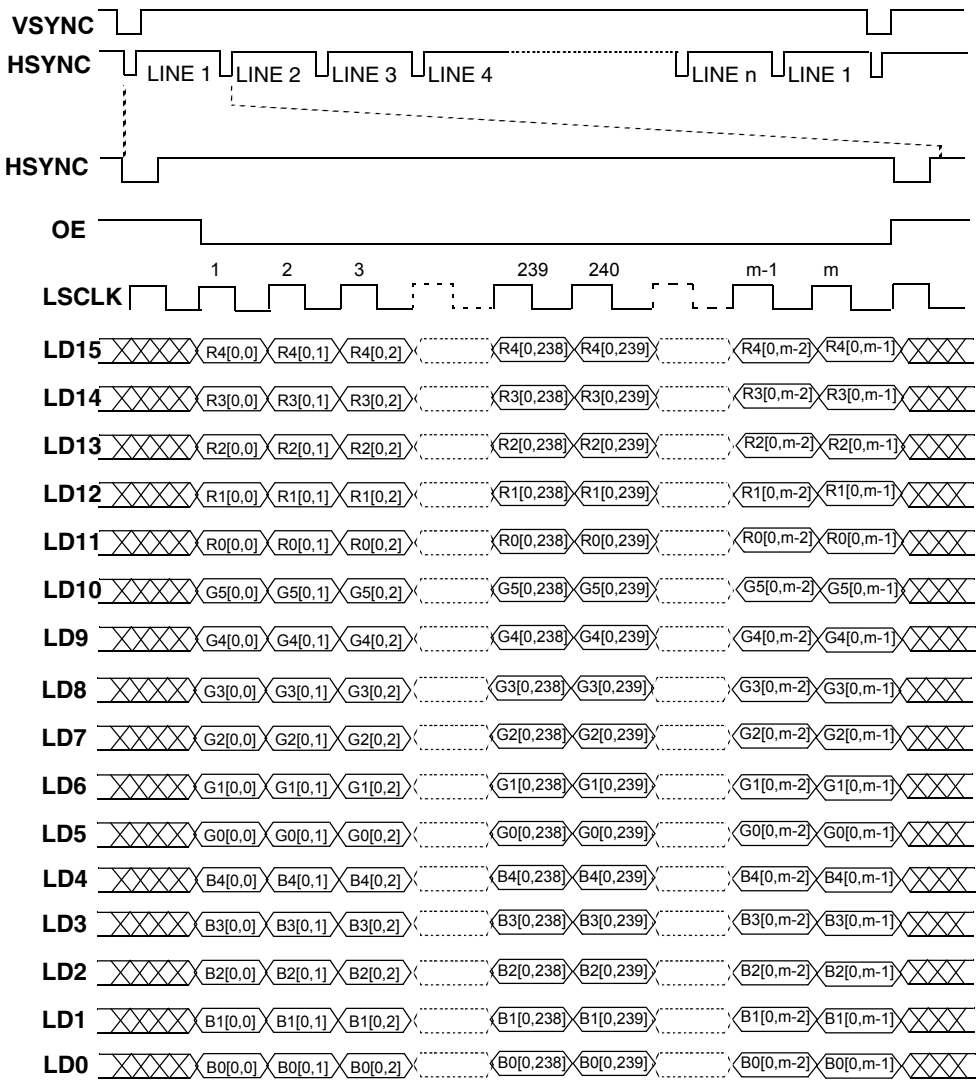


Figure 17-14. LCDC Interface Timing for Active Matrix Color Panels

17.3.9.2 Active Panel Interface Timing

Figure 17-15 on page 17-17 shows the horizontal timing (timing of one line), including both the horizontal sync pulse and the data. The width of HSYNC and delays both before and after HSYNC are programmable. The timing signal parameters are defined as follows:

- H_WIDTH defines the width of the HSYNC pulse and must be at least 1.
- H_WAIT_2 defines the delay from the end of HSYNC to the beginning of the OE pulse.
- H_WAIT_1 defines the delay from end of OE to the beginning of the HSYNC pulse.
- XMAX defines the (total) number of pixels per line.

NOTE:

All parameters are defined in pixel periods, not LSCLK periods.

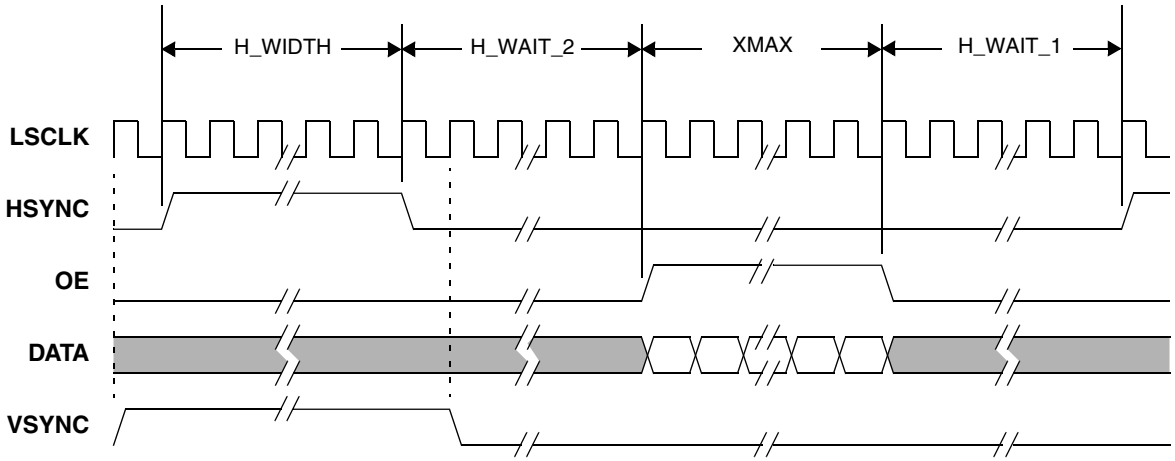


Figure 17-15. Horizontal Sync Pulse Timing in TFT Mode

Figure 17-16 shows the vertical timing (timing of one frame). The delay from the end of one frame until the beginning of the next is programmable. The memory timing signal parameters are:

- V_WAIT_1 is a delay measured in lines. For V_WAIT_1= 1 there is a delay of one HSYNC (time = one line period) before VSYNC. The HSYNC pulse is output during the V_WAIT_1 delay.
- For V_WIDTH (vertical sync pulse width) = 0, VSYNC encloses one HSYNC pulse. For V_WIDTH = 2, VSYNC encloses two HSYNC pulses.
- V_WAIT_2 is a delay measured in lines. For V_WAIT_2 = 1, there is a delay of one HSYNC (time = one line period) after VSYNC. The HSYNC pulse is output during the V_WAIT_2 delay.

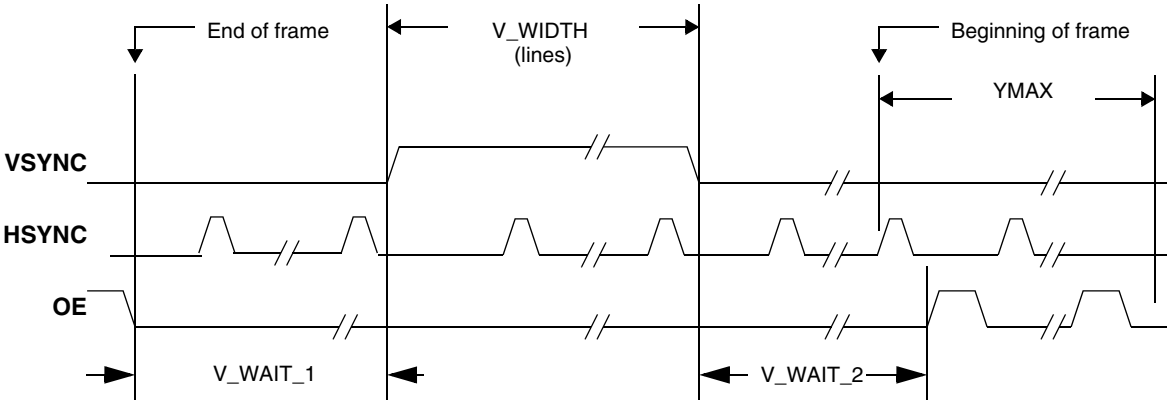


Figure 17-16. Vertical Sync Pulse Timing TFT Mode

17.4 Programming Model

The LCDC memory space contains fifteen 32-bit registers for display parameters, a read-only status register, and a 256 x 12 Color Mapping RAM. The color mapping RAM is physically located inside the Palette Lookup Table module.

Table 17-8 summarizes these registers and their addresses. Only WORD access is supported. Byte and halfword access is undefined.

Table 17-8. LCDC Register Memory Map

Description	Name	Address
Screen Start Address Register	SSA	0x00205000
Size Register	SIZE	0x00205004
Virtual Page Width Register	VPW	0x00205008
LCD Cursor Position Register	CPOS	0x0020500C
LCD Cursor Width Height and Blink Register	LCWHB	0x00205010
LCD Color Cursor Mapping Register	LCHCC	0x00205014
Panel Configuration Register	PCR	0x00205018
Horizontal Configuration Register	HCR	0x0020501C
Vertical Configuration Register	VCR	0x00205020
Panning Offset Register	POS	0x00205024
Sharp Configuration 1 Register	LSCR1	0x00205028
PWM Contrast Control Register	PWMR	0x0020502C
DMA Control Register	DMACR	0x00205030
Refresh Mode Control Register	RMCR	0x00205034
Interrupt Configuration Register	LCDICR	0x00205038
Interrupt Status Register	LCDISR	0x00205040

Figure 17-9 on page 17-19 provides a quick overview of the fields of all the registers. There are intentional gaps between the addresses for the read-write register section and the status register, and between the status register and the mapping RAM.

Table 17-9. Register Memory Mapping Summary

Register Name	Register Location	Register Bits															
		31 (15)	30 (14)	29 (13)	28 (12)	27 (11)	26 (10)	25 (9)	24 (8)	23 (7)	22 (6)	21 (5)	20 (4)	19 (3)	18 (2)	17 (1)	16 (0)
SSA	0x00205000	Screen Start Address - SSA															
		Screen Start Address - SSA															
SIZE	0x00205004							Screen Width - XMAX									
									Screen Height - YMAX								
VPW	0x00205008																
								Virtual Page Width - VPW									
CPOS	0x0020500C	CC1	CC0		OP			Cursor X Position - CXP									
									Cursor Y Position - CYP								
LCWHB	0x00205010	BK_EN			Cursor Width - CW								Cursor Height - CH				
										BD							
LCHCC	0x00205014																
		Cursor Red - CUR_COL_R					Cursor Green - CUR_COL_G						Cursor Blue - CUR_COL_B				
PCR	0x00205018	TFT	COLOR	Bus Width PBSIZ		Bits Per Pixel BPIX			PIX POL	FLM POL	LP POL	CLK POL	OE POL	SCLK IDLE	END_ SEL	END_ BYTE_ SWAP	REV _VS
		ACD SEL	Crystal Direction Toggle - ACD							SCLK SEL	SHARP	Pixel Clock Divider - PCD					
HCR	0x0020501C	Horizontal Sync Width - H_WIDTH															
		Horizontal Wait 1 - H_WAIT_1								Horizontal Wait 2 - H_WAIT_2							
VCR	0x00205020	Vertical Sync Width - V_WIDTH															
		Vertical Wait 1 - V_WAIT_1								Vertical Wait 2 - V_WAIT_2							
POS	0x00205024																
													Panning Offset - POS				
LSCR1	0x00205028	PS_RISE_DELAY								CLS_RISE_DELAY							
						REV_TOGGLE_DELAY				Gray 1				Gray 2			
PWMR	0x0020502C								CLS High Width								
		LD MSK					SCR1	SCR0	CC _EN	Pulse Width - PW							
DMACR	0x00205030	BURST												DMA High Mark - HM			
														DMA Trigger Mark - TM			
RMCR	0x00205034																
																LCDC _EN	SELF _REF
LCDICR	0x00205038																
															INT SYN		INT CON
LCDISR	0x00205040																
														UDR _ERR	ERR _RES	EOF	BOF
	0x00205800																
						First RAM Location(R [3:0], G [3:0], B [3:0])											
	0x00205BFC																
						Last RAM Location(R [3:0], G [3:0], B [3:0])											

17.4.1 Screen Start Address Register

The Screen Start Address Register specifies the start address of the LCD screen. See Figure 17-2.

SSA Screen Start Address Register **Addr 0x00205000**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SSA															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SSA															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 17-10. Screen Start Address Register Description

Name	Description
SSA Bits 31–2	**Screen Start Address of LCD Panel**—Holds pixel data for a new frame from the SSA address. This field must start at a location that enables a complete picture to be stored in a 4 Mbyte memory boundary (A [21:0]). A [31:22] has a fixed value for a picture's image.
Reserved Bits 1–0	Reserved—These bits are reserved and should read 0.

17.4.2 Size Register

The Size Register defines the height and width of the LCD screen.

SIZE Size Register **Addr 0x00205004**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
							XMAX									
TYPE	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								YMAX								
TYPE	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 17-11. Size Register Description

Name	Description
Reserved Bits 31–26	Reserved—These bits are reserved and should read 0.
XMAX Bits 25–20	**Screen Width Divided by 16**—Holds screen x-axis size, divided by 16. For black-and-white panels (1 bpp), XMAX [20] is ignored, forcing the x-axis of the screen size to be a multiple of 32 pixels/line.
Reserved Bits 19–9	Reserved—These bits are reserved and should read 0.
YMAX Bits 8–0	**Screen Height**—Specifies the height of the LCD panel in terms of pixels or lines. The lines are numbered from 1 to YMAX for a total of YMAX lines.

17.4.3 Virtual Page Width Register

The Virtual Page Width Register defines the width of the virtual page for the LCD panel. See Figure 17-2 on page 17-3.

VPW Virtual Page Width Register **Addr 0x00205008**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							VPW									
TYPE	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 17-12. Virtual Page Width Register Description

Name	Description
Reserved Bits 31–10	Reserved—These bits are reserved and should read 0.
VPW Bits 9–0	**Virtual Page Width**—Defines the virtual page width of the LCD panel. The VPW bits represent the number of 32-bit words required to hold the data for one virtual line. VPW is used in calculating the starting address representing the beginning of each displayed line.

17.4.4 Panel Configuration Register

The Panel Configuration Register defines all of the properties of the LCD screen.

PCR Panel Configuration Register **Addr 0x00205018**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TFT	COLOR	PBSIZ		BPIX			PIX POL	FLM POL	LP POL	CLK POL	OE POL	SCLK IDLE	END_ SEL	SWAP_SEL	REV_ VS
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
										0x0000						

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ACD SEL	ACD							SCLK SEL	SHARP	PCD					
TYPE	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
										0x0000						

Table 17-13. Panel Configuration Register Description

Name	Description	Settings
TFT Bit 31	**Interfaces to TFT Display**—Controls the format and timing of the output control signals. Active and passive displays use different signal timing formats as described in Section 17.3.8, “Panel Interface Signals and Timing.” TFT also controls the use of the FRC in color mode.	0 = The LCD panel is a passive display 1 = The LCD panel is an active display: “digital CRT” signal format, FRC is bypassed, LD bus width is fixed at 16-bits
COLOR Bit 30	**Interfaces to Color Display**—Activates three channels of FRC in passive mode to allow use of the special 2 2/3 pixels per output vector format.	0 = The LCD panel is a monochrome display 1 = The LCD panel is a color display **Note:** For a TFT panel, set COLOR=1
PBSIZ Bits 29–28	**Panel Bus Width**—Specifies the panel bus width. Applicable for monochrome or passive matrix color monitors. For active (TFT) color panels, the panel bus width is fixed at 16. For passive color panels, only an 8-bit panel bus width is supported.	00 = 1-bit 01 = 2-bit 10 = 4-bit 11 = 8-bit
BPIX Bits 27–25	**Bits Per Pixel**—Indicates the number of bits per pixel in memory.	000 = 1 bpp, FRC bypassed 001 = 2 bpp 010 = 4 bpp 011 = 8 bpp 100 = 12 bpp/16 bpp (16 bits of memory used) 11x = reserved 1x1 = reserved
PIXPOL Bit 24	**Pixel Polarity**—Sets the polarity of the pixels.	0 = Active high 1 = Active low

Table 17-13. Panel Configuration Register Description (continued)

Name	Description	Settings
FLMPOL Bit 23	**First Line Marker Polarity**—Sets the polarity of the first line marker symbol.	0 = Active high 1 = Active low
LPPOL Bit 22	**Line Pulse Polarity**—Sets the polarity of the line pulse signal.	0 = Active high 1 = Active low
CLKPOL Bit 21	**LCD Shift Clock Polarity**—Sets the polarity of the active edge of the LCD shift clock.	0 = Active negative edge of LSCLK (in TFT mode, active on positive edge of LSCLK) 1 = Active positive edge of LSCLK (in TFT mode, active on negative edge of LSCLK)
OEPOL Bit 20	**Output Enable Polarity**—Sets the polarity of the output enable signal.	0 = Active high 1 = Active low
SCLKIDLE Bit 19	**LSCLK Idle Enable**—Enables/Disables LSCLK when VSYNC is idle in TFT mode.	0 = Disable LSCLK 1 = Enable LSCLK
END_SEL Bit 18	**Endian Select**—Selects the image download into memory as big or little endian format.	0 = Little endian 1 = Big endian
SWAP_SEL Bit 17	**Swap Select**—Controls the swap of data in little endian mode (when END_SEL = 1 this bit has no effect).	0 = 16 bpp mode 1 = 8 bpp. 4 bpp, 2 bpp, 1 bpp mode
REV_VS Bit 16	**Reverse Vertical Scan**—Selects the vertical scan direction as normal or reverse (the image flips along the x-axis). The SSA register must be changed accordingly.	0 = Vertical scan in normal direction 1 = Vertical scan in reverse direction
ACDSEL Bit 15	**ACD Clock Source Select**—Selects the clock source used by the alternative crystal direction counter.	0 = Use FRM as clock source for ACD count 1 = Use LP/HSYN as clock source for ACD count
ACD Bits 14–8	**Alternate Crystal Direction**—Toggles the ACD signal once every 1-16 FLM cycles based on the value specified in this field. The actual number of FLM cycles between toggles is the programmed value plus one.	For active mode (TFT=1), this parameter is not used. For passive mode (TFT=0), see description.
SCLKSEL Bit 7	**LSCLK Select**—Selects whether to enable or disable LSCLK in TFT mode when there is no data output.	0 = Disable OE and LSCLK in TFT mode when no data output 1 = Always enable LSCLK in TFT mode even if there is no data output
SHARP Bit 6	**Sharp Panel Enable**—Enables/Disables signals for Sharp HR-TFT panels.	0 = Disable Sharp signals 1 = Enable Sharp signals

Table 17-13. Panel Configuration Register Description (continued)

Name	Description	Settings
PCD Bits 5–0	**Pixel Clock Divider**—Specifies the divide ratio applied to LCDC_CLK. The LCDC_CLK (PerCLK2) is divided down by the Pixel Clock Divider to generate the pixel clock. The LSCLK will be the same as pixel clock in TFT mode, and will be 1/8 the frequency of the pixel clock in passive mode. For passive displays (TFT bit = 0), the frequency of LSCLK must be < 1/9 HCLK at 12 bpp and < 1/15 HCLK at 4 or 8 bpp. For passive matrix color panels (COLOR = 1 and PBSIZ = 11), PCD must be ≥ 2 and PCLK_DIV2 (PCDR Register) must = 0 (divide by 1). See Phase-Locked Loop and Clock Controller chapter in this manual for PCDR register information. For active displays (TFT bit = 1), the frequency of LSCLK must be < 1/5 HCLK. When PCD = 0, the pixel clock frequency is equal to the LCDC_CLK frequency. See application notes at http:\\www.freescale.com\imx.	TFT (TFT bit = 1) 000000 = divide ratio is 1 000001 = divide ratio is 2 000010 = divide ratio is 3 … 111111 = divide ratio is 64 Passive (TFT bit = 0) 000000 = divide ratio is 8[1] 000001 = divide ratio is 16[1] 000010 = divide ratio is 24 … 111111 = divide ratio is 512 **Note 1:** Do not use. For passive panels, PCD must be greater than or equal to 2 or 000010.

17.4.5 Horizontal Configuration Register

The Horizontal Configuration Register defines the horizontal sync pulse timing.

HCR Horizontal Configuration Register **Addr 0x0020501C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	H_WIDTH															
TYPE	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	0x0400															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	H_WAIT_1								H_WAIT_2							
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 17-14. Horizontal Configuration Register Description

Name	Description
H_WIDTH Bits 31–26	**Horizontal Sync Pulse Width**—Specifies the number of pixel clk periods that HSYNC is activated. The active time is equal to (H_WIDTH + 1) of the pixel clk period. For Sharp HR-TFT panels, H_WIDTH is typically set to 1.
Reserved Bits 25–16	Reserved—These bits are reserved and should read 0.
H_WAIT_1 Bits 15–8	**Wait Between OE and HSYNC**—Specifies the number of pixel clk periods between the last LD of each line and the beginning of the HSYNC. Total delay time equals (H_WAIT_1 + 1). For Sharp HR-TFT panels, H_WAIT_1 is typically set to 14.
H_WAIT_2 Bits 7–0	**Wait Between HSYNC and Start of Next Line**—Specifies the number of pixel clk periods between the end of HSYNC and the beginning of the first data of next line. Total delay time equals (H_WAIT_2 + 3).

17.4.6 Vertical Configuration Register

The Vertical Configuration Register defines the vertical sync pulse timing.

VCR Vertical Configuration Register **Addr 0x00205020**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	V_WIDTH															
TYPE	rw	rw	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
	0x0401															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	V_WAIT_1								V_WAIT_2							
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 17-15. Vertical Configuration Register Description

Name	Description
V_WIDTH Bits 31–26	**Vertical Sync Pulse Width**—Specifies the width, in lines, of the VSYNC pulse for active (TFT =1) mode. For a value of “000001”, the vertical sync pulse encompasses one HSYNC pulse. For a value of “000002”, the vertical sync pulse encompasses two HSYNC pulses, and so on. For passive (TFT=0) mode and non-color mode, see Figure 17-12.
Reserved Bits 25–16	Reserved—These bits are reserved and should read 0.
V_WAIT_1 Bits 15–8	**Wait Between Frames 1**—Defines the delay, in lines, between the end of the OE pulse and the beginning of the VSYNC pulse for active (TFT=1) mode. This field has no meaning in passive non-color mode. The actual delay is (V_WAIT_1). In passive color mode, this field is the delay, measured in virtual clock periods, between the last line of the frame to the beginning of the next frame.
V_WAIT_2 Bits 7–0	**Wait Between Frames 2**—Defines the delay, in lines, between the end of the VSYNC pulse and the beginning of the OE pulse of the first line in active (TFT=1) mode. The actual delay is V_WAIT_2) lines. Set this field to zero for passive non-color mode. The minimum value of this field is 0x01.

17.4.7 Panning Offset Register

The Panning Offset Register sets up the panning for the image.

POS Panning Offset Register **Addr 0x00205024**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												POS				
TYPE	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 17-16. Panning Offset Register Description

Name	Description	Settings
Reserved Bits 31–5	Reserved—These bits are reserved and should read 0.	
POS Bits 4–0	**Panning Offset**—Defines the number of bits that the data from memory is panned to the left before processing. POS is read by the LCDC once at the beginning of each frame.	To achieve panning of the final image by N bits:

Bits Per Pixel	POS	Effective # of Bits Panned on Image
1	N	N
2	2N	N
4	4N	N
8	8N	N
12	16N	N

17.4.8 LCD Cursor Position Register

The LCD Cursor Position Register is used to determine the starting position of the cursor on the LCD panel.

CPOS LCD Cursor Position Register **Addr 0x0020500C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CC			OP			CXP									
TYPE	rw	rw	r	rw	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								CYP								
TYPE	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 17-17. LCD Cursor X Position Register Description

Name	Description	Settings
CC Bits 31–30	**Cursor Control**—Controls the format of the cursor and the type of arithmetic operations.	When OP = 0 00 = Transparent, cursor is disabled 01 = Full cursor (Black for non-color displays; full-color for color displays) 10 = Reversed video 11 = Full (white) cursor When OP = 1, and color mode 00 = Transparent, cursor is disabled 01 = OR between background and cursor 10 = XOR between background and cursor 11 = AND between background and cursor
Reserved Bit 29	Reserved—This bit is reserved and should read 0.	
OP Bit 28	**Arithmetic Operation Control**—Enables/Disables arithmetic operations between the background and the cursor.	0 = Disable arithmetic operation 1 = Enable arithmetic operation
Reserved Bits 27–26	Reserved—These bits are reserved and should read 0.	
CXP Bits 25–16	**Cursor X Position**—Represents the cursor's horizontal starting position X in pixel count (from 0 to XMAX).	
Reserved Bits 15–9	Reserved—These bits are reserved and should read 0.	
CYP Bits 8–0	**Cursor Y Position**—Represents the cursor's vertical starting position Y in pixel count (from 0 to YMAX).	

17.4.9 LCD Cursor Width Height and Blink Register

The LCD Cursor Width Height and Blink Register is used to determine the width and height of the cursor, and how it blinks.

LCWHB LCD Cursor Width Height and Blink Register **Addr 0x00205010**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	BK_EN			CW								CH				
TYPE	rw	r	r	rw	rw	rw	rw	rw	r	r	r	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
								0X0101								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									BD							
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
								0X007F								

Table 17-18. LCD Cursor Width Height and Blink Register Description

Name	Description	Settings
BK_EN Bit 31	**Blink Enable**—Determines whether the blink enable cursor will blink or remain steady.	0 = Blink is disabled 1 = Blink is enabled
Reserved Bits 30–29	Reserved—These bits are reserved and should read 0.	
CW Bits 28–24	**Cursor Width**—Specifies the width of the hardware cursor in pixels.	This field can be any value between 1 and 31 (setting this field to zero disables the cursor)
Reserved Bits 23–21	Reserved—These bits are reserved and should read 0.	
CH Bits 20–16	**Cursor Height**—Specifies the height of the hardware cursor in pixels.	This field can be any value between 1 and 31 (setting this field to zero disables the cursor)
Reserved Bits 15–8	Reserved—These bits are reserved and should read 0.	
BD Bits 7–0	**Blink Divisor**—Sets the cursor blink rate. A 32 Hz clock from RTC is used to clock the 8-bit up counter. When the counter value equals BD, the cursor toggles on/off.	

17.4.10 LCD Color Cursor Mapping Register

The LCD Color Cursor Mapping Register defines the color of the cursor in passive or TFT color modes.

LCHCC LCD Color Cursor Mapping Register **Addr 0x00205014**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CUR_COL_R					CUR_COL_G						CUR_COL_B				
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 17-19. LCD Color Cursor Mapping Register Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
CUR_COL_R Bits 15–11	**Cursor Red Field**—Defines the red component of the cursor color in color mode.	**For 8 bpp/12 bpp:** 0000x = No red ... 1111x = Full red **For 16 bpp:** 00000 = No red ... 11111 = Full red
CUR_COL_G Bits 10–5	**Cursor Green Field**—Defines the green component of the cursor color in color mode.	**For 8 bpp/12 bpp:** 0000xx = No green .... 1111xx = Full green **For 16 bpp:** 000000 = No green ... 111111 = Full green
CUR_COL_B Bits 4–0	**Cursor Blue Field**—Defines the blue component of the cursor color in color mode.	**For 8 bpp/12 bpp:** 0000x = No blue ... 1111x = Full blue **For 16 bpp:** 00000 = No blue ... 11111 = Full blue

17.4.11 Sharp Configuration 1 Register

For 2 bpp modes, full black and full white are the two predefined display levels. The other two intermediate gray-scale shading densities can be adjusted within the Sharp Configuration 1 Register. This register also controls the relative delay timing of the additional signals CLS, REV, and PS required for Sharp TFT displays. The TFT timing diagram that shows the relationship between these signals is shown in Figure 17-17 on page 17-33.

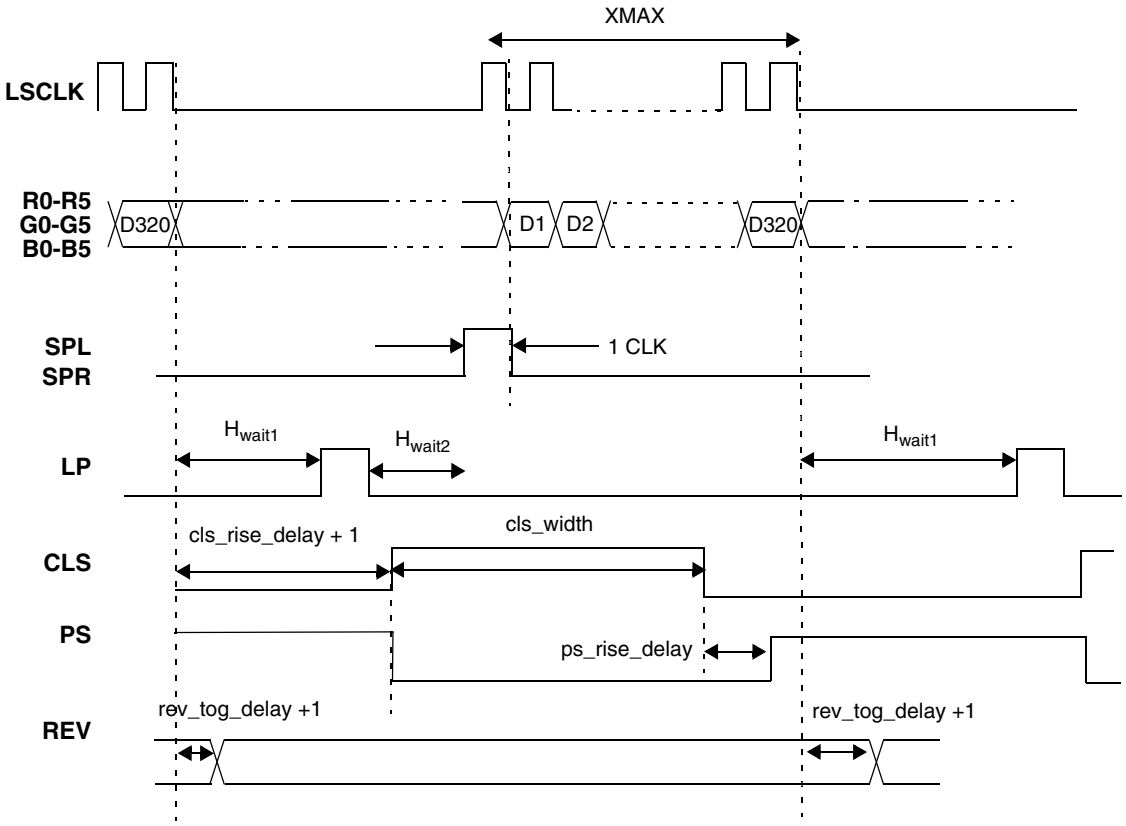
LSCR1 Sharp Configuration 1 Register **0x00205028**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PS_RISE_DELAY								CLS_RISE_DELAY							
TYPE	rw	rw	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	0x0400															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					REV_TOGGLE_DELAY				GRAY 2				GRAY 1			
TYPE	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	1	1	0	1	1	1	0	0	1	1
	0x0373															

Table 17-20. Sharp Configuration 1 Register Description

Name	Description	Settings
PS_RISE_DELAY Bits 31–26	**PS Rise Delay**—Controls the delay of the rising edge of PS relative to the falling edge of CLS. Delay is measured in LCDC_CLK (PerCLK2) periods.	000000 = 0 LSCLK periods 000001 = 1 LSCLK period ... 111111 = 63 LSCLK periods
Reserved Bits 25–24	Reserved—These bits are reserved and should read 0.	
CLS_RISE_DELAY Bits 23–16	**CLS Rise Delay**—Controls the delay of the rising edge of CLS relative to the last LD of the line. Delay is measured in LCDC_CLK (PerCLK2) periods.	00000000 = 1 LSCLK period 00000001 = 2 LSCLK periods ... 11111111 = 256 LSCCLK periods
Reserved Bits 15–12	Reserved—These bits are reserved and should read 0.	
REV_TOGGLE_DELAY Bits 11–8	**REV Toggle Delay**—Controls the transition delay of REV relative to the last LD of the line. Delay is measured in LCDC_CLK (PerCLK2) periods.	0000 = 1 LSCLK period 0001 = 2 LSCLK periods ... 1111 = 16 LSCLK periods
GRAY 2 Bits 7–4	**Gray-Scale 2**—Represents one of the two gray-scale shading densities.	This field is programmable to any value between 0 and 16 (0 and 16 are defined as two of the four colors).
GRAY 1 Bits 3–0	**Gray-Scale 1**—Represents the other gray-scale shading.	This field is programmable to any value between 0 and 16 (0 and 16 are already defined as two of the four colors).



Falling edge of PS aligns with rising edge of CLS
The rising edge delay of PS is programmed by PS_RISE_DELAY
CLS_HI_WIDTH is equal to PWM_SCR0 • 256 + PWM_WIDTH in units of LSCLK.
SPL/SPR pulse width is fixed and aligned to the first data of the line.
REV toggles every LP period.

Figure 17-17. Horizontal Timing in MC9328MXL

17.4.12 PWM Contrast Control Register

The PWM Contrast Control Register is used to control the signal output at the CONTRAST pin, which controls the contrast of the LCD panel.

PWMR PWM Contrast Control Register **Addr 0x0020502C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								CLS_HI_WIDTH								
TYPE	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LDMSK					SCR		CC_EN	PW							
TYPE	rw	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 17-21. PWM Contrast Control Register Description

Name	Description	Settings
Reserved Bits 31–25	Reserved—These bits are reserved and should read 0.	
CLS_HI_WIDTH Bit 24–16	**CLS High Pulse Width**—Controls the Hi Pulse width of CLS in units of SCLK. The actual pulse width = CLS_HI_WDITH +1.	
LDMSK Bit 15	**LD Mask**—Enables/Disables the LD outputs to zero for panel power-off sequence as required by Sharp TFT or other panels.	0 = LD [15:0] is normal 1 = LD [15:0] always equals 0
Reserved Bits 14–11	Reserved—These bits are reserved and should read 0.	
SCR Bits 10–9	**Source Select**—Selects the input clock source for the PWM counter. The PWM output frequency is equal to the frequency of the input clock divided by 256.	00 = Line pulse 01 = Pixel clock 10 = LCD clock 11 = Reserved
CC_EN Bit 8	**Contrast Control Enable**—Enables/Disables the contrast control function.	0 = Contrast control is off 1 = Contrast control is on
PW Bits 7–0	**Pulse-Width**—Controls the pulse-width of the built-in pulse-width modulator, which controls the contrast of the LCD screen.	

17.4.13 Refresh Mode Control Register

The Refresh Mode Control Register is used to control refresh characteristics.

RMCR Refresh Mode Control Register **Addr 0x00205034**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
									0x0000							

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															LCDC_EN	SELF_REF
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
									0x0000							

Table 17-22. Refresh Mode Control Register Description

Name	Description	Settings
Reserved Bits 31–2	Reserved—These bits are reserved and should read 0.	
LCDC_EN Bit 1	**LCDC Enable**—Enables/Disables the LCDC.	0 = Disable the LCDC 1 = Enable the LCDC
SELF_REF Bit 0	**Self-Refresh**—Enables/Disables self-refresh mode.	0 = Disable self-refresh 1 = Enable self-refresh

NOTE:

1. On entering self-refresh mode, the LSCLK and LD [15:0] signals stay low. HYSN and VSYN operate normally.
2. Except for the SSA and Mapping RAM registers, all configurations must be performed before enabling the LCDC to avoid a malfunction.
3. The SSA must always match the address range of the RAM selected. If the user wants to switch between various types of RAM, the LCDC must be disabled before switching.

17.4.14 DMA Control Register

There is a 16 × 32 bit line buffer in the LCDC that stores DMA data from system memory. The DMA Control Register controls the DMA burst length and when to trigger a DMA burst in terms of the number of data bytes left in the pixel buffer.

DMACR DMA Control Register **Addr 0x00205030**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	BURST												HM			
TYPE	rw	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw
RESET	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
								0x8008								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													TM			
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
								0x0002								

Table 17-23. DMA Control Register Description

Name	Description	Settings
BURST Bit 31	**Burst Length**—Determines whether the burst length is fixed or dynamic.	0 = Burst length is dynamic 1 = Burst length is fixed
Reserved Bits 30–20	Reserved—These bits are reserved and should read 0.	
HM Bits 19–16	**DMA High Mark**—Establishes the high mark for DMA requests. For dynamic burst length, once the DMA request is made, data is loaded and the pixel buffer continues to be filled until the number of empty words left in the DMA FIFO is equal to the high mark minus 2. For fixed burst length, the burst length (in words) of each request is equal to the DMA high mark setting.	
Reserved Bits 15–4	Reserved—These bits are reserved and should read 0.	
TM Bits 3–0	**DMA Trigger Mark**—Sets the low level mark in the pixel buffer to trigger a DMA request. The low level mark equals the number of words left in the pixel buffer.	

NOTE:

For SDRAM access, a fixed burst length of 8 is recommended
fixed burst length = 1
high mark = 8
low mark = 4
For bus that is heavy loaded that requires SDRAM access, a dynamic burst length is recommended
fixed burst length = 0

high mark = 3
low mark = 8
For an especially heavily loaded system, increasing the low mark value increases the chance of granting of the system bus, at the expense of more frequent bus requests.

The low mark should never be set higher than 10, and high mark should always be set to 3.

17.4.15 Interrupt Configuration Register

The Interrupt Configuration Register is used to configure the interrupt conditions.

LCDICR Interrupt Configuration Register **Addr 0x00205038**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0X0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														INT SYN		INT CON
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	r	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 17-24. Interrupt Configuration Register Description

Name	Description	Settings
Reserved Bits 31–3	Reserved—These bits are reserved and should read 0.	
INTSYN Bit 2	**Interrupt Source**—Determines if an interrupt flag is set during last data/first data of frame loading or on last data/first data of frame output to the LCD panel. **Note:** There is a latency between loading the last/first data of frame to output to LCD panel.	0 = Interrupt flag is set on loading the last data/first data of frame from memory 1 = Interrupt flag is set on output of the last data/first data of frame to LCD panel
Reserved Bit 1	Reserved—This bit is reserved and should read 0.	
INTCON Bit 0	**Interrupt Condition**—Determines if an interrupt condition is set at the beginning or the end of frame condition.	0 = Interrupt flag is set when the End of Frame (EOF) is reached 1 = Interrupt flag is set when the Beginning of Frame (BOF) is reached

17.4.16 Interrupt Status Register

The read-only Interrupt Status Register indicates whether an interrupt has occurred. The interrupt flag is cleared by reading the register.

LCDISR Interrupt Status Register **Addr 0x00205040**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													UDR_ ERR	ERR_ RES	EOF	BOF
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 17-25. Interrupt Status Register Description

Name	Description	Settings
Reserved Bits 31–4	Reserved—These bits are reserved and should read 0.	
UDR_ERR Bit 3	**Under Run Error**—Indicates whether the LCDC FIFO has hit an under-run condition. This is when the data output rate is faster than the data input rate to the FIFO. Under-run can cause erroneous data output to LD. The LD data output rate must be adjusted to prevent this error.	0 = Interrupt has not occurred 1 = Interrupt has occurred
ERR_RES Bit 2	**Error Response**—Indicates whether the LCDC has issued a read data request and has received a response from the memory controller not equal to ‘OK.’ It is cleared by reading the status register, at power on reset, or when the LCDC is disabled.	0 = Interrupt has not occurred 1 = Interrupt has occurred
EOF Bit 1	**End of Frame**—Indicates whether the end of frame has been reached. It is cleared by reading the status register, at power on reset, or when the LCDC is disabled.	0 = Interrupt has not occurred 1 = Interrupt has occurred
BOF Bit 0	**Beginning of Frame**—Indicates whether the beginning of frame has been reached. It is cleared by reading the status register, at power on reset, or when the LCDC is disabled.	0 = Interrupt has not occurred 1 = Interrupt has occurred

17.4.17 Mapping RAM Registers

The mapping RAM is used for mapping 4-bit codes for grayscale to the 16 gray shades, and for mapping 4- or 8-bit color codes to either the 4096 (for active panels) or 512 colors (for passive panels). The color RAM (0x00205800-0x00205BFC) contains 256 entries and each entry is 12 bits wide. Each RAM entry use 4 bytes of address space. The RAM can be accessed with word transactions only and the address must be word aligned. Unimplemented bits are read as 0. Byte or halfword access to the RAM corrupts its contents. All read/write data use least significant twelve bits.

In 4 bpp mode, the first sixteen RAM entries are used. In 8 bpp mode, all 256 RAM entries are used. The color RAM is not initialized at reset. With any given panel, only one of the following settings is valid:

- 1 bpp monochrome mode
- 4 bpp gray-scale mode
- 4 bpp passive matrix color mode
- 8 bpp passive matrix color mode
- 4 bpp active matrix color mode
- 8 bpp active matrix color mode
- 12/16 bpp active matrix color mode

17.4.17.1 One Bit/Pixel Monochrome Mode

The mapping RAM is not used in this mode because the LCDC uses the display data in memory to drive the panel directly.

17.4.17.2 Four Bits/Pixel Gray-Scale Mode

In four bits/pixel gray-scale mode, a 4-bit code represents a gray-scale level. The first 16 mapping RAM entries must be written to define the codes for all 16 combinations.

BIT	11	10	9	8	7	6	5	4	3	2	1	0
									GPM			
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	?	?	?	?

Table 17-26. Four Bits/Pixel Gray-Scale Mode

Name	Description
Reserved Bits 11–4	Reserved—These bits are reserved and should read 0.
GPM Bits 3–0	**Gray Palette Map**—Represents the gray-scale level for a given pixel code.

17.4.17.3 Four Bits/Pixel Passive Matrix Color Mode

In four bits/pixel passive matrix color mode, a 4-bit code represents a 12-bit color. Because just four bits are used to encode the color, a maximum of 16 colors can be selected out of a palette of 4096. The first 16 mapping RAM entries must be written to define the codes for the 16 available combinations.

BIT	11	10	9	8	7	6	5	4	3	2	1	0
	R				G				B			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	?	?	?	?	?	?	?	?	?	?	?	?
	0X???											

Table 17-27. Four Bits/Pixel Passive Matrix Color Mode

Name	Description
R Bits 11–8	**Red Level (color display)**—Represents the red component level in the color.
G Bits 7–4	**Green Level (color display)**—Represents the green component level in the color.
B Bits 3–0	**Blue Level (color display)**—Represents the blue component level in the color.

17.4.17.4 Eight Bits/Pixel Passive Matrix Color Mode

In eight bits/pixel passive matrix color mode, an 8-bit code represents a 12-bit color. Because eight bits are used to encode the color, a maximum of 256 colors can be selected out of a palette of 4096. All 256 mapping RAM entries must be written to define the codes for the 256 available combinations.

BIT	11	10	9	8	7	6	5	4	3	2	1	0
	R				G				B			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	?	?	?	?	?	?	?	?	?	?	?	?
	0x???											

Table 17-28. Eight Bits/Pixel Passive Matrix Color Mode

Name	Description
R Bits 11–8	**Red Level (color display)**—Represents the red component level in the color.
G Bits 7–4	**Green Level (color display)**—Represents the green component level in the color.
B Bits 3–0	**Blue Level (color display)**—Represents the blue component level in the color.

17.4.17.5 Four Bits/Pixel Active Matrix Color Mode

In four bits/pixel active color mode, a 4-bit code represents a 12-bit color. Because just four bits are used to encode the color, a maximum of 16 colors can be selected out of a palette of 4096. The first 16 mapping RAM entries must be written to define the codes for the 16 available combinations.

BIT	11	10	9	8	7	6	5	4	3	2	1	0
	R				G				B			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	?	?	?	?	?	?	?	?	?	?	?	?
	0x???											

Table 17-29. Four Bits/Pixel Active Matrix Color Mode

Name	Description
R Bits 11–8	**Red Level (color display)**—Represents the red component level in the color.
G Bits 7–4	**Green Level (color display)**—Represents the green component level in the color.
B Bits 3–0	**Blue Level (color display)**—Represents the blue component level in the color.

17.4.17.6 Eight Bits/Pixel Active Matrix Color Mode

In eight bits/pixel active color mode, an 8-bit code represents a 12-bit color. Because eight bits are used to encode the color, a maximum of 256 colors can be selected out of a palette of 4096. All 256 mapping RAM entries must be written to define the codes for the 256 available combinations.

BIT	11	10	9	8	7	6	5	4	3	2	1	0
	R				G				B			
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	?	?	?	?	?	?	?	?	?	?	?	?
	0x???											

Table 17-30. Eight Bits/Pixel Active Matrix Color Mode

Name	Description
R Bits 11–8	**Red Level (color display)**—Represents the red component level in the color.
G Bits 7–4	**Green Level (color display)**—Represents the green component level in the color.
B Bits 3–0	**Blue Level (color display)**—Represents the blue component level in the color.

17.4.17.7 Twelve Bits/Pixel and Sixteen Bits/Pixel Active Matrix Color Mode

In this mode the mapping RAM is not used, because the LCDC uses the display data in memory to drive the panel directly.

Chapter 18
Multimedia Card/Secure Digital Host Controller Module (MMC/SD)

18.1 Introduction

The Multimedia Card (MMC) is a universal low cost data storage and communication medium implemented as a hardware card with a simple control unit and a compact, easy-to-implement interface that is designed to cover a wide variety of applications such as electronic toys, organizers, PDAs, and smart phones. MMC communication is based on an advanced 7-pin serial bus designed to operate in a low voltage range at medium speed (20 Mbps).

The Secure Digital Card (SD) is an evolution of the MMC with an additional 2 pins in the form factor that is specifically designed to meet the security, capacity, performance, and environmental requirements inherent in new audio and video consumer electronic devices. The physical form factor, pin assignment, and data transfer protocol are compatible with the MMC. The SD is composed of a memory card and an I/O card. The memory card includes a copyright protection mechanism that complies with the security requirements of the Secure Digital Music Initiative (SDMI) standard, and is faster and has a higher memory capacity. The I/O card combines high-speed data I/O with low-power consumption for mobile electronic devices.

The Multimedia Card/Secure Digital Host Controller module (MMC/SD module) integrates MMC support with SD memory and I/O functions. The copyright protection mechanism employs mutual authentication and a new cipher algorithm, and is handled in software post-processing.

18.2 Features List

The MMC/SD module includes the following features:

- Compatible with the *MultiMediaCard System Specification* (SPI mode excluded), version 3.1
- Compatible to 1/4 bit with the *SD Memory Card Specification* (SPI mode excluded), version 1.0 and *SD I/O Specification* (SPI mode excluded), version 1.0 with 1 or 4 channel(s)
- 20–100 Mbps maximum data rate supports up to 10 cards (including one SD card)
- Password protection for cards
- Built-in programmable frequency counter for MMC/SD bus
- Maskable hardware interrupt for card detection (insertion/removal), SD I/O interrupt, internal status, and FIFO status
- Contains an integrated 32 × 16-bit FIFO
- Supports plug-and-play (PnP)
- Supports many SD functions including multiple I/O and combined I/O and memory
- Supports up to seven I/O functions plus one memory on a single SD I/O card

- Card can interrupt MMC/SD module
- MMC/SD module is IP bus compatible with Motorola's microcontrollers
- Supports single or multiple block access, or stream access to the card for read, write, or erase operations
- Supports SD I/O ReadWait and interrupt detection during 1- or 4-bit access

18.3 MMC/SD Module Block Diagram

Figure 18-1 is the block diagram of the MMC/SD module. Figure 18-2 is the system interconnection diagram for the MMC/SD Host Controller module.

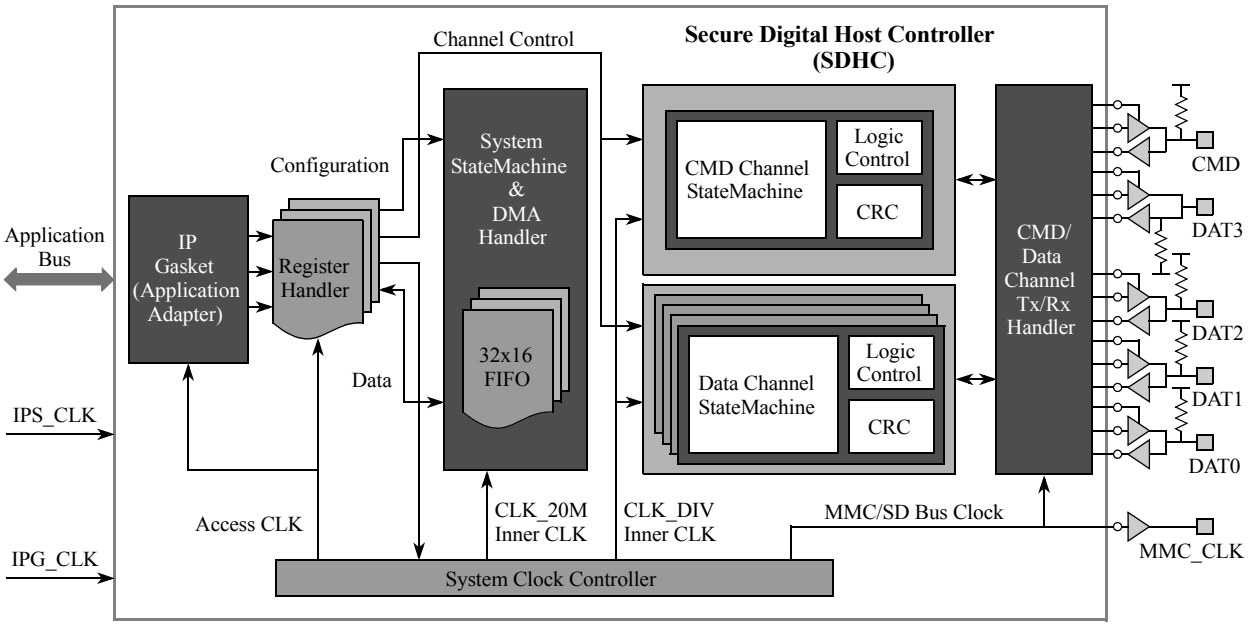


Figure 18-1. MMC/SD Module Block Diagram

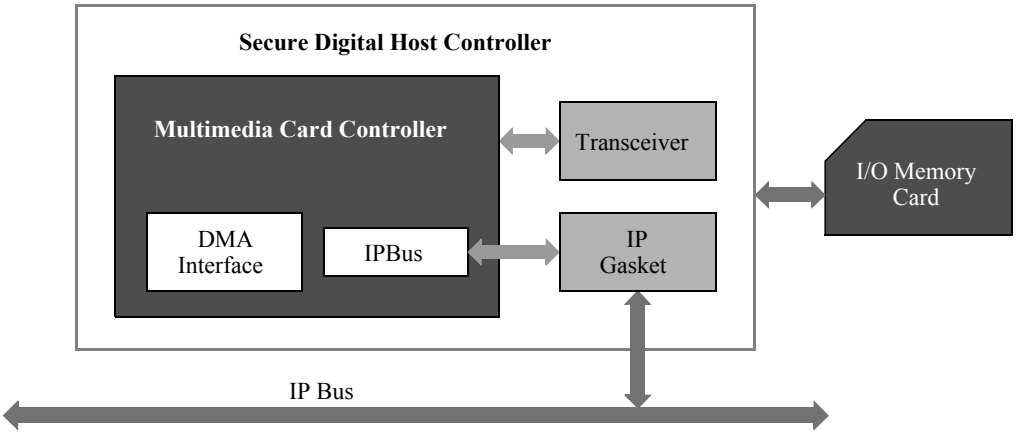


Figure 18-2. System Interconnection with MMC/SD Module

18.4 MMC/SD Module and Card Information

The following sections provide information about the MMC and SD cards, signal descriptions, module and card communication, and pin configuration.

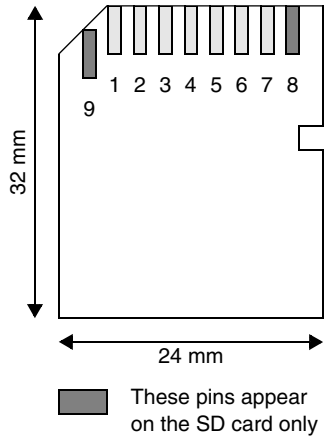
18.4.1 MMC and SD Card Pin Assignments and Registers

The MMC and SD cards are 7- or 9-pin cards that operate as external memory storage for the MC9328MXL. The pin assignment and form factor are shown in Table 18-1.

There are three types of cards:

- **Read-only (ROM) cards**—manufactured with fixed data content and used as distribution media, such as for software, audio, and video.
- **Read/write (RW) cards**—typically sold blank and used as mass data storage or end user storage, such as FLASH, one-time programmable (OTP), or multiple-time programmable (MTP).
- **I/O cards**—typically feature an additional interface link and are intended for communication (such as modems).

Table 18-1. MMC/SD Card Pin Assignment

Form Factor and Pinout	Pin Number	MMC Card	SD Card	
			1-Bit Mode	4-Bit Mode
	1	Reserved	Card Detect	Data Line DAT [3]
	2	Command / Response (CMD)		
	3	Supply Voltage Ground (Vss)		
	4	Supply Voltage (Vdd)		
	5	Clock (CLK)		
	6	Supply Voltage Ground (Vss)		
	7	Data Line DAT [0]		
	8		Interrupt (IRQ)	Data Line DAT [1] or Interrupt (IRQ)
	9		ReadWait (RW)	Data Line DAT [2] or ReadWait (RW)

Each card has a set of information registers that hold the operating parameters and other card conditions and are described in Table 18-2.

Table 18-2. MMC/SD Card Registers

MMC or SD	Identifier	Register Name	Description	Size (Bits)
Both	CID	Card Identification Number	Each card has a unique CID.	128[1]

Table 18-2. MMC/SD Card Registers (continued)

MMC or SD	Identifier	Register Name	Description	Size (Bits)
Both	RCA	Relative Card Address	Assigned by the MMC/SD module during initialization.	16
Both	DSR	Driver Stage	Configures the card's output drivers. Usage is optional, not required.	16
Both	CSD	Card Specific Data	Contains information on the card's operating conditions.	128[1]
Both	OCR	Operation Conditions	Indicates the card's operating voltage range. Detects restricted cards and indicates whether power-up is complete. Usage is optional, not required.	32
Both	CSR	Card Status Register	Contains card's error and status information. Sent to the MMC/SD module in response format R1.	32
SD only	SCS	SD Card Status	Contains status information proprietary to the SD card (such as protection, card type, and bus width).	512
SD only	SCR	SD Configuration Register	Contains additional configuration information only applicable to the SD card.	64

1. There can be fewer bits for the SD I/O Card, depending on implementation (see *SDIO Card Specification,* version 1.0 for more information).

NOTE:

The SD I/O card contains additional registers that are not included in Table 18-2.

18.4.2 Communication

The MMC/SD module and the card communicate via commands, responses, and data bytes, defined as follows:

- **Commands**—Commands initiate operations. Commands are transferred serially on the SD_CMD line from the MMC/SD module to a single card or to all cards. Each command token is preceded by a start bit (0) and concluded by an end bit (1). The total length of a command is 48 bits, and is protected by CRC bits.
- **Responses**—Responses are answers to commands and cannot be initiated. Responses are transferred serially on the SD_CMD lines from a single addressed card or multiple responses can be sent synchronously from all cards to the MMC/SD module. The format of each response varies based on the command that initiated the response. See Section 18.7.8.5, "Response Formats," for more information.
- **Data**—Data is transferred via the SD_DAT line(s) from the card to the MMC/SD module or from the MMC/SD module to the card. Not all operations include data transfer.

18.4.3 Signal Description

The MMC/SD module uses six I/O pins to communicate with external MMC and SD cards.

- **SD_CMD**—Bidirectional command/response signal between the MMC/SD module and the card. Open-drain for initialization state and push-pull for fast command transfers.
- **SD_CLK**—MMC/SD module to card clock signal (output).

- **SD_DAT [3:0]**—Four bidirectional data signals. When in push-pull mode, one card or the MMC/SD module can drive each line at a time.

18.4.4 Pin Configuration for the MMC/SD Module

Section 18.4.3, "Signal Description," lists the pins used by the MMC/SD module. These pins are multiplexed with other functions on the device and must be configured for MMC/SD module operation.

NOTE:

The user must ensure that the data direction bits in the GPIO are set to the correct direction for proper operation. See Section 29.5.1, "Data Direction Registers," on page 29-8 for details.

Table 18-3. Pin Configuration

Pin	Setting	Configuration Procedure
SD_CMD	Primary function of GPIO Port B [13]	1. Clear bit 13 of Port B GPIO in Use Register (GIUS_B) 2. Clear bit 13 of Port B General Purpose Register (GPR_B) 3. Set bit 13 of Port B Pulling Enable Register (PUEN_B)
SD_CLK	Primary function of GPIO Port B [12]	1. Clear bit 12 of Port B GPIO in Use Register (GIUS_B) 2. Clear bit 12 of Port B General Purpose Register (GPR_B
SD_DAT [3]	Primary function of GPIO Port B [11]	1. Clear bit 11 of Port B GPIO in Use Register (GIUS_B) 2. Clear bit 11 of Port B General Purpose Register (GPR_B))
SD_DAT [2]	Primary function of GPIO Port B [10]	1. Clear bit 10 of Port B GPIO in Use Register (GIUS_B) 2. Clear bit 10 of Port B General Purpose Register (GPR_B) 3. Set bit 10 of Port B Pulling Enable Register (PUEN_B) SD_DAT1:1. Clear bit 9 of Port B GPIO in Use Register (GIUS_B)
SD_DAT [1]	Primary function of GPIO Port B [9]	1. Clear bit 9 of Port B GPIO in Use Register (GIUS_B) 2. Clear bit 9 of Port B General Purpose Register (GPR_B) 3. Set bit 9 of Port B Pulling Enable Register (PUEN_B)
SD_DAT [0]	Primary function of GPIO Port B [8]	1. Clear bit 8 of Port B GPIO in Use Register (GIUS_B) 2. Clear bit 8 of Port B General Purpose Register (GPR_B) 3. Set bit 8 of Port B Pulling Enable Register (PUEN_B)

18.5 Functional Description

Figure 18-1 on page 18-2 shows a block diagram of the MMC/SD module. The following sections provide brief functional descriptions of the major system blocks, including the DMA interface, memory controller (register handler), logic/command controller, and system clock controller.

18.5.1 DMA Interface

The DMA interface block controls all data routing between the external data bus (DMA access), internal MMC/SD module data bus, and internal system FIFO access through a dedicated state machine that monitors the status of FIFO content (empty or full), FIFO address, and byte/block counters for the MMC/SD module (inner system) and the application (user programming).

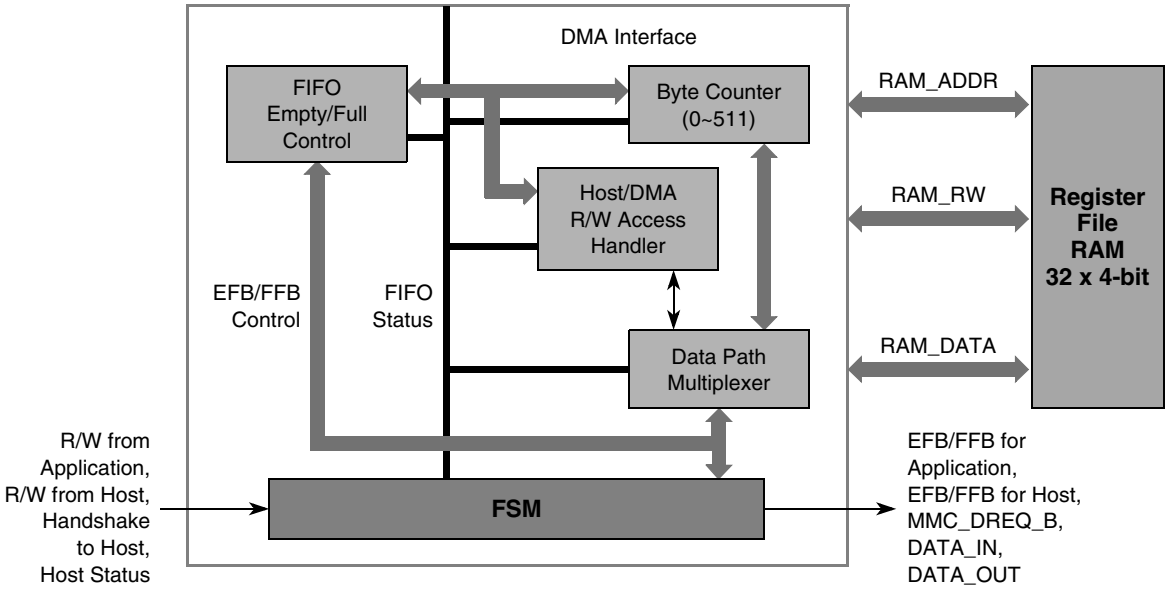


Figure 18-3. DMAC Interface Block Diagram

The DMAC interface block also handles burst requests to the external DMA controller, the internal register write-error detection, the SD I/O's ReadWait handling, and all IP-related output responses.

18.5.1.1 DMA Burst Request

In the MMC/SD module, there is a 32 × 16-bit FIFO to decrease the latency during the data transfer on the MMC/SD bus. The FIFO is configured differently for 1-bit and 4-bit access modes. The FIFO is operated as four 8 × 16-bit FIFOs for 1-bit access and as one 32 × 16-bit FIFO for 4-bit access.

Code Example 18-1. DMA Configuration Example

```
// DMAC init for SDHC channel setup
        *(P_U32)DMA_DCR         = 0x0001;       // Enable the DMA (DEN)
        *(P_U32)DMA_IMR         = 0x07FF;       // Disable all I/O Channel IRQ
             if (direction == 1){                    //write
        *(P_U32)DMA_SAR1        = Memory_Addr;  // Source Address
        *(P_U32)DMA_DAR1        = 0x00214038;   // Destination Address
             }
        else{                                   //read
        *(P_U32)DMA_DAR1        = Memory_Addr;  // Destination Address
        *(P_U32)DMA_SAR1        = 0x00214038;   // Source Address
             }
             *(P_U32)DMA_CNTR1       = Size;         // Set No of Byte transfer
        *(P_U32)DMA_CCR1        = dir; // Ch1: FIFO as the target, Linear Mem source,
                                       // Mem inc, 16-bit target, 32-bit source,
                                       // Request Enable, DMA Disable
```

```
*(P_U32)DMA_RSSR1       = 0x000D;       // Ch1: DMA request select; SDHC is bit[13]
```

```
            if(SD_4bit_enable)
        *(P_U32)DMA_BLR1= 0x0000; // Ch1: No. of FIFO to be read, burst length x32
         else
        *(P_U32)DMA_BLR1= 0x0010;       // Ch1: No. of FIFO to be read, burst length x8

// Start DMA and Poll end of DMA transfer done
         *(P_U32)DMA_ISR = 0x0002;       // Clear DMA ISR for MMC
         *(P_U32)DMA_CCR1 = dir_en;       // Ch1: FIFO as the target, Linear Mem source,
                                                 // Mem inc, 16-bit target, 32-bit source,
                                        // Request Enable, DMA Enable
         while ( ((*(P_U32)DMA_ISR) & 0x2) ==0 );//wait for the data transfer complete
         *(P_U32)DMA_ISR = 0x0002;       // Clear DMA ISR for MMC
         *(P_U32)DMA_CCR1 = dir;          // Ch1: FIFO as the target, Linear Mem source,
                                        // Mem inc, 16-bit target, 32-bit source,
                                        // Request Enable, DMA Disable
// End of DMA usage
```

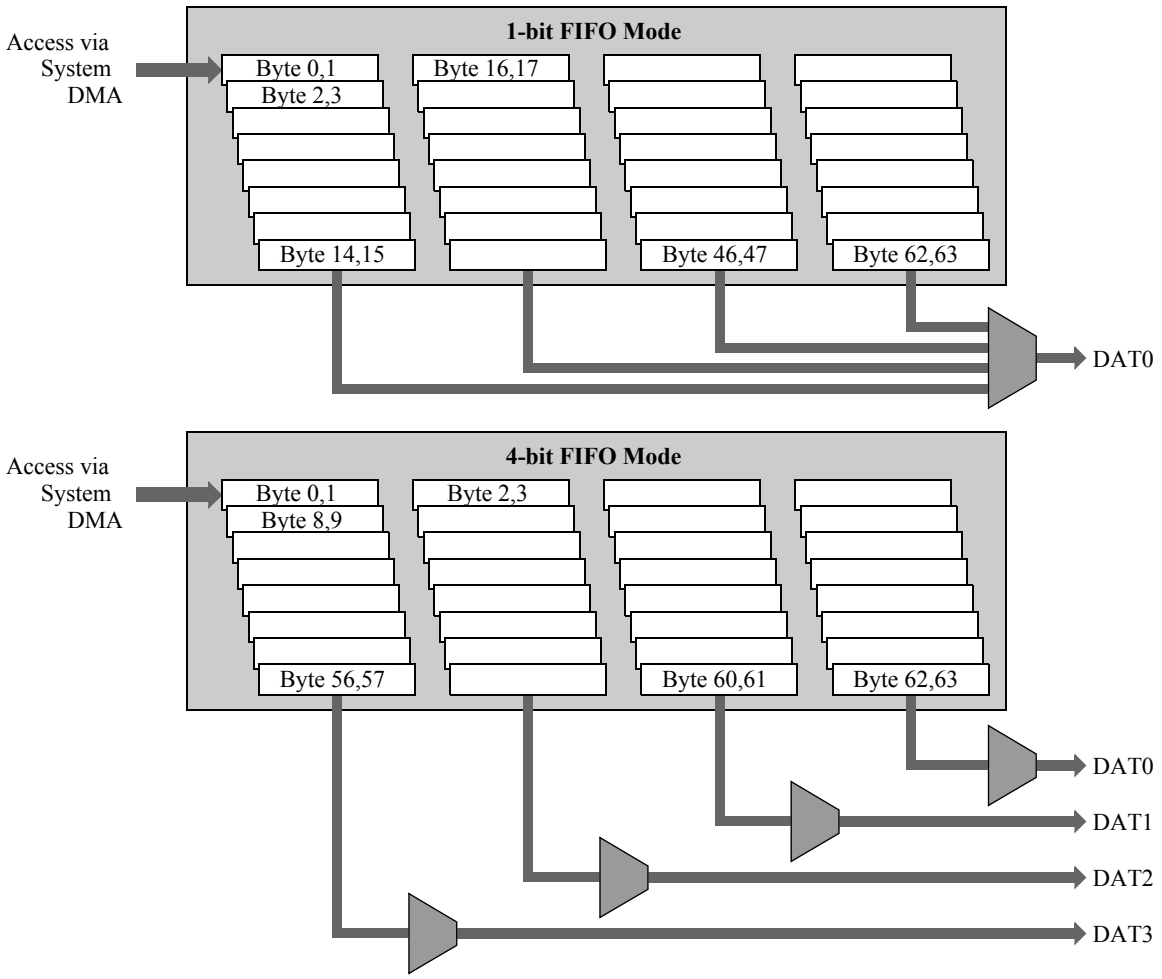


Figure 18-4. FIFO Usage for Different Modes

18.5.1.2 Write-Error Detection

To avoid incorrect register access and provide error indications, the DMA interface monitors the transfer type and access size of data transactions, generates bus errors, and ignores the current invalid configuration. See Table 18-4 on page 18-12 for the valid register addresses.

18.5.2 Memory Controller (Register Handler)

The memory controller provides SD I/O-IRQ and ReadWait service handling, card detection, command response handling, and all MMC/SD module interrupt handling. It is the sub-module where the user must place the register table. Figure 18-5 shows the memory controller block diagram.

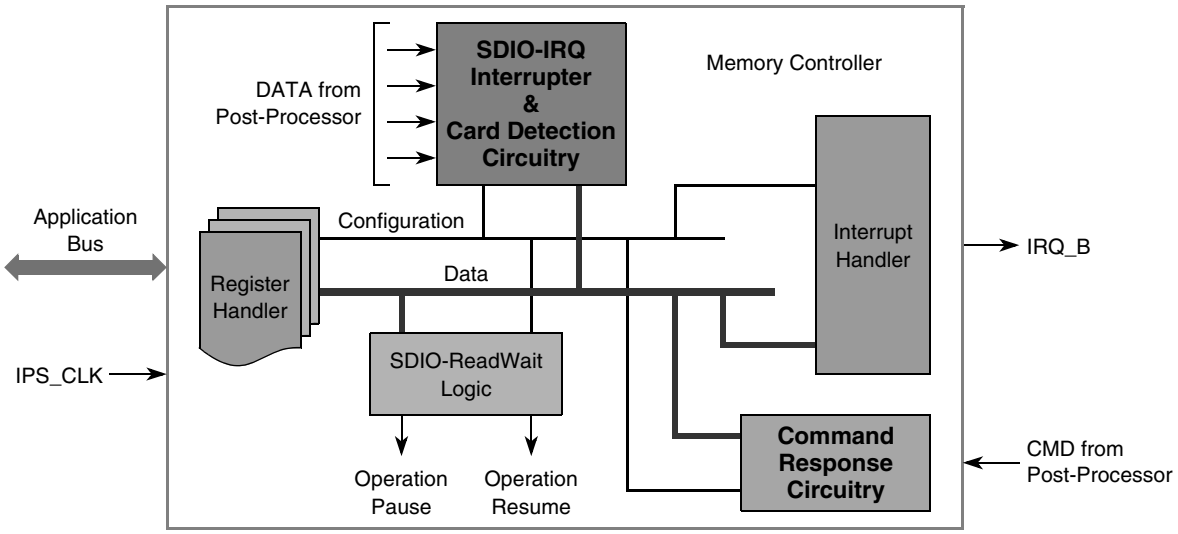


Figure 18-5. Memory Controller Block Diagram

18.5.2.1 SD I/O—IRQ and ReadWait Service Handling

There is a 1-bit or 4-bit interrupt response from the SD I/O peripheral card. In 1-bit mode, the interrupt response is simply that the SD_DAT [1] line is held low and is not used as data. The memory controller use SD_CLK to sample the status of SD_DAT[1] and generates and interrupt according to this low and the system interrupt continues until the source is removed (SD_DAT [1] returns to a high level).

Because pin 8 of the SD card is shared by the IRQ and SD_DAT [1] lines in 4-bit SD mode, an interrupt is sent by the card and recognized by the MMC/SD module only during a specific time. This is known as the interrupt period. The controller must sample SD_DAT [1] during this period to determine the IRQ status of the attached card. The interrupt period happens only at the boundary of each block (512 bytes).

ReadWait is another feature in SD I/O. It allows the user to submit command(s) during the data transfer. In ReadWait mode, the block temporarily pauses the data transfer operation counter and related status, however it keeps the clock running and allows the user to submit command(s) normally. After all commands are submitted, the user can switch back to the data transfer operation and all counter and status values are resumed as access continues. The feature is only available with a ReadWait enabled SDIO device.

18.5.2.2 Card Detection

The SD_DAT [3] pin detects card presence and is pulled low by default. When there is no card on the MMC/SD bus, SD_DAT [3] shows a low voltage level. When any card is inserted on the bus, the card's SD_DAT [3] pin pulls the bus line high to trigger the memory controller block's detection circuit and interrupt the processor to indicate that a card is inserted. The detection circuitry is sampled by the MC9328MXL system clock (HCLK), therefore card detection is still valid even if the MMC/SD module is not enabled. Similarly, card removal also

generates an interrupt. Because the mechanism is based on the value of the SD_DAT [3] line, only single-card systems benefit from card detection. After the card is detected, the user must mask the card detection interrupt to avoid misleading interrupt generation during card access as the SD_DAT lines change.

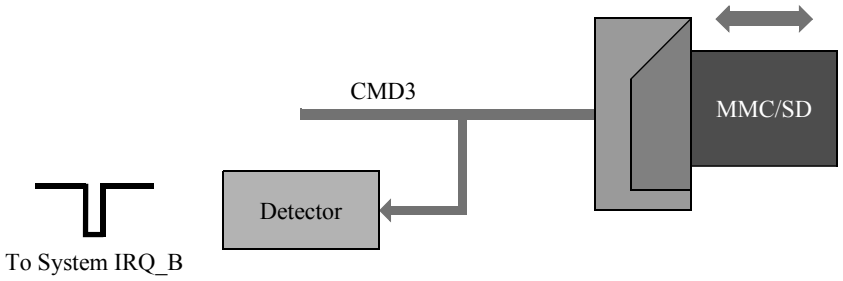


Figure 18-6. Card Detection Mechanism

18.5.2.3 MMC/SD Module Interrupt Handling

Interrupts generated from the MMC/SD module originate from errors or are status indicators. The MMC/SD module checks response and data CRCs and the internal watchdog timer. An error is generated when any of those checks fail. Status indicators include response done, transfer done, and FIFO status. Interrupt masking and generation is handled in the interrupt handler block.

18.5.3 Logic and Command Interpreters

The two interpreters are built similarly and consist of 3 parts: inner state machine, sub-module controller, and CRC accelerator.

The command controller handles all interrupts related to the command line (SD_CMD) including command data sequence generation, command response extraction, CRC generation and checking, and response time-out. A state machine, logic controller, and CRC accelerator control these functions.

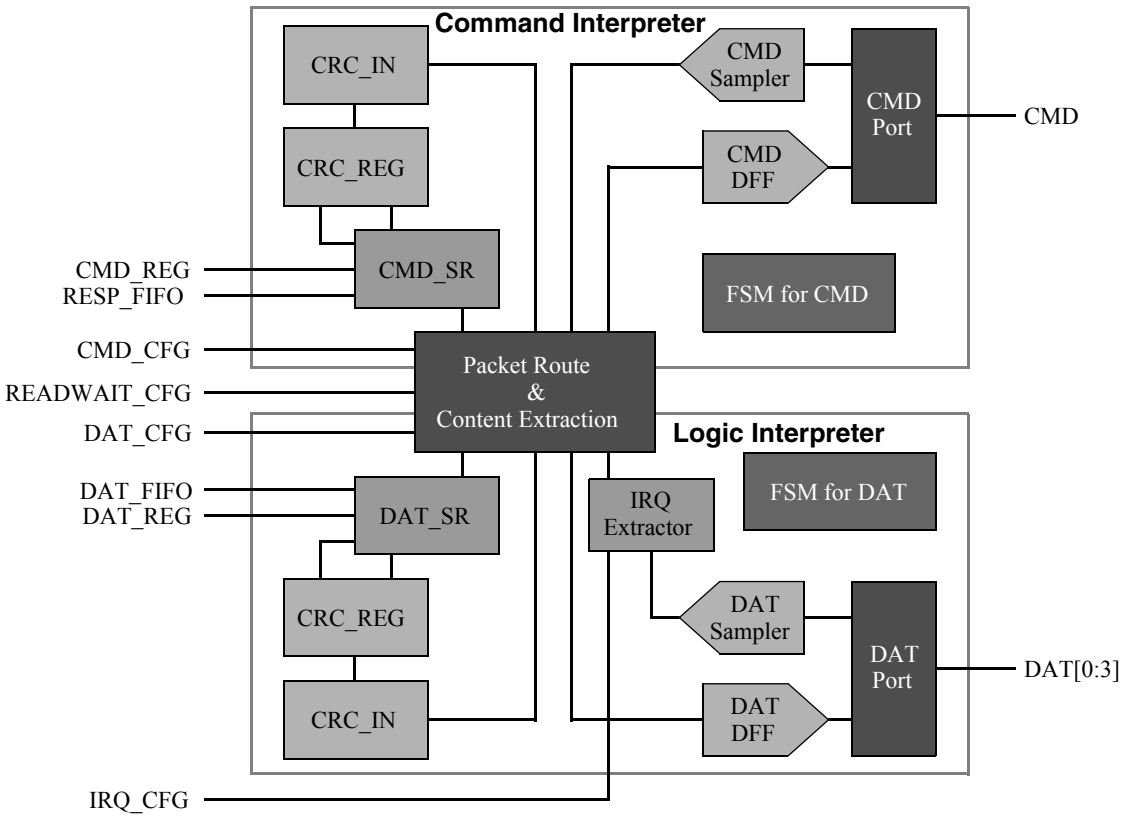


Figure 18-7. Block Diagram for Logic and Command Interpreters

The CRC Generation and Detection Block is one of the primary hardware functions in the module. To minimize the gate count, the internal command shift register is also used for the CRC shift register (see Figure 18-8 on page 18-11). The polynomials for CRC generation are:

For SD_CMD:

$$\text{Generator polynomial: } G(x) = x^7 + x^3 + 1$$
$$M(x) = (\text{first bit}) \bullet x^n + (\text{second bit}) \bullet x^{n-1} + \ldots + (\text{last bit}) \bullet x^0$$
$$\text{CRC [6:0]} = \text{Remainder } [(M(x) \bullet x^7) / G(x)] \qquad \textit{Eqn. 18-1}$$

For SD_DAT:

$$\text{Generator polynomial: } G(x) = x^{16} + x^{12} + x^5 + 1$$
$$M(x) = (\text{first bit}) \bullet x^n + (\text{second bit}) \bullet x^{n-1} + \ldots + (\text{last bit}) \bullet x^0$$
$$\text{CRC [15:0]} = \text{Remainder } [(M(x) \bullet x^{16}) / G(x)] \qquad \textit{Eqn. 18-2}$$

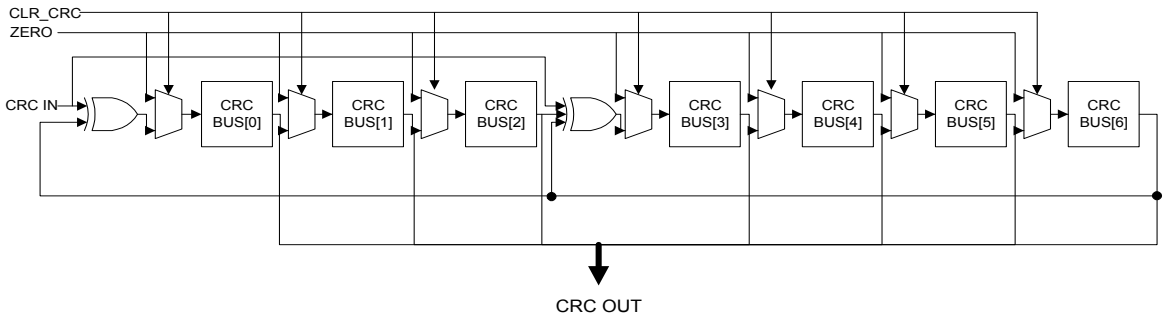


Figure 18-8. Command CRC Shift Register (SD_DAT Has a Similar Structure)

18.5.4 System Clock Controller

To maximize the power-saving capability of the MMC/SD module, two clock stages are used within. The input clock, PERCLK2, runs at a frequency between 20–100 MHz and passes through a prescaler to adjust and maintain the inner clock to under 20 MHz, the maximum operating frequency of the MMC/SD cards. The output of this prescaler is called CLK_20M. Only about 10% of the total circuit runs on this clock.

CLK_20M feeds into a user-programmable clock divider. The resulting clock is called CLK_DIV and it runs at a frequency between 0–20 MHz. This clock runs most of the circuitry in the module.

The prescaler and divider ratios are set in the MMC/SD Clock Rate Register (CLK_RATE).

The clocks are paused while they are not used, such as when the FIFO is full during the card read operation or when there are no further read operations to the FIFO.

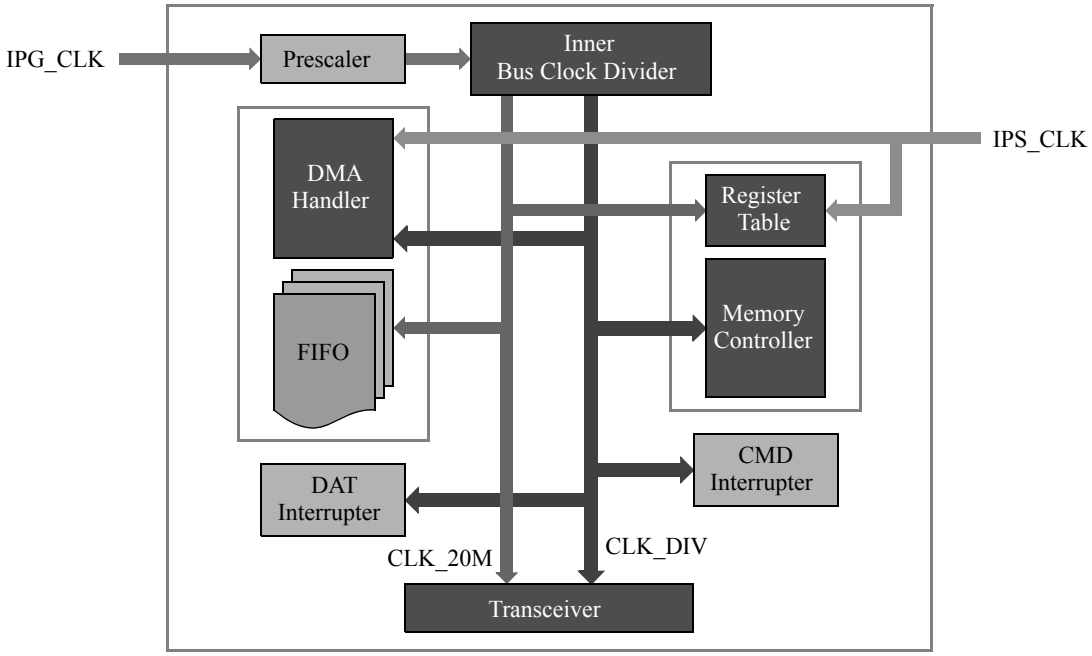


Figure 18-9. Clock Tree for the MMC/SD Module

The system clock controller sets the rate of the MMC/SD module main clock and checks whether it is on or off. The clock is turned off by setting the STOP_CLK bit in the MMC/SD Clock Control Register (STR_STP_CLK) and is turned on by setting the START_CLK bit. To change the clock rate, the application writes new prescaler and divider values in the CLK_RATE register.

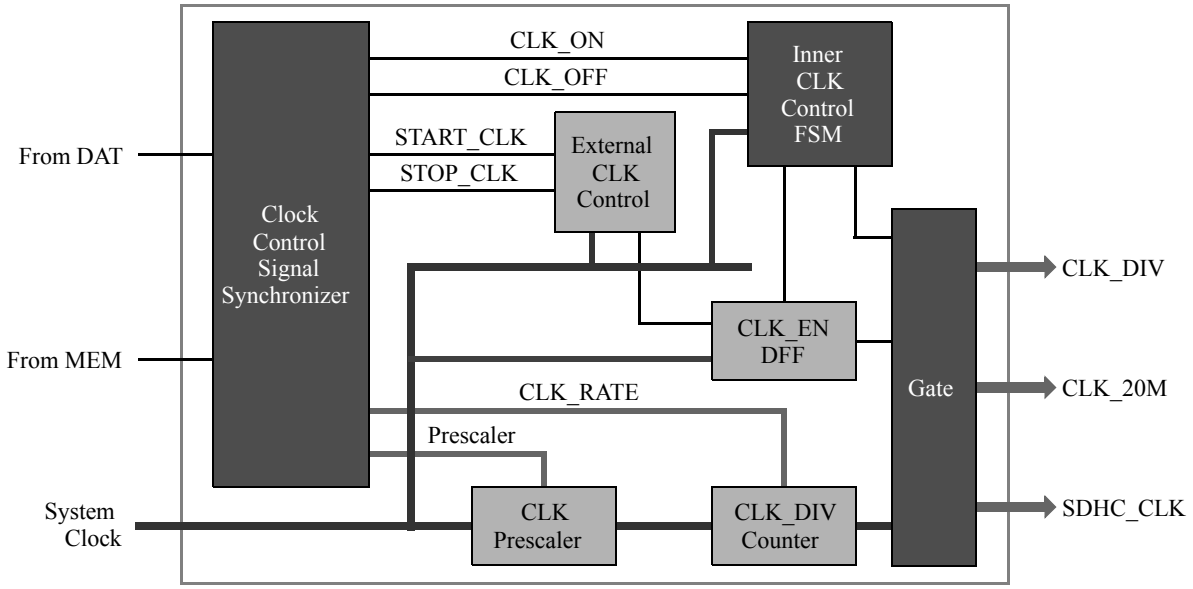


Figure 18-10. System Clock Control Unit

The clock controller also ensures that every clock stop occurs when all clocks are low so the clock division and enable/disable occur without generating a glitch in the process. The stop clock data unit enables the data state machine to stop the clock when the application is too slow and does not update the FIFO on time in a multiple block/stream write or read.

18.5.4.1 Card Clock Control

The MMC/SD host controller controls the clocking for the MMC/SD cards. The MMC/SD module can lower the clock frequency or even turn off the clock to the cards to save energy. The MMC/SD module provides at least 8 clock cycles after the last MMC/SD bus transaction (command, response, data, CRC) before shutting off the clock.

18.6 Programming Model

The MMC/SD Host Controller module includes fifteen 32-bit registers that the application configures before every operation on the multimedia bus. Table 18-4 summarizes these registers and their addresses.

Table 18-4. Multimedia Controller Register Memory Map

Description	Name	Address
MMC/SD Clock Control Register	STR_STP_CLK	0x00214000
MMC/SD Status Register	STATUS	0x00214004

Table 18-4. Multimedia Controller Register Memory Map (continued)

Description	Name	Address
MMC/SD Clock Rate Register	CLK_RATE	0x00214008
MMC/SD Command and Data Control Register	CMD_DAT_CONT	0x0021400C
MMC/SD Response Time-Out Register	RES_TO	0x00214010
MMC/SD Read Time-Out Register	READ_TO	0x00214014
MMC/SD Block Length Register	BLK_LEN	0x00214018
MMC/SD Number of Blocks Register	NOB	0x0021401C
MMC/SD Revision Number Register	REV_NO	0x00214020
MMC/SD Interrupt Mask Register	INT_MASK	0x00214024
MMC/SD Command Number Register	CMD	0x00214028
MMC/SD Higher Argument Register	ARGH	0x0021402C
MMC/SD Lower Argument Register	ARGL	0x00214030
MMC/SD Response FIFO Register	RES_FIFO	0x00214034
MMC/SD Buffer Access Register	BUFFER_ACCESS	0x00214038

18.6.1 MMC/SD Clock Control Register

The MMC/SD Clock Control Register allows the user to reset the system, enable the MMC/SD module, and control the MMC/SD module clock.

NOTE:

To perform a system soft-reset and an MMC/SD module enable, the user must program the MMC/SD Clock Control Register with a particular sequence of values. The programmer must first write the value 0x0008, then 0x000D, and then write 0x0005 eight times.

STR_STP_CLK MMC/SD Clock Control Register

Addr 0x00214000

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											ENDIAN		MMCSD_RESET	MMCSD_ENABLE	START_CLK	STOP_CLK
TYPE	r	r	r	r	r	r	r	r	r	r	rw	r	w	rw	w	w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 18-5. MMC/SD Clock Control Register Description

Name	Description	Settings
Reserved Bits 31–6	Reserved—These bits are reserved and should read 0.	
ENDIAN Bit 5	**ENDIAN FIFO Mode Selection**—Setting this bit selects the Endian mode of the FIFO.	0 = Little endian 1 = Big endian
Reserved Bit 4	Reserved—This bit is reserved and should read 0.	
MMCSD_RESET Bit 3	**MMCSD Reset**—Resets the MMC/SD module.	0 = No effect 1 = Reset the MMC/SD module
MMCSD_ENABLE Bit 2	**MMC/SD Enable**—Enables/Disables the MMC/SD module. When the module is disabled, the clock source and all internal operations are halted.	0 = Disable MMC/SD module 1 = Enable MMC/SD module
START_CLK Bit 1	**Start Clock**—Starts the MMC/SD module clock. The MMCSD_ENABLE bit must be set for START_CLK to have any meaning. When START_CLK is changed while the MMC/SD module is in a transmission period, the status of the operation determines when the clock is started. Otherwise, the clock begins immediately. Setting a value of 11 on Bits 1:0 is prohibited. **Note:** A transmission period is defined as the time from a card data-access-related command is submitted to the end of the data access operation.	0 = MMC/SD clock inactive 1 = MMC/SD clock active

Table 18-5. MMC/SD Clock Control Register Description (continued)

Name	Description	Settings
STOP_CLK Bit 0	**Stop Clock**—Stops the MMC/SD clock. The MMCSD_ENABLE bit must be set for STOP_CLK to have any meaning. When STOP_CLK is changed while the MMC/SD module is in a transmission period, the status of the operation determines when the clock is stopped. Otherwise, the clock halts immediately. Setting a value of 11 on Bits 1:0 is prohibited. **Note:** A transmission period is defined as the time from a card data-access-related command is submitted to the end of the data access operation.	0 = MMC/SD clock active 1 = MMC/SD clock stopped

18.6.2 MMC/SD Status Register

The read-only MMC/SD Status Register provides the programmer with information about the status of MMC/SD module operations, application FIFO status, and error conditions.

STATUS MMC/SD Status Register **Addr 0x00214004**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CARD _PRE-SENCE	SDIO_ INT_ ACTIVE	END_ CMD_ RESP	WRITE _OP_ DONE	DATA_ TRANS_ DONE	WR_CRC _ERROR _CODE		CARD _BUS _CLK _RUN	APPL _BUFF _FF	APPL _BUFF _FE	RESP _CRC _ERR		CRC_ READ _ERR	CRC_ WRITE _ERR	TIME _OUT _RESP	TIME _OUT _READ
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 18-6. MMC/SD Status Register Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
CARD_PRESENCE Bit 15	**Card Presence**—Detects whether an MMC/SD card is present based on SD_DAT [3].	0 = No cards are present 1 = At least 1 card is present
SDIO_INT_ACTIVE Bit 14	**SD I/O Interrupt Active**—Indicates whether an interrupt is detected at the SD I/O card. A separate acknowledge command to the card is required to clear this interrupt.	0 = No interrupt detected 1 = Interrupt detected via SDIO card bus

Table 18-6. MMC/SD Status Register Description (continued)

Name	Description	Settings
END_CMD_RESP Bit 13	**End Command Response**—Indicates that the command was successfully transmitted to the card. A response package is expected after each successful command operation. END_CMD_RESP is clear when the clock is stopped.	0 = Command not successful, incomplete, or not applicable (no response) 1 = Command transmitted successfully (response received)
WRITE_OP_DONE Bit 12	**Write Operation Done**—Indicates when an access operation is complete. The flash card needs extra idle time to write and it requires the MMC/SD module to wait until the card writes the buffered data to the inner flash memory. The MMC/SD module automatically detects the status. WRITE_OP_DONE determines the end of the write operation.	0 = Write operation in progress or incomplete 1 = Write operation complete
DATA_TRANS_DONE Bit 11	**Data Transfer Done**—Indicates that a data transfer is complete. DATA_TRANS_DONE determines the end of the read operation.	0 = Data transfer in progress or incomplete 1 = Data transfer complete
WR_CRC_ERROR_CODE Bits 10–9	**Write CRC Error Code**—Specifies whether there was a CRC error, and if so, what kind. The error code is valid only when a CRC error has occurred (Bit 5, RESP_CRC_ERR, is set).	00 = No transmission error, CRC status is 010 (positive) 01 = Transmission error, CRC status is 110 (negative) 10 = No CRC response 11 = Reserved
CARD_BUS_CLK_RUN Bit 8	**MMC/SD Card Clock Running**—Indicates whether the clock is running. The clock rate setting and system configuration can be modified when the clock is turned off by setting the STOP_CLK bit of the STR_STP_CLK register.	0 = MMC/SD clock is not running 1 = MMC/SD clock is running
APPL_BUFF_FF Bit 7	**Application Buffer FIFO Full**—Indicates the status of the 32 x 16-bit inner data FIFO. Usually used in the card read operation.	0 = Buffer is not full 1 = Buffer is full
APPL_BUFF_FE Bit 6	**Application Buffer FIFO Empty**—Indicates the status of the 32 x 16-bit inner data FIFO. Usually used in the card write operation.	0 = Buffer is not empty 1 = Buffer is empty
RESP_CRC_ERR Bit 5	**Response CRC Error**—Indicates that a transmission error occurred on the SD_CMD line during the command and response transfer. This type of error usually results from the electrical environment. **Note:** RESP_CRC_ERR is cleared only by an internal status change or by removing the source of the error.	0 = No error 1 = Response CRC error occurred
Reserved Bit 4	Reserved—This bit is reserved and should read 0.	

Table 18-6. MMC/SD Status Register Description (continued)

Name	Description	Settings
CRC_READ_ERR Bit 3	**CRC Read Error**—Indicates that a transmission error occurred on the SD_DAT line during a card read. The user should re-try the transmission. **Note:** CRC_READ_ERR is cleared only by an internal status change or by removing the source of the error.	0 = No error 1 = CRC read error occurred
CRC_WRITE_ERR Bit 2	**CRC Write Error**—Indicates that a transmission error occurred on the SD_DAT line during a card write. See the WR_CRC_ERROR_CODE field for more information. **Note:** CRC_WRITE_ERR is cleared only by an internal status change or by removing the source of the error.	0 = No error 1 = CRC write error occurred
TIME_OUT_RESP Bit 1	**Time-Out Response Error**—Indicates that a command response is not received in the time specified in the RES_TO register. This can be caused by: • An unsupported command received at the card(s) • Another MMC/SD_OP_COND command submitted after all cards have sent their voltage ranges and the power-up routine is finished • An identification command issued when all cards are already in standby state • No card is on the bus **Note:** TIME_OUT_RESP is cleared only by an internal status change or by removing the source of the error.	0 = No error 1 = Time-out response error occurred
TIME_OUT_READ Bit 0	**Time-Out Read Data Error**—Indicates that the expected data from the card is not received in the time specified in the READ_TO register. TIME_OUT_READ is cleared only by an internal status change or by removing the source of the error.	0 = No error 1 = Time-out read data error occurred

18.6.3 MMC/SD Clock Rate Register

The MMC/SD Clock Rate Register defines the clock divider values for CLK_20M and CLK_DIV. These two signals are used to clock the MMC/SD module. See Section 18.5.4, “System Clock Controller,” for more information on clock schemes.

NOTE:

The user can write to the MMC/SD Clock Rate Register only when the bus clock is stopped. Check the CARD_BUS_CLK_RUN bit of the STATUS register to determine the state of the bus clock.

CLK_RATE MMC/SD Clock Rate Register **Addr 0x00214008**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											PRESCALER			CLK RATE		
TYPE	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0
	0x0036															

Table 18-7. MMC/SD Clock Rate Register Description

Name	Description	Settings
Reserved Bits 31–6	Reserved—These bits are reserved and should read 0.	
PRESCALER Bits 5–3	**Prescaler Divider**—Specifies the divider value to generate CLK_20M. PERCLK2 feeds into the MMC/SD module from the clock controller.	000 = CLK_20M is PERCLK2 001 = CLK_20M is PERCLK2 ÷ 2 010 = CLK_20M is PERCLK2 ÷ 3 (recommended) 011 = CLK_20M is PERCLK2 ÷ 3 100 = CLK_20M is PERCLK2 ÷ 4 101 = CLK_20M is PERCLK2 ÷ 5 (recommended) 110 = CLK_20M is PERCLK2 ÷ 5 111 = CLK_20M is PERCLK2 ÷ 5

Table 18-7. MMC/SD Clock Rate Register Description (continued)

Name	Description	Settings
CLK RATE Bits 2–0	**Inner Bus Clock Divider**—Specifies the divider value to generate CLK_DIV. The CLK_20M signal feeds into the inner bus clock divider from the prescaler.	000 = CLK_DIV is CLK_20M 001 = CLK_DIV is CLK_20M ÷ 2 010 = CLK_DIV is CLK_20M ÷ 4 011 = CLK_DIV is CLK_20M ÷ 8 100 = CLK_DIV is CLK_20M ÷ 16 101 = CLK_DIV is CLK_20M ÷ 32 110 = CLK_DIV is CLK_20M ÷ 64 111 = CLK_DIV is CLK_20M ÷ 128

18.6.4 MMC/SD Command and Data Control Register

The MMC/SD Command and Data Control Register allows the user to specify the format of data and response, and to control the ReadWait cycle.

CMD_DAT_CONT MMC/SD Command and Data Control Register **Addr 0x0021400C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				CMD_ RESP_ LONG_ OFF	STOP_ READ WAIT	START_ READ WAIT	BUS_ WIDTH		INIT	BUSY	STREAM _BLOCK	WRITE _READ	DATA_ ENABLE	FORMAT_ OF_ RESPONSE		
TYPE	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 18-8. MMC/SD Command and Data Control Register Description

Name	Description	Settings
Reserved Bits 31–13	Reserved—These bits are reserved and should read 0.	
CMD_RESP_LONG_OFF Bit 12	**Command Response Long Off**—Allows bit clearance when status is read. used in ReadWait cycle.	0 = Bit not cleared when read 1 = Allows bit clearance
STOP_READWAIT Bit 11	**Stop ReadWait**—Ends the ReadWait cycle.	0 = No effect 1 = Ends cycle

Table 18-8. MMC/SD Command and Data Control Register Description (continued)

Name	Description	Settings
START_READWAIT Bit 10	**Start ReadWait**—Starts the ReadWait cycle.	0 = No effect 1 = Starts cycle
BUS_WIDTH Bits 9–8	**Bus Width**—Specifies the width of the data bus.	00 = 1-bit 01 = Reserved 10 = 4-bit 11 = Reserved
INIT Bit 7	**Initialize**—Specifies whether the optional 80 clock cycle prefix (to initialize the card) will occur before every command. INIT enables/disables the 80 clock initialization time.	0 = Disable 80 clocks 1 = Enable 80 clocks
BUSY Bit 6	**Busy Signal**—Specifies whether a busy signal is expected after the current command. A busy signal occurs when the SD_DAT line is pulled low while there are no free data receive buffers. Only SEND_STATUS (CMD13) and SELECT/DESELECT_CARD (CMD7) are used during busy. A busy signal is expected after the stop transmission, card select, erase, and program CID commands. Normally used in write-related operations.	0 = No busy signal expected 1 = Busy signal expected after current command
STREAM_BLOCK Bit 5	**Stream or Block**—Specifies whether the data transfer of the current command is in stream or block mode.	0 = Block mode 1 = Stream mode
WRITE_READ Bit 4	**Write or Read**—Specifies whether the data transfer of the current command is a write or read operation.	0 = Read 1 = Write
DATA_ENABLE Bit 3	**Data Transfer**—Specifies whether the current command includes a data transfer.	0 = No data transfer included 1 = Data transfer included
FORMAT_OF _RESPONSE Bits 2–0	**Format of Response**—Sets the response format. See Section 18.7.8.5, “Response Formats.”	000 = No response 001 = Format R1 010 = Format R2 011 = Format R3 100 = Format R4 101 = Format R5 110 = Format R6

18.6.5 MMC/SD Response Time-Out Register

The MMC/SD Response Time-Out Register defines the time-out error for a received response.

RES_TO MMC/SD Response Time-Out Register **Addr 0x00214010**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									RESPONSE TIME OUT							
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
								0x0040								

Table 18-9. MMC/SD Response Time-Out Register Description

Name	Description	Settings
Reserved Bits 31–8	Reserved—These bits are reserved and should read 0.	
RESPONSE TIME OUT Bits 7–0	**Response Time-Out**—Specifies the number of clock counts between the command and when the MMC/SD module turns on the time-out error for the received response.	0x01 = 1 clock counts ... 0xFF = 255 clock counts

18.6.6 MMC/SD Read Time-Out Register

The MMC/SD Read Time-Out Register defines the time-out error for received data.

READ_TO MMC/SD Read Time-Out Register **Addr 0x00214014**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA READ TIME OUT															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0xFFFF															

Table 18-10. MMC/SD Read Time-Out Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
DATA READ TIME OUT Bits 15–0	**Received Data Time-Out**—Specifies the number of clocks between the command and when the MMC/SD module turns on the time-out error for the received data. The unit is CLK_20M ÷ 256. A value of 0x2DB4 is recommended.

18.6.7 MMC/SD Block Length Register

The MMC/SD Block Length Register defines how many bytes are in a block.

BLK_LEN MMC/SD Block Length Register **Addr 0x00214018**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							BLOCK LENGTH									
TYPE	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 18-11. MMC/SD Block Length Register Description

Name	Description	Settings
Reserved Bits 31–10	Reserved—These bits are reserved and should read 0.	
BLOCK LENGTH Bits 9–0	**Block Length**—Specifies the number of bytes in a block, and is normally set to 0x200 for MMC/SD module data transactions. The value is specified in the card's CSD **Note:** This version of Host Controller only supports fixed block lengths of 0x200.	0x000 = 0 byte 0x001 = 1 byte ... 0x3FF = 1023 bytes

18.6.8 MMC/SD Number of Blocks Register

The MMC/SD Number of Blocks Register defines the number of blocks in a data transfer.

NOB MMC/SD Number of Blocks Register **Addr 0x0021401C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NOB															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 18-12. MMC/SD Number of Blocks Register Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
NOB Bits 15–0	**Block Length**—Specifies the number of blocks in a data transfer. One block is a possibility.	0x0000 = 0 block 0x0001 = 1 block ... 0xFFFF = 65535 blocks

18.6.9 MMC/SD Revision Number Register

The read-only MMC/SD Revision Number Register is a read-only register that displays the revision number of the module.

REV_NO MMC/SD Revision Number Register **Addr 0x00214020**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	REVISION NUMBER															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0

Table 18-13. MMC/SD Revision Number Register Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
REVISION NUMBER Bits 15–0	**Module Revision Number**—Specifies the revision number of the MMC/SD module.	Fixed at

18.6.10 MMC/SD Interrupt Mask Register

When certain conditions exist in the module, the MMC/SD module has the ability to set an interrupt. The MMC/SD Interrupt Mask Register allows the user to control whether these interrupts occur.

Rewriting this register clears the interrupt $\overline{\text{MMC_IRQ}}$. When the interrupt source is from the SD I/O, the MMC/SD module continues to interrupt the system. In this case, the user must write to the internal registers on the SD I/O card. This is the only way to acknowledge the interrupt. In the interrupt service routine the status indicators should be polled.

INT_MASK MMC/SD Interrupt Mask Register **Addr 0x00214024**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
										0x0000						

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										AUTO_ CARD_ DETECT	DAT0 _EN	SDIO	BUF_ READY	END_ CMD_ RES	WRITE _OP_ DONE	DATA_ TRAN
TYPE	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
										0x0000						

Table 18-14. MMC/SD Interrupt Mask Register Description

Name	Description	Settings
Reserved Bits 31–7	Reserved—These bits are reserved and should read 0.	
AUTO_CARD_DETECT Bit 6	**Auto Card Detect (SD card only)**—Masks the auto card detect interrupt. Card insertion results in the SD_DAT [3:0] pins changing from 0111 to 1111, and card removal results in the pins changing from 1111 to 0111. After the card is detected, the user program must mask AUTO_CARD_DETECT to avoid misleading interrupt generation while the SD_DAT lines change during card access.	0 = Not masked 1 = Masked
DAT0_EN Bit 5	**SD_DAT0 Enable**—Identifies how the SD I/O interrupt is detected. An interrupt is determined by SD_DAT [1] = 0, however this bit is an optional setting for the SDIO bit.	0 = SD I/O's Interrupt detection based on: SD_DAT [3:1] = 110 1 = SD I/O's Interrupt detection based on: SD_DAT [3:0] = 1101
SDIO Bit 4	**MMC/SD I/O**—Masks the interrupt from the SD I/O card to the MMC/SD module I/O interrupt mask.	0 = Not masked 1 = Masked

Table 18-14. MMC/SD Interrupt Mask Register Description (continued)

Name	Description	Settings
BUF_READY Bit 3	**Buffer Ready**—Masks the Application buffer FIFO ready (full or empty) interrupt.	0 = Not masked 1 = Masked
END_CMD_RES Bit 2	**End Command Response**—Masks the end command response interrupt.	0 = Not masked 1 = Masked
WRITE_OP_DONE Bit 1	**Write Operation Done**—Masks the write operation done interrupt.	0 = Not masked 1 = Masked
DATA_TRAN Bit 0	**Data Transfer Done**—Masks the data transfer done interrupt.	0 = Not masked 1 = Masked

Table 18-15 summarizes the interrupt mechanisms used in the MMC/SD module.

Table 18-15. Interrupt Mechanisms

Int #	STATUS Register Source Bit Name (Number)	Generate Interrupt?	INT_MASK Register Bit Name (Number)	Interrupt/ Status Clear Method
1	TIME_OUT_READ (0)	No, alert via the DATA_TRANS_DONE bit in the MMC/SD Status Register	DATA_TRAN (0)	Clear status by setting STOP_CLK in STR_STP_CLK
2	TIME_OUT_RESP (1)	No, alert via the END_CMD_RESP bit in the MMC/SD Status Register	END_CMD_RES (2)	Clear status by setting STOP_CLK in STR_STP_CLK
3	CRC_WRITE_ERR (2)	No, alert via the DATA_TRANS_DONE bit in the MMC/SD Status Register	DATA_TRAN (0)	Clear status by setting STOP_CLK in STR_STP_CLK
4	CRC_READ_ERR (3)	No, alert via the DATA_TRANS_DONE bit in the MMC/SD Status Register	DATA_TRAN (0)	Clear status by setting STOP_CLK in STR_STP_CLK
5	RESP_CRC_ERR (5)	No, alert via the END_CMD_RESP bit in the MMC/SD Status Register	END_CMD_RES (2)	Clear status by setting STOP_CLK in STR_STP_CLK
6	APPL_BUFF_FE (6)	Yes	BUF_READY (3)	Clear interrupt by writing to INT_MASK bit Clear status by writing to BUFFER_ACCESS

Table 18-15. Interrupt Mechanisms (continued)

Int #	STATUS Register Source Bit Name (Number)	Generate Interrupt?	INT_MASK Register Bit Name (Number)	Interrupt/ Status Clear Method
7	APPL_BUFF_FF (7)	Yes	BUF_READY (3)	Clear interrupt by writing to INT_MASK bit Clear status by reading BUFFER_ACCESS
8	DATA_TRANS_DONE (11)	Yes	DATA_TRAN (0)	Clear interrupt by writing to INT_MASK bit Clear status by setting STOP_CLK in STR_STP_CLK
9	WRITE_OP_DONE (12)	Yes	WRITE_OP_DONE (1)	Clear interrupt by writing to INT_MASK bit Clear status by setting STOP_CLK in STR_STP_CLK
10	END_CMD_RESP (13)	Yes	END_CMD_RES (2)	Clear interrupt by writing to INT_MASK bit Clear status by setting STOP_CLK in STR_STP_CLK
11	SDIO_INT_ACTIVE (14)	Yes	SDIO (4)	Clear interrupt by writing to INT_MASK bit; Clear status by resolving interrupt source on SD I/O card (requires separate acknowledge command to SD I/O card)
12	CARD_PRESENCE (15)	Yes	AUTO_CARD_DETECT (6)	Clear interrupt by writing to INT_MASK bit Clear status by removing card (this is a status indicator, not an error)

18.6.11 Commands and Arguments

The MMC/SD module communicates with the MMC/SD card(s) by sending commands and arguments. The command to send is set in the MMC/SD Command Number Register (CMD), and the argument is defined in two registers, the MMC/SD Higher Argument Register (ARGH) and the MMC/SD Lower Argument Register (ARGL). The full list of commands is shown in Table 18-25 on page 18-58.

18.6.11.1 MMC/SD Command Number Register

CMD MMC/SD Command Number Register **Addr 0x00214028**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											COMMAND NUMBER					
TYPE	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 18-16. MMC/SD Command Number Register Description

Name	Description	Settings
Reserved Bits 31–6	Reserved—These bits are reserved and should read 0.	
COMMAND NUMBER Bits 5–0	**Command Number**—Specifies the command number to be executed.	0x00 = CMD0 0x01 = CMD1 ... 0x3F = CMD63

18.6.11.2 MMC/SD Higher Argument Register

ARGH MMC/SD Higher Argument Register **Addr 0x0021402C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ARGUMENT HIGH															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 18-17. MMC/SD Higher Argument Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
ARGUMENT HIGH Bits 15–0	**Higher Argument**—Specifies the higher word of the argument for the current command.

18.6.11.3 MMC/SD Lower Argument Register

ARGL MMC/SD Lower Argument Register **Addr 0x00214030**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ARGUMENT LOW															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 18-18. MMC/SD Lower Argument Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
ARGUMENT LOW Bits 15–0	**Lower Argument**—Specifies the lower word of the argument for the current command.

18.6.12 MMC/SD Response FIFO Register

The read-only MMC/SD Response FIFO Register holds the response sent back to the MMC/SD module after every command. The size of this FIFO is 8 × 32-bits, however only bits 15-0 are valid data.

NOTE:

The MMC/SD module does not copy the last 8 bits of CID/CSD into this FIFO, which is its CRC and stop bit, because the incoming CRC is automatically validated by the module CRC check mechanism during receive.

RES_FIFO MMC/SD Response FIFO Register **Addr 0x00214034**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RESPONSE CONTENT															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 18-19. MMC/SD Response FIFO Register Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
RESPONSE CONTENT Bits 15–0	**Response Content**—Contains the responses to every command that is sent by the MMC/SD module. This size of this FIFO register is 8x16-bit.	

18.6.13 MMC/SD Buffer Access Register

The MMC/SD Buffer Access Register contains the data transmitted to or received from the card during data access operations.

BUFFER_ACCESS MMC/SD Buffer Access Register **Addr 0x00214038**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FIFO CONTENT															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 18-20. MMC/SD Buffer Access Register Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
FIFO CONTENT Bits 15–0	**FIFO Content**—Holds transfer or receive data between system memory and card memory. The size of the FIFO buffer is 8 x16-bit in 1-bit mode, and 32 x16-bit in 4-bit mode. FIFO Full or Empty active $\overline{\text{MMC_IRQ}}$ and $\overline{\text{MMC_DREQ}}$. Only a completed FIFO read or write can clear the $\overline{\text{MMC_DREQ}}$ request.	

18.7 Functional Example for the MMC/SD Module

All communication between the MMC/SD module and the cards is controlled by the MMC/SD module. The MMC/SD module sends commands of two types: broadcast and addressed (point-to-point) commands.

Broadcast commands are intended for all cards. These include commands such as GO_IDLE_STATE (CMD0), SEND_OP_COND (CMD1), ALL_SEND_CID (CMD2), and SET_RELATIVE_ADDR (CMD3), that are broadcast out to all cards to go to a common condition. With broadcast-type commands, all cards are in open-drain mode to avoid bus contention.

After the broadcast command SET_RELATIVE_ADDR is issued, cards enter standby state, and only addressed commands are used. In this state, the pins SD_CMD and SD_DAT [3:0] operate in push-pull mode for maximum drive at maximum operation frequency.

Addressed commands are sent to the card selected by SELECT/DESELECT_CARD (CMD7) and require a response from that card.

The MMC and the SD are similar products and with the exception of the 4x bandwidth and the built-in encryption, they are programmed similarly. The following sections illustrate how to initialize, access, and protect the cards.

18.7.1 Basic Operation

Code Example 18-2 on page 18-33 is the program flow used to submit a command to the card(s), <command_no> is the targeted command, <argh_no,argl_no> are the corresponding arguments, <cmd_dat_cont> is the command configuration required, and <int_mask_value> is the interrupt mask in the user program.

Code Example 18-2. Send_Cmd_Wait_Resp

```
send_cmd_wait_resp(command_no, argh_no, argl_no, cmd_dat_cont, int_mask_value)
            {
   write_reg(COMMAND, <command_no>);
   write_reg(ARGH, <argh_no>);
   write_reg(ARGL, <argl_no>);
   write_reg(CMD_DAT_CONT, <cmd_dat_cont>);
   write_reg(STR_STP_CLK, 0x6);
   read_reg(STATUS);
   while(!STATUS[8]) Read_reg(STATUS);        l// Wait until clock is started
                                              //     to submit commands
   while(irq_status);                         // Wait interrupt generated
                                              //     (End Command Response)
   Write_reg(INT_MASK, <int_mask_value>);
   read_reg(STATUS);                          // Check whether interrupt is an
                                              //     End Command Response or a Time
                                              //     out.
   write_reg(STR_STP_CLK, 0x5);
   read_reg(STATUS);
   while(STATUS[8]) Read_reg(STATUS);         // Wait until clock is stopped;
                                              //     command - response end.
      }
```

18.7.2 Card Identification State

All data communication during the card identification state uses the command line (SD_CMD) only.

While in card identification state, the MMC/SD module performs the following steps:

1. Detects the cards
2. Resets all cards that are in the card identification state
3. Validates operation voltage range
4. Identifies the cards
5. Asks each card (separately, on its own SD_CMD line) to publish its relative card address (RCA)

18.7.2.1 Card Detect

The Code Example 18-3 demonstrates how to detect a card via the MMC/SD module.

Code Example 18-3. Card_Detect

```
card_detect()
            {
            while(irq_status);                // Wait interrupt generated (Card
                                              //     Presence)
            while(!STATUS[15]) Read_reg(STATUS); // Wait until card is detected
            Write_reg(INT_MASK, 0x40);
            }
```

18.7.2.2 Reset

There are three types of reset available for the MMC/SD module:

- **Hardware reset**—Resets both the card(s) and the MMC/SD module, driven by power on reset (POR).
- **Software reset**—Resets the MMC/SD module only. This is controlled via the MMC/SD Clock Control Register (STR_STP_CLK). See Section 18.6.1, “MMC/SD Clock Control Register,” for more information on software resets.
- **Card reset**—Resets the cards only. GO_IDLE_STATE (CMD0) sets the MMC and SD memory cards into idle state. IO_RW_DIRECT (CMD52) resets the SD I/O Card. The cards are initialized with a default relative card address (RCA = 0x0001) and with a default driver stage register setting (lowest speed, highest driving current capability).

Code Example 18-4. Software_Reset

```
software_reset()
            {
            write_reg(STR_STP_CLK, 0x8);
            write_reg(STR_STP_CLK, 0xd);
            write_reg(STR_STP_CLK, 0x5);
            write_reg(STR_STP_CLK, 0x5);
            write_reg(STR_STP_CLK, 0x5);
            write_reg(STR_STP_CLK, 0x5);
            write_reg(STR_STP_CLK, 0x5);
            write_reg(STR_STP_CLK, 0x5);
            write_reg(STR_STP_CLK, 0x5);
            write_reg(STR_STP_CLK, 0x5);
            write_reg(CLK_RATE, 0x3F);        // Set the lowest clock for
                                              //     initialization
            write_reg(READ_TO, 0x2DB4);
            send_cmd_wait_resp(CMD_GO_IDLE_STATE, 0x0, 0x0, 0x80, 0x40);
            }
```

18.7.2.3 Voltage Validation

All cards can communicate with the MMC/SD module using any operating voltage within the specification range. The supported minimum and maximum Vdd values are defined in the Operation Conditions Register (OCR) on the card.

Cards that store the card identification number (CID) and card-specific data (CSD) in the payload memory are able to communicate this information only under data transfer Vdd conditions. When the MMC/SD module and card have incompatible Vdd ranges, the card is not able to complete the identification cycle and cannot send CSD data. For this purpose, the special commands SEND_OP_COND (CMD1), SD_APP_OP_COND (ACMD41 for SD Memory), and IO_SEND_OP_COND (CMD5 for SD I/O) are designed to provide a mechanism to identify and reject cards that do not match the Vdd range desired by the MMC/SD module. The MMC/SD module sends the required Vdd voltage window as the operand of these commands. Cards that cannot perform data transfer in the specified range disconnect from the bus and go to inactive state.

By uses these commands without including the voltage range as the operand, the MMC/SD module can query each card and determine the common voltage range before sending out-of-range cards into the inactive state. This query is used when the MMC/SD module is able to select a common voltage range or when the user requires notification that cards are not usable.

Code Example 18-5. Voltage_Validation

```
voltage_validation(voltage_range_h, voltage_range_l,voltage_set_h,voltage_set_l)
  {
    send_cmd_wait_resp(IO_RW_DIRECT, 0x8800, 0x0608, 0x05, 0x40);
                    // Reset IO card
    send_cmd_wait_resp(IO_SEND_OP_COND, 0x0, 0x0, 0x04, 0x40);
                    // Send SDIO operation voltage command
    if (End Command Response true & No. of IO functions> 0)
                                            // it is an SDIO card
    {
      IORDY = 0;
      while (!(IORDY in I/O ORC))        // set voltage range
      {
      send_cmd_wait_resp(IO_SEND_OP_COND,voltage_set_h,voltage_set_l,0x04,0x40);
      }
      if(Memory Present flag true)
        Card = combo;                    // i.e. SDIO + Memory
      else
        Card = sdio;
    }
    else                                    // SD or MMC
    {
      send_cmd_wait_resp(APP_CMD, voltage_range_h, voltage_range_l, 0x03, 0x40);

                                            // MMC, SD reset
      send_cmd_wait_resp(APP_CMD, 0x0, 0x0, 0x01, 0x40);
                                            // Application Command follows
      if(End Command Response true)
      {
        send_cmd_wait_resp(SEND_OP_COND, 0x0, 0x0, 0x01, 0x40);
                                            // SD card found
        while(!(card init finished))
        {
          send_cmd_wait_resp(APP_CMD, 0x0, 0x0, 0x01, 0x40);
          send_cmd_wait_resp(SEND_OP_COND, voltage_range_h, voltage_set_l,0x03,0x40);
        }
        Card = sd;
      }
      else
      {
      send_cmd_wait_resp(SEND_OP_COND, voltage_range_h, voltage_range_l, 0x03, 0x40);

                                            // MMC card found
        if(End Command Response true)
        {
          Card = mmc;
          while(!(card init finished))
          {
            send_cmd_wait_resp(SEND_OP_COND,voltage_set_h,voltage_set_l,0x03,0x40);

          }
        }
        else
        {
          Card = No card or failed contact;
        }
      }
    }
  }
```

18.7.2.4 Card Registry

Card registry differs for the MMC and the SD cards. For the MMC cards, the identification process starts at clock rate F_{od}. The SD_CMD line output drivers are open-drain and allow parallel card operation during this process. The registration process is accomplished as follows:

1. The bus is activated.
2. The MMC/SD module broadcasts SEND_OP_COND (CMD1) to receive operation conditions.
3. The response is the wired AND operation of the Operation Condition Registers from all cards.
4. The incompatible cards are set to inactive state.
5. The MMC/SD module broadcasts ALL_SEND_CID (CMD2) to all active cards.
6. The active cards simultaneously send CID numbers serially. Cards with outgoing CID bits that do not match the bits on the command line stop transmitting and must wait for the next identification cycle. One card successfully transmits a full CID to the MMC/SD module and goes into identification state.
7. The MMC/SD module issues SET_RELATIVE_ADDR (CMD3) to that card. This new address is called the relative card address (RCA); it is shorter than the CID and addresses the card. The assigned card changes to standby state, it does not react to further identification cycles, and its output switches from open-drain to push-pull.
8. The MMC/SD module repeats steps 5 through 7 until it receives a time-out condition.

For the SD card, the identification process starts at clock rate F_{od}, and the SD_CMD line output drives are push-pull drivers instead of open-drain. The registration process is accomplished as follows:

1. The bus is activated.
2. The MMC/SD module broadcasts SD_APP_OP_COND (ACMD41).
3. The cards respond with the content of their Operation Condition Registers.
4. The incompatible cards are set to inactive state.
5. The MMC/SD module broadcasts ALL_SEND_CID (CMD2) to all active cards.
6. The cards send back their unique card identification numbers (CIDs) and go into identification state.
7. The MMC/SD module issues SET_RELATIVE_ADDR (CMD3) to an active card with an address. This new address is called the relative card address (RCA); it is shorter than the CID and addresses the card. The assigned card changes to standby state. The MMC/SD module can re-issue this command to change the RCA. The RCA of the card is the last assigned value.
8. The MMC/SD module repeats steps 5 through 7 with all active cards.

Code Example 18-6. Card_Registry

```
card_registry()
  {
    while(ResponseTO from STATUS)
    {
      if(card==combo or sdio)
      {
        send_cmd_wait_resp(ALL_SEND_CID, 0x00, 0x00, 0x02, 0x40);
        send_cmd_wait_resp(SET_RELATIVE_ADDR, 0x00, 0x00, 0x01, 0x40);
        rca = SDIO_RCA = address from response FIFO;
      }
      else if(card==sd)
      {
        send_cmd_wait_resp(ALL_SEND_CID, 0x00, 0x00, 0x02, 0x40);
        send_cmd_wait_resp(SET_RELATIVE_ADDR, 0x00, 0x00, 0x01, 0x40);
        rca = SD_RCA = address from response FIFO;
      }
      else if(card==mmc)
      {
        send_cmd_wait_resp(ALL_SEND_CID, 0x00, 0x00, 0x02, 0x40);
        rca = MMC_RCA = 0x1;
        send_cmd_wait_resp(SET_RELATIVE_ADDR, MMC_RCA, 0x00, 0x01, 0x40);
      }
      else
        exit due to card not identified;
    }
    send_cmd_wait_resp(SELECT_CARD, rca, 0x00, 0x41, 0x40);
  }
```

18.7.3 Card Access

This section discusses the MMC/SD card access with block write and read, stream write and read, erase and wide bus selection and deselection.

18.7.3.1 Block Access: Block Write and Block Read

The module supports both block write and block read operation to the card. Block Mode data transfers are grouped as 512 byte package/block and they are protected by CRC.

18.7.3.1.1 Block Write

During the block write commands (CMD24–CMD27), one or more blocks of data are transferred from the MMC/SD module to the card with a CRC appended to the end of each block by the MMC/SD module. When the CRC fails, the card indicates the failure on the SD_DAT line, the transferred data is discarded and not written, and all further transmitted blocks (in multiple block write mode) are ignored. A card supporting block write is always able to accept a block of data of the size defined by WRITE_BL_LEN.

When the MMC/SD module uses partial blocks with accumulated lengths that are not block aligned and block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card detects the block alignment error and aborts programming before the beginning of the first misaligned block. The card sets the ADDRESS_ERROR bit in the Card Status Register, ignores all further data transfers, and waits in the receive state for a stop command. The write operation is aborted when the MMC/SD module tries to write over a write protected area. In this case, the card sets the WP_VIOLATION bit in the Card Status Register.

Block write commands are used to program the CID and CSD registers which do not require a previous block length setting. The transferred data is CRC protected.

When a part of the CSD or CID register is stored in ROM, this unchangeable part must match the corresponding part of the receive buffer or the card reports an error and does not change any register contents.

Some cards can require long and unpredictable times to write a block of data. After receiving a block of data and completing the CRC check, the card begins writing. When the write buffer is full and unable to accept new data, the card holds the SD_DAT line(s) low. The MMC/SD module can poll the status of the card by sending a SEND_STATUS (CMD13) at any time, and the card responds with its status. The READY_FOR_DATA bit in the Card Status Register indicates whether the card can accept new data or when a write is still in progress. The MMC/SD module can deselect the card by issuing SELECT/DESELECT_CARD (CMD7), which displaces the card into the disconnect state and releases the SD_DAT line(s) without interrupting the write operation. When the card is reselected using SELECT/DESELECT_CARD (CMD7), when programming is still in progress and the write buffer is unavailable, the SD_DAT line(s) are pulled low. Code Example 18-7 provides the program code for the block write with DMA.

Code Example 18-7. Block_Write with DMA

```
block_write(rca, nob, addr_h, addr_l, buswidth)
            {
                  send_cmd_wait_resp(SEND_STATUS, rca, 0x00, 0x01, 0x40);
                  while(!Ready for data in card status is true)
                  {
                        send_cmd_wait_resp(SEND_STATUS, rca, 0x00, 0x01, 0x40);
                  }
                  write_reg(NOB, <nob>);
                  send_cmd_wait_resp(SET_BLOCKLEN, 0x00, 0x0200, 0x01, 0x40);
                  if(buswidth==4-bit mode)
                  {
                        send_cmd_wait_resp(APP_CMD, rca, 0x0, 0x01, 0x40);
                        send_cmd_wait_resp(SET_BUS_WIDTH, 0x00, 0x02, 0x01, 0x40);
                  }

                        // Configure the DMA for FIFO write operation (BUFFER_ACCESS,
                        // SDRAM_ADDR, nob);

                        // Set DMA source address = SDRAM_ADDR
                        // Set DMA target address = BUFFER_ACCESS
                        // Set DMA total byte transfer = nob
                        // Set DMA burst depth = 8 if 1-bit mode, = 32 if 4-bit mode

                  if(nob==1)
                  { if(buswidth=4-bit mode)
                        send_cmd_wait_resp(WRITE_SINGLE_BLOCK, addr_h, addr_l, 0x219,
0x40);
                  else
                        send_cmd_wait_resp(WRITE_SINGLE_BLOCK, addr_h, addr_l, 0x19,
0x40);
                  }
                  else
                  { if(buswidth=4-bit mode)
                        send_cmd_wait_resp(WRITE_MULTIPLE_BLOCK, addr_h, addr_l, 0x219,
0x40);
                  else
                        send_cmd_wait_resp(WRITE_MULTIPLE_BLOCK, addr_h, addr_l, 0x19,
0x40);
                  }
                  while(!FIFO empty in STATUS is true);
                  Enable DMA operation;
                  while(!Access Operation Done in STATUS true);
                  while(!card bus is stop);
                  if(nob > 1)
                  {
                  send_cmd_wait_resp(STOP_TRANS, 0x00, 0x00, 0x1, 0x40);
                  }
            }
```

Code Example 18-8 on page 18-41 provides the program code for the block write with polling.

Code Example 18-8. Block_Write with Polling

```
block_write(rca, nob, addr_h, addr_l, buswidth)
  {
    send_cmd_wait_resp(SEND_STATUS, rca, 0x00, 0x01, 0x40);
    while(!Ready for data in card status is true)
    {
      send_cmd_wait_resp(SEND_STATUS, rca, 0x00, 0x01, 0x40);
    }
    write_reg(NOB, <nob>);
    send_cmd_wait_resp(SET_BLOCKLEN, 0x00, 0x0200, 0x01, 0x40);
    if(buswidth==4-bit mode)
    {
      send_cmd_wait_resp(APP_CMD, rca, 0x0, 0x01, 0x40);
      send_cmd_wait_resp(SET_BUS_WIDTH, 0x00, 0x02, 0x01, 0x40);
    }
    if(nob==1)
    { if(buswidth=4-bit mode)
      send_cmd_wait_resp(WRITE_SINGLE_BLOCK, addr_h, addr_l, 0x219, 0x40);
    else
    send_cmd_wait_resp(WRITE_SINGLE_BLOCK, addr_h, addr_l, 0x19, 0x40);
    }
    else
    if(buswidth==4-bit mode)
      send_cmd_wait_resp(WRITE_MULTIPLE_BLOCK, addr_h, addr_l, 0x219, 0x40);
      send_cmd_wait_resp(WRITE_MULTIPLE_BLOCK, addr_h, addr_l, 0x19, 0x40);
    }
    while(!FIFO empty in STATUS is true);
    {
      for(i=0;i<(nob*8);i++)
      {
        while(!FIFO full in STATUS);// polling instead of irq or dma req
        for(j=0;j<32;j++)
        {
          BUFFER_ACCESS = SDRAM_ADDR[i*32+j];
        }
    }
    else// 1-bit mode
    {
      for(i=0;i<(nob*32);i++)
      {
        while(!FIFO full in STATUS);// polling instead of irq or dma req
        for(j=0;j<8;j++)
        {
          BUFFER_ACCESS = SDRAM_ADDR[i*8+j];
        }
      }
    }
    while(!Access Operation Done in STATUS true);
    while(!card bus is stop);
    if(nob > 1)
    {
      send_cmd_wait_resp(STOP_TRANS, 0x00, 0x00, 0x1, 0x40);
    }
  }
```

18.7.3.1.2 Block Read

In Block Mode transfer, CRC is used and appended to the end of each block ensuring data transfer integrity. READ_SINGLE_BLOCK (CMD17) initiates a block read and after completing the transfer, the card returns to the transfer state. READ_MULTIPLE_BLOCK (CMD18) starts a transfer of several consecutive blocks. Blocks are continuously transferred until a stop command is issued. When the MMC/SD module uses partial blocks with an

accumulated length that is not block aligned and block misalignment is not allowed, the card detects a block misalignment at the beginning of the first mis-aligned block, set the ADDRESS_ERROR error bit in the Card Status Register, abort transmission, and wait in the data state for a stop command. Code Example 18-9 provides the program code for the block read with DMA.

Code Example 18-9. Block_Read with DMA

```
block_read(rca, nob, addr_h, addr_l, buswidth)
            {
                  send_cmd_wait_resp(SEND_STATUS, rca, 0x00, 0x01, 0x40);
                  while(!Ready for data in card status is true)
                  {
                        send_cmd_wait_resp(SEND_STATUS, rca, 0x00, 0x01, 0x40);
                  }
                  write_reg(NOB, <nob>);
                  send_cmd_wait_resp(SET_BLOCKLEN, 0x00, 0x0200, 0x01, 0x40);
                  if(buswidth==4-bit mode)
                  {
                        send_cmd_wait_resp(APP_CMD, rca, 0x0, 0x01, 0x40);
                        send_cmd_wait_resp(SET_BUS_WIDTH, 0x00, 0x02, 0x01, 0x40);
                  }

                        // Configure the DMA for FIFO read operation (BUFFER_ACCESS,
SDRAM_ADDR, nob);
                        // Set DMA source address = BUFFER_ACCESS
                        // Set DMA target address = SDRAM_ADDR
                        // Set DMA total byte transfer = nob
                        // Set DMA burst depth = 8 if 1-bit mode, = 32 if 4-bit mode

                  if(nob==1)
                  { if(buswidth=4-bit mode)
                        send_cmd_wait_resp(READ_SINGLE_BLOCK, addr_h, addr_l, 0x209,
0x40);
                  else
                        send_cmd_wait_resp(READ_SINGLE_BLOCK, addr_h, addr_l, 0x09,
0x40);
                  }
                  else
                  { if(buswidth=4-bit mode)
                        send_cmd_wait_resp(READ_MULTIPLE_BLOCK, addr_h, addr_l, 0x209,
0x40);
                  else
                        send_cmd_wait_resp(READ_MULTIPLE_BLOCK, addr_h, addr_l, 0x09,
0x40);
                  }
                  Enable DMA operation;
                  while(!Data Transfer Done in STATUS true);
                  while(!card bus is stop);
                  if(nob > 1)
                  {
                        send_cmd_wait_resp(STOP_TRANS, 0x00, 0x00, 0x1, 0x40);
                  }
            }
```

Code Example 18-10 on page 18-43 provides the program code for the block read with polling.

Code Example 18-10. Block_Read with Polling

```
block_read(rca, nob, addr_h, addr_l, buswidth)
  {
    send_cmd_wait_resp(SEND_STATUS, rca, 0x00, 0x01, 0x40);
    while(!Ready for data in card status is true)
    {
      send_cmd_wait_resp(SEND_STATUS, rca, 0x00, 0x01, 0x40);
    }
    write_reg(NOB, <nob>);
    send_cmd_wait_resp(SET_BLOCKLEN, 0x00, 0x0200, 0x01, 0x40);
    if(buswidth==4-bit mode)
    {
      send_cmd_wait_resp(APP_CMD, rca, 0x0, 0x01, 0x40);
      send_cmd_wait_resp(SET_BUS_WIDTH, 0x00, 0x02, 0x01, 0x40);
    }
    if(nob==1)
    { if(buswidth=4-bit mode)
      send_cmd_wait_resp(READ_SINGLE_BLOCK, addr_h, addr_l, 0x209, 0x40);
    else
    send_cmd_wait_resp(READ_SINGLE_BLOCK, addr_h, addr_l, 0x09, 0x40);
    }
    else
    { if(buswidth=4-bit mode)
      send_cmd_wait_resp(READ_MULTIPLE_BLOCK, addr_h, addr_l, 0x209, 0x40);
    else
      send_cmd_wait_resp(READ_MULTIPLE_BLOCK, addr_h, addr_l, 0x09, 0x40);
}
    if(buswidth==4-bit mode)
    {
      for(i=0;i<(nob*8);i++)
      {
        while(!FIFO full in STATUS);// polling instead of irq or dma req
        for(j=0;j<32;j++)
        {
          SDRAM_ADDR[i*32+j] = BUFFER_ACCESS;
        }
      }
    }
    else// 1-bit mode
    {
      for(i=0;i<(nob*32);i++)
      {
        while(!FIFO full in STATUS);// polling instead of irq or dma req
        for(j=0;j<8;j++)
        {
          SDRAM_ADDR[i*8+j] = BUFFER_ACCESS;
        }
      }
    }
    while(!Data Transfer Done in STATUS true);
    while(!card bus is stop);
    if(nob > 1)
    {
      send_cmd_wait_resp(STOP_TRANS, 0x00, 0x00, 0x1, 0x40);
    }
  }
```

18.7.3.2 Stream Access—Stream Write and Stream Read (MMC Only)

The module also supports Stream Mode access to the card. In Stream Mode data is transferred in bytes and no CRC appended at the end of each package/block. So it is relatively more effective. However with CRC, the user may use either the checking mechanism to verify the content, or the bit content self-independent, like raw sound track.

18.7.3.2.1 Stream Write

WRITE_DAT_UNTIL_STOP (CMD20) starts the data transfer from the MMC/SD module to the card, beginning at the specified address and continuing until the MMC/SD module issues a stop command. When partial blocks are allowed (CSD parameter WRITE_BL_PARTIAL is set), the data stream can start and stop at any address within the card address space, otherwise it can only start and stop at block boundaries. Because the amount of data to be transferred is not determined in advance, a CRC cannot be used. When the end of the memory range is reached while sending data and no stop command is sent by the MMC/SD module, any additional data transferred is discarded.

The maximum clock frequency for a stream write operation is given by Equation 18-3 using fields of the Card-Specific Data Register.

$$\text{MAXIMUM SPEED} = \text{MIN}\left(\text{TRAN_SPEED}, \frac{8 \bullet 2^{\text{WRITE_BL_LEN}} - \text{NSAC}}{\text{TAAC} \bullet \text{R2W_FACTOR}}\right) \qquad \textbf{\textit{Eqn. 18-3}}$$

- TRAN_SPEED = Maximum Data Transfer Rate
- READ_BL_LEN = Maximum Read Data Block Length
- NSAC = Data Read Access Time 2 in CLK Cycles
- TAAC = Data Read Access Time 1
- R2W_FACTOR = Write Speed Factor

When the MMC/SD module attempts to use a frequency higher than the value defined in Equation 18-3, the card may be unable to process the data, causing an overrun condition. When an overrun condition is detected, the card stops programming, sets the OVERRUN error bit in the Card Status Register, ignores all further data transfer attempts, and waits in the receive state for a stop command. When the MMC/SD module tries to write over a write-protected area, the write operation is aborted and the card sets the WP_VIOLATION bit in the Card Status Register.

Code Example 18-11 on page 18-45 provides the program code for the stream write.

Code Example 18-11. Stream_Write

```
stream_write(rca, nob, addr_h, addr_l, buswidth)
  {
    send_cmd_wait_resp(SEND_STATUS, rca, 0x00, 0x01, 0x40);
    while(!Ready for data in card status is true)
    {
      send_cmd_wait_resp(SEND_STATUS, rca, 0x00, 0x01, 0x40);
    }
    write_reg(NOB, 0xffff);
    send_cmd_wait_resp(SET_BLOCKLEN, 0x00, 0x0200, 0x01, 0x40);
    if(buswidth==4-bit mode)
    {
      send_cmd_wait_resp(APP_CMD, rca, 0x0, 0x01, 0x40);
      send_cmd_wait_resp(SET_BUS_WIDTH, 0x00, 0x02, 0x01, 0x40);
    }
    send_cmd_wait_resp(WRITE_DAT_UNTIL_STOP, addr_h, addr_l, 0x79, 0x40);
    while(!FIFO empty in STATUS is true);
    if(buswidth==4-bit mode)
    {
      for(i=0;i<(nob*8);i++)
      {
        while(!FIFO full in STATUS);// polling instead of irq or dma req
        for(j=0;j<32;j++)
        {
          SDRAM_ADDR[i*32+j] = BUFFER_ACCESS;
        }
      }
    }
    else// 1-bit mode
    {
      for(i=0;i<(nob*32);i++)
      {
        while(!FIFO full in STATUS);// polling instead of irq or dma req
        for(j=0;j<8;j++)
        {
          SDRAM_ADDR[i*8+j] = BUFFER_ACCESS;
        }
      }
    }
    send_cmd_wait_resp(STOP_TRANS, 0x00, 0x00, 0x1, 0x40);
  }
```

18.7.3.2.2 Stream Read

READ_DAT_UNTIL_STOP (CMD11) controls a stream-oriented data transfer. This command instructs the card to send its data, starting at a specified address, until the MMC/SD module sends STOP_TRANSMISSION (CMD12). The stop command has an execution delay due to the serial command transmission and the data transfer stops after the end bit of the stop command. When the end of the memory range is reached while sending data and no stop command is sent by the MMC/SD module, any subsequent data sent is considered undefined.

The maximum clock frequency for a stream read operation is given by Equation 18-4 and uses fields of the Card-Specific Data Register.

$$\text{MAXIMUM SPEED} = \text{MIN}\left(\text{TRAN_SPEED}, \frac{8 \bullet 2^{\text{READ_BL_LEN}} - \text{NSAC}}{\text{TAAC}}\right) \qquad \textit{Eqn. 18-4}$$

- TRAN_SPEED = Maximum Data Transfer Rate
- READ_BL_LEN = Maximum Read Data Block Length
- NSAC = Data Read Access Time 2 in CLK Cycles

- TAAC = Data Read Access Time 1

When the MMC/SD module attempts to use a frequency higher than the value defined in Equation 18-4, the card may be unable to sustain data transfer, causing an underrun condition. When an underrun condition is detected, the card sets the UNDERRUN error bit in the Card Status Register, aborts the transmission, and waits in the data state for a stop command. Code Example 18-12 provides the program code for the stream read.

Code Example 18-12. Stream_Read

```
stream_read(rca, nob, addr_h, addr_l, buswidth)
      {
            send_cmd_wait_resp(SEND_STATUS, rca, 0x00, 0x01, 0x40);
            while(!Ready for data in card status is true)
            {
                  send_cmd_wait_resp(SEND_STATUS, rca, 0x00, 0x01, 0x40);
            }
            write_reg(NOB, 0xffff);
            send_cmd_wait_resp(SET_BLOCKLEN, 0x00, 0x0200, 0x01, 0x40);
            if(buswidth==4-bit mode)
            {
                  send_cmd_wait_resp(APP_CMD, rca, 0x0, 0x01, 0x40);
                  send_cmd_wait_resp(SET_BUS_WIDTH, 0x00, 0x02, 0x01, 0x40);
            }
            send_cmd_wait_resp(READ_DAT_UNTIL_STOP, addr_h, addr_l, 0x29, 0x40);
            if(buswidth==4-bit mode)
            {
                  for(i=0;i<(nob*8);i++)
                  {
                        while(!FIFO full in STATUS);// polling instead of irq or dma req
                        for(j=0;j<32;j++)
                        {
                              SDRAM_ADDR[i*32+j] = BUFFER_ACCESS;
                        }
                  }
            }
            else// 1-bit mode
            {
                  for(i=0;i<(nob*32);i++)
                  {
                        while(!FIFO full in STATUS);// polling instead of irq or dma req
                        for(j=0;j<8;j++)
                        {
                              SDRAM_ADDR[i*8+j] = BUFFER_ACCESS;
                        }
                  }
            }
            send_cmd_wait_resp(STOP_TRANS, 0x00, 0x00, 0x1, 0x40);
      }
```

18.7.3.3 Erase—Group Erase (MMC Only) and Sector Erase

It is often desirable to erase multiple sectors simultaneously to enhance the data throughput. Identification of these sectors is accomplished with the Tag commands.

MultimediaCard and Secure Digital support the Sector Erase commands (CMD 32–33), while the Group Erase commands (CMD 35–36) are only supported by MMC.

The unit of measure for an erase is either a sector or an erase group. The size of an erase group is card-specific and set in the CSD Register. To define particular sectors to erase, all selected sectors must lie within the same erase group. Alternately, the user can erase multiple erase groups at a time.

To erase efficiently, select a group of sectors or erase groups by using either TAG_SECTOR_START (CMD32) or TAG_ERASE_GROUP_START (CMD35) which identifies the starting address. Follow this command with either TAG_SECTOR_END (CMD 33) or TAG_ERASE_GROUP_END (CMD36). All sectors or erase groups within the range are selected. The user can deselect a sector or erase group using the UNTAG_SECTOR (CMD34) or UNTAG_ERASE_GROUP (CMD37) commands.

To erase by sectors, perform the following command sequence:

1. TAG_SECTOR_START (CMD32)
2. TAG_SECTOR_END (CMD33)
3. UNTAG_SECTOR (CMD34), to optionally de-select sectors within the address range
4. ERASE (CMD38)

To erase by groups, perform the following command sequence:

1. TAG_ERASE_GROUP_START (CMD35)
2. TAG_ERASE_GROUP_END (CMD36)
3. UNTAG_ERASE_GROUP (CMD37), to optionally de-select erase groups within the address range
4. ERASE (CMD38)

Up to 16 UNTAG* commands can be sent within one erase cycle. When an ERASE, TAG*, or UNTAG* command is received out-of-sequence, the card sets the ERASE_SEQ_ERROR bit in the Card Status Register and resets the whole sequence. When an out-of-sequence command (except SEND_STATUS) is received, the card sets the ERASE_RESET status bit in the Card Status Register, resets the erase sequence, and executes the last command. When the erase range includes write protected sectors, they are left intact, only the non-protected sectors are erased, and the WP_ERASE_SKIP status bit in the Card Status Register is set. The address field in the TAG* commands is a sector or a group address in byte units. The card ignores all least significant bits (LSBs) below the group or sector size, respectively.

As with block write, the card indicates that an erase is in progress by holding the SD_DAT line(s) low. The actual erase time can be quite long, and the MMC/SD module can issue SELECT/DESELECT_CARD (CMD7) to deselect the card.

18.7.3.4 Wide Bus Selection or Deselection

Wide bus (4-bit bus width) operation mode is selected or deselected using SET_BUS_WIDTH (ACMD6). The default bus width after power-up or GO_IDLE_STATE (CMD0) is 1-bit. SET_BUS_WIDTH (ACMD6) is only valid in a transfer state which means the bus width can be changed only after a card is selected by SELECT/DESELECT_CARD (CMD7).

18.7.4 Protection Management

Three write-protection methods for the cards are supported in the SDHC module:

- Card internal write protection (card responsibility)
- Mechanical write protection switch (MMC/SD module responsibility only)
- Password protection card lock operation

18.7.4.1 Card Internal Write Protection

Card data can be protected against write and erase. By setting the permanent or temporary write-protect bits in the CSD, the entire card can be permanently write-protected by the manufacturer or content provider. For cards that support write-protection of groups of sectors by setting the WP_GRP_ENABLE bit in the CSD, portions of the data can be protected, and the write-protection can be changed by the application. The write-protection is in units of WP_GRP_SIZE sectors as specified in the CSD. The commands SET_WRITE_PROT and CLR_WRITE_PROT control the protection of the addressed group.

The SEND_WRITE_PROT command is similar to a single block read command. The card sends a data block containing 32 write protection bits (representing 32 write protect groups starting at the specified address) followed by 16 CRC bits. The address field in the write protect commands is a group address in byte units. The card ignores all LSBs below the group size.

18.7.4.2 Mechanical Write Protect Switch

A mechanical sliding tab on the side of the card allows the user to set or clear write protection on a card. When the sliding tab is positioned with the window open, the card is write protected, and when the window is closed, the card contents can be changed.

A proper, matched switch on the socket side indicates to the MMC/SD module that the card is write protected. The MMC/SD module is responsible for protecting the card. The position of the write protect switch is unknown to the internal circuitry of the card.

18.7.4.3 Password Protect

The password protection feature enables the MMC/SD module to lock and unlock a card with a password. The password is stored in the 128-bit PWD Register and its size is set in the 8-bit PWD_LEN Register. These registers are non-volatile so that a power cycle does not erase them.

Locked cards respond to and execute certain commands. This means that the MMC/SD module is allowed to reset, initialize, select, and query for status, however it is not allowed to access data on the card. When the password is set (as indicated by a nonzero value of PWD_LEN), the card is locked automatically after power on. As with the CSD and CID Register write commands, the lock/unlock commands are available in transfer state only. In this state, the command does not include an address argument and the card must be selected before using it. The card lock/unlock commands have the structure and bus transaction types of a regular single block write command. The transferred data block includes all of the required information for the command (the password setting mode, the PWD itself, and card lock/unlock). The command data block size is defined by the MMC/SD module before it sends the card lock/unlock command, and has the structure shown in Table 18-21.

Table 18-21. Structure of Command Data Block

Byte#	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Reserved	Reserved	Reserved	Reserved	ERASE	LOCK_ UNLOCK	CLR_ PWD	SET_ PWD
1	PWD_LEN							
2	PWD							
...								
PWD_LEN +1								

The bit settings are as follows:

- **ERASE**—Setting forces an erase operation. All other bits must be zero, and only the command byte is sent.
- **LOCK_UNLOCK**—Setting locks the card. LOCK_UNLOCK can be set simultaneously with SET_PWD, however not with CLR_PWD.
- **CLR_PWD**—Setting clears the password data.
- **SET_PWD**—Setting saves the password data to memory.
- **PWD_LEN**—Defines the length of the password in bytes.
- **PWD**—The password (new or currently used, depending on the command).

The following sections list the command sequences to set/reset a password, lock/unlock the card, and force an erase.

18.7.4.3.1 Setting the Password

1. Select a card (SELECT/DESELECT_CARD, CMD7), if not already selected.
2. Define the block length (SET_BLOCKLEN, CMD16) to send, given by the 8-bit card lock/unlock mode (Byte 0 in Table 18-21), the 8-bit PWD_LEN, and the number of bytes of the new password. When a password replacement is done, the block size must take into account that both the old and the new passwords are sent with the command.
3. Send LOCK/UNLOCK (CMD42) with the appropriate data block size on the data line including the 16-bit CRC. The data block indicates the mode (SET_PWD = 1), the length (PWD_LEN), and the password (PWD) itself. When a password replacement is done, the length value (PWD_LEN) includes the length of both passwords, the old and the new one, and the PWD field includes the old password (currently used) followed by the new password.
4. When the password is matched, the new password and its size are saved into the PWD and PWD_LEN fields, respectively. When the old password sent is not correct in size and/or content with the expected password, the LOCK_UNLOCK_FAILED error bit is set in the Card Status Register and the password is not changed.

NOTE:

The password length field (PWD_LEN) indicates whether a password is currently set. When this field is nonzero, there is a password set and the card locks itself after power-up. It is possible to lock the card immediately in the current power session by setting the LOCK_UNLOCK bit (while setting the password) or sending an additional command for card locking.

18.7.4.3.2 Resetting the Password

1. Select a card (SELECT/DESELECT_CARD, CMD7), if not already selected.
2. Define the block length (SET_BLOCKLEN, CMD16) to send, given by the 8-bit card lock/unlock mode (Byte 0 in Table 18-21), the 8-bit PWD_LEN, and the number of bytes of the current password.
3. Send LOCK/UNLOCK (CMD42) with the appropriate data block size on the data line including the 16-bit CRC. The data block indicates the mode (CLR_PWD = 1), the length (PWD_LEN), and the password (PWD) itself. The LOCK_UNLOCK bit is ignored.

4. When the password is matched, the PWD field is cleared and PWD_LEN is set to 0. When the password sent is not correct in size and/or content with the expected password, the LOCK_UNLOCK_FAILED error bit is set in the Card Status Register and the password is not changed.

18.7.4.3.3 Locking a Card

1. Select a card (SELECT/DESELECT_CARD, CMD7), if not already selected.
2. Define the block length (SET_BLOCKLEN, CMD16) to send, given by the 8-bit card lock/unlock mode (Byte 0 in Table 18-21), the 8-bit PWD_LEN, and the number of bytes of the current password.
3. Send LOCK/UNLOCK (CMD42) with the appropriate data block size on the data line including the 16-bit CRC. The data block indicates the mode (LOCK_UNLOCK = 1), the length (PWD_LEN), and the password (PWD) itself.
4. When the password is matched, the card is locked and the CARD_IS_LOCKED status bit is set in the Card Status Register. When the password sent is not correct in size and/or content with the expected password, the LOCK_UNLOCK_FAILED error bit is set in the Card Status Register and the lock fails.

NOTE:

It is possible to set the password and to lock the card in the same sequence. In this case, the MMC/SD module performs all the required steps for setting the password (see Section 18.7.4.3.1, “Setting the Password,” on page 18-49), however it is necessary to set the LOCK_UNLOCK bit in Step 3 when the new password command is sent.

When the password is previously set (PWD_LEN is not 0), the card is locked automatically after power on reset. An attempt to lock a locked card or to lock a card that does not have a password fails and the LOCK_UNLOCK_FAILED error bit is set in the Card Status Register.

18.7.4.3.4 Unlocking the Card

1. Select a card (SELECT/DESELECT_CARD, CMD7), if not already selected.
2. Define the block length (SET_BLOCKLEN, CMD16) to send, given by the 8-bit card lock/unlock mode (Byte 0 in Table 18-21), the 8-bit PWD_LEN, and the number of bytes of the current password.
3. Send LOCK/UNLOCK (CMD42) with the appropriate data block size on the data line including the 16-bit CRC. The data block indicates the mode (LOCK_UNLOCK = 0), the length (PWD_LEN), and the password (PWD) itself.
4. When the password is matched, the card is unlocked and the CARD_IS_LOCKED status bit is cleared in the Card Status Register. When the password sent is not correct in size and/or content with the expected password, the LOCK_UNLOCK_FAILED error bit is set in the Card Status Register and the card remains locked.

NOTE:

The unlocking function is only valid for the current power session. When the PWD field is not clear, the card is locked automatically on the next power up.

An attempt to unlock an unlocked card fails and the LOCK_UNLOCK_FAILED error bit is set in the Card Status Register.

18.7.4.3.5 Forcing Erase

When the user has forgotten the password (PWD content), it is possible to access the card after clearing all the data on the card. This forced erase operation erases all card data and all password data.

1. Select a card (SELECT/DESELECT_CARD, CMD7), if not already selected.
2. Set the block length (SET_BLOCKLEN, CMD16) to 1 byte. Only the 8-bit card lock/unlock byte (Byte 0 in Table 18-21) is sent.
3. Send LOCK/UNLOCK (CMD42) with the appropriate data byte on the data line including the 16-bit CRC. The data block indicates the mode (ERASE = 1). All other bits must be zero.
4. When the ERASE bit is the only bit set in the data field, all card content is erased, including the PWD and PWD_LEN fields, and the card is no longer locked. When any other bits are set, the LOCK_UNLOCK_FAILED error bit is set in the Card Status Register and the card retains all of its data, and remains locked.

An attempt to use a force erase on an unlocked card fails and the LOCK_UNLOCK_FAILED error bit is set in the Card Status Register.

18.7.5 Card Status Register

The response format R1 contains a 32-bit card status field. This field transmits the card's status information (which is stored in a local status register) to the MMC/SD module. When not specified, the status entries are always related to the last command issued.

Table 18-22 defines the different entries of the Card Status Register. The type and clear condition fields in the table are abbreviated as follows:

Type:

- **E**—Error bit.
- **S**—Status bit.
- **R**—Detected and set for the actual command response.
- **X**—Detected and set during command execution. The MMC/SD module must poll the card by issuing the status command to read these bits.

Clear Condition:

- **A**—According to the card current state.
- **B**—Always related to the previous command. Reception of a valid command clears it (with a delay of one command).
- **C**—Clear by read.

Table 18-22. Card Status Register Description

Bit	Bit Name	Type	Settings	Description	Clear Condition
31	OUT_OF _RANGE	E R	0 = No error 1 = Error	The command's argument is out of the allowed range for this card.	C
30	ADDRESS _ERROR	E R X	0 = No error 1 = Error	A misaligned address not matching the block length is used in the command.	C

Table 18-22. Card Status Register Description (continued)

Bit	Bit Name	Type	Settings	Description	Clear Condition
29	BLOCK_LEN _ERROR	E R	0 = No error 1 = Error	The transferred block length is not allowed, or the number of transferred bytes does not match the block length.	C
28	ERASE_SEQ _ERROR	E R	0 = No error 1 = Error	The sequence of erase commands is invalid.	C
27	ERASE _PARAM	E X	0 = No error 1 = Error	The selected sectors or groups for erase are invalid.	C
26	WP _VIOLATION	E R X	0 = No error 1 = Protected Block	An attempt is made to program a write protected block.	C
25	CARD_IS _LOCKED	S X	0 = Card Unlocked 1 = Card Locked	The card is locked by the MMC/SD module.	A
24	LOCK _UNLOCK _FAILED	E R X	0 = No error 1 = Error	Any of the following has occurred: • Password incorrect in length or content • An attempt to lock a locked card • An attempt to unlock an unlocked card • A forced erase attempted on an unlocked card	C
23	COM_CRC _ERROR	E R	0 = No error 1 = Error	The CRC check of the previous command failed.	B
22	ILLEGAL _COMMAND	E R	0 = No error 1 = Error	The executed command is not legal for the card state.	B
21	CARD_ECC _FAILED	E X	0 = No failure 1 = Failure	Card internal ECC is applied but failed to correct the data.	C
20	CC_ERROR	E R X	0 = No error 1 = Error	Internal card controller error.	C
19	ERROR	E R X	0 = No error 1 = Error	A general or an unknown error occurred during the operation.	C
18	UNDERRUN	E X	0 = No error 1 = Error	The card could not sustain data transfer in stream read mode.	C
17	OVERRUN	E X	0 = No error 1 = Error	The card could not sustain data programming in stream write mode.	C
16	CID/CSD_ OVERWRITE	E R X	0 = No error 1 = Error	Any of the following has occurred: • The CID register is already written and cannot be overwritten • The read-only section of the CSD does not match the card content • An attempt is made to reverse the copy (set as original) or permanent WP (unprotected) bits	C

Table 18-22. Card Status Register Description (continued)

Bit	Bit Name	Type	Settings	Description	Clear Condition
15	WP_ERASE _SKIP	S X	0 = All selected areas erased 1 = Some selected areas are protected	Parts of the selected erase area are protected. Only partial address space is erased due to existing write protected blocks.	C
14	CARD_ECC _DISABLED	S X	0 = Internal ECC used 1 = Internal ECC disabled	The command is executed without using the internal ECC.	A
13	ERASE _RESET	S R	0 = Erase completed 1 = Erase sequence not completed	An out-of-sequence erase command is received, so the erase sequence is cleared before executing.	C
12:9	CURRENT _STATE	S X	0000 (0) = Idle 0001 (1) = Ready 0010 (2) = Identification 0011 (3) = Standby 0100 (4) = Transfer 0101 (5) = Data 0110 (6) = Receive 0111 (7) = Programming 1000 (8) = Disconnect 1001 (9)–1111 (15) are Reserved	The state of the card when receiving the command. When the command execution causes a state change, it is visible to the MMC/SD module in the response to the next command.	B
8	READY_FOR _DATA	S X	0 = Not ready 1 = Ready	Corresponds to buffer empty signalling on the bus	A
7:6	Reserved				
5	APP_CMD	S R	0 = Disabled 1 = Enabled	The card expects an ACMD, or indication that the command is interpreted as an ACMD.	C
4:0	Reserved				

18.7.6 SD Status Register

The SD Status Register contains status bits that are related to the SD card proprietary features and are available for future application-specific usage. The size of the SD Status Register is one 512-bit data block. The content of this register (and a 16-bit CRC) is transmitted to the MMC/SD module over the SD_DAT bus when the MMC/SD module sends the SD_STATUS (ACMD13) command. This command consists of APP_CMD (CMD55) followed by CMD13. SD_STATUS (ACMD13) can only be sent to a card in transfer state. SD Status Register structure is described in this section.

Table 18-22 defines the different entries of the SD Status Register. The type and clear condition fields in the table are abbreviated as follows:

Type:

- **E**—Error bit.
- **S**—Status bit.

- **R**—Detected and set for the actual command response.
- **X**—Detected and set during command execution. The MMC/SD module must poll the card by issuing the status command to read these bits.

Clear Condition:

- **A**—According to the card current state.
- **B**—Always related to the previous command. Reception of a valid command clears it (with a delay of one command).
- **C**—Clear by read

Table 18-23. SD Status Register

Bit	Bit Name	Type	Settings	Description	Clear Condition
511:510	DAT_BUS_WIDTH	S R	00 = 1-bit width 01 = reserved 10 = 4-bit width 11 = reserved	Shows the currently defined data bus width that is defined by SET_BUS_WIDTH (ACMD6, 1-bit width is default)	A
509	SECURED_MODE	S R	0 = Not in secured mode 1 = In secured mode	Card is in secured mode of operation	A
508:496	Reserved				
495:480	SD_CARD_TYPE	S R	0 = SD Memory Cards 1 = SD I/O Cards	Selects between SD Memory and SD I/O cards.	A
479:448	SIZE_OF_ PROTECTED_AREA	S R	See Description	Size of protected area (in units defined by SET_BLOCKLEN, CMD16)	A
447:312	Reserved				
311:0	Reserved for manufacturer				

18.7.7 SD I/O

I/O access differs from memory access because the I/O registers can be written and read individually and directly without a File Allocation Table (FAT) file structure or the concept of blocks (although block access is supported). These registers allow access to the I/O data, control of the I/O functions, report on status of data, and transfer I/O data to/from the MMC/SD module.

Each SD I/O card can have between 1 and 7 functions plus one memory function built into it. A function is a self-contained I/O device. These functions can be identical or completely different from each other. All I/O functions are organized as a collection of registers, and there are a maximum of 131,072 registers possible for each I/O function.

18.7.7.1 SD I/O Interrupts

To allow the SD I/O card to interrupt the MMC/SD module, and interrupt function is available on a pin on the SD interface. Pin 8, used as SD_DAT [1] when operating in the 4-bit SD mode, signals the card's interrupt to the MMC/SD module. The use of the interrupt is optional for each card or function within a card. The SD I/O interrupt

is level sensitive, which means that the interrupt line must be held active (low) until it is either recognized and acted upon by the MMC/SD module or deasserted due to the end of the interrupt period. After the MMC/SD module has serviced the interrupt, the interrupt status bit is cleared via an I/O write to the appropriate bit in the SD I/O card internal registers. The interrupt output of all SD I/O cards is active low and the MMC/SD module provides pull-up resistors on all data lines (SD_DAT [3:0]).

The MMC/SD module samples the level of Pin 8 (SD_DAT [1]/IRQ) into the interrupt detector only during the interrupt period. At all other times, the MMC/SD module ignores this value.

NOTE:

The interrupt period is applicable for both memory and I/O operations. The definition of the interrupt period for operations with single blocks is different from the definition for multiple block data transfers.

18.7.7.2 SD I/O Suspend and Resume

Within a multi-function SD I/O or a card with both I/O and memory functions, there are multiple devices (I/O and memory) that share access to the MMC/SD bus. To share access to the MMC/SD module among multiple devices, SD I/O and combo cards optionally implement the concept of suspend/resume. When a card supports suspend/resume, the MMC/SD module can temporarily halt a data transfer operation to one function or memory (suspend) to free the bus for a higher priority transfer to a different function or memory. After this higher-priority transfer is complete, the original transfer is resumed (re-started) where it left off. Support of suspend/resume is optional on a per-card basis.

To perform the suspend/resume operation on the MMC/SD bus, the MMC/SD module performs the following steps:

1. Determines the function currently using the SD_DAT [3:0] line(s).
2. Requests the lower priority or slower transaction to suspend.
3. Waits for the transaction suspension to complete.
4. Begins the higher priority transaction.
5. Waits for the completion of the higher priority transaction.
6. Restores the suspended transaction.

18.7.7.3 SD I/O ReadWait

The optional ReadWait (RW) operation is defined only for the SD 1-bit and 4-bit modes. The ReadWait operation allows the MMC/SD module to signal a card that it is reading multiple registers (IO_RW_EXTENDED, CMD53) to temporarily stall the data transfer while allowing the MMC/SD module to send commands to any function within the SD I/O device. To determine when a card supports the ReadWait protocol, the MMC/SD module must test capability bits in the card internal registers. The timing for ReadWait is based on the interrupt period.
Code Example 18-13 on page 18-56 provides the programming code for the SD I/O RW operation.

Code Example 18-13. Read_Wait

```
block_read_with_read_wait_without_DMA(rca, nob, addr_h, addr_l, buswidth)
  {
    send_cmd_wait_resp(SEND_STATUS, rca, 0x00, 0x01, 0x40);
    while(!Ready for data in card status is true)
    {
      send_cmd_wait_resp(SEND_STATUS, rca, 0x00, 0x01, 0x40);
    }
    write_reg(NOB, <nob>);
    send_cmd_wait_resp(SET_BLOCKLEN, 0x00, 0x0200, 0x01, 0x40);
    if(buswidth==4-bit mode)
    {
      send_cmd_wait_resp(APP_CMD, rca, 0x0, 0x01, 0x40);
      send_cmd_wait_resp(SET_BUS_WIDTH, 0x00, 0x02, 0x01, 0x40);
    }
    write_reg(CMD_DAT_CONT, set bit 10); // Enable Read Wait at the
              // following block boundary
    if(nob==1)
      send_cmd_wait_resp(READ_SINGLE_BLOCK, addr_h, addr_l, 0x09, 0x40);
    else
      send_cmd_wait_resp(READ_MULTIPLE_BLOCK, addr_h, addr_l, 0x09, 0x40);
    if(buswidth==4-bit mode)
    {
      for(i=0;i<(nob*8);i++)
      {
        while(!FIFO full in STATUS);// polling instead of irq or dma req
        for(j=0;j<32;j++)
        {
          SDRAM_ADDR[i*32+j] = BUFFER_ACCESS;
        }
        send_cmd_wait_resp(IO_RW_DIRECT, arg_h, arg_l, 0x5, 0x40);
        write_reg(CMD_DAT_CONT, set bit 11 to stop Read Wait);
      }
    }
    else// 1-bit mode
    {
      for(i=0;i<(nob*32);i++)
      {
        while(!FIFO full in STATUS);// polling instead of irq or dma req
        for(j=0;j<8;j++)
        {
          SDRAM_ADDR[i*8+j] = BUFFER_ACCESS;
        }
        send_cmd_wait_resp(IO_RW_DIRECT, arg_h, arg_l, 0x5, 0x40);
        write_reg(CMD_DAT_CONT, set bit 11 to stop Read Wait);
      }
    }
    while(!Data Transfer Done in STATUS true);
    while(!card bus is stop);
    if(nob > 1)
    {
      send_cmd_wait_resp(STOP_TRANS, 0x00, 0x00, 0x41, 0x40);
    }
  }
```

18.7.8 Commands and Responses

This section describes application-specific and general commands in addition to the command types and formats. Table 18-25 on page 18-58 is a list of all the MMC/SD module commands. Section 18.7.8.5, “Response Formats,” identifies and describes all command response formats.

18.7.8.1 Application-Specific and General Commands

The MMC/SD module system is designed to provide a standard interface for a variety of applications types. In this environment, there is a need for specific customers/applications features. To implement these features, two types of generic commands are defined in the standard: application-specific commands (ACMD) and general commands (GEN_CMD).

When the card receives the APP_CMD (CMD55) command, the card expects the next command to be an application-specific command. ACMDs have the same structure as regular MMC commands and can have the same CMD number. The card recognizes it as ACMD because it appears after APP_CMD (CMD55). When the command immediately following the APP_CMD (CMD55) is not a defined application-specific command, the standard command is used.

For example, when the card has a definition for SD_STATUS (ACMD13), and receives CMD13 immediately following APP_CMD (CMD55), this is interpreted as SD_STATUS (ACMD13). However, when the card receives CMD7 immediately following APP_CMD (CMD55) and the card does not have a definition for ACMD7, this is interpreted as the standard (SELECT/DESELECT_CARD)CMD7.

To use one of the manufacturer specific ACMDs the MMC/SD module must perform the following steps:

1. Send APP_CMD (CMD55).
   — The card responds to the MMC/SD module, indicating that the APP_CMD bit is set and an ACMD is now expected.
2. Send the required ACMD.
   — The card responds to the MMC/SD module, indicating that the APP_CMD bit is set and that the accepted command is interpreted as an ACMD. When a non-ACMD is sent it is handled by the card as a normal MMC command and the APP_CMD bit in the Card Status Register stays clear.

When an invalid command is sent (neither ACMD nor CMD) it is handled as a standard MMC illegal command error.

The bus transaction for a GEN_CMD is the same as the single block read or write commands (WRITE_BLOCK, CMD24 or READ_SINGLE_BLOCK,CMD17). In this case, the argument denotes the direction of the data transfer rather than the address and the data block has vendor-specific format and meaning.

The card must be selected (in transfer state) before sending GEN_CMD (CMD56). The data block size is defined by SET_BLOCKLEN (CMD16). The response to GEN_CMD (CMD56) is in R1b format.

18.7.8.2 Command Types

Both application-specific and general commands are divided into the following four types:

- **Broadcast command (BC)**—Sent to all cards; no responses returned.
- **Broadcast Command with Response (BCR)**—Sent to all cards; responses received from all cards simultaneously.
- **Addressed (Point-to-Point) Command (AC)**—Sent to the card that is selected; does not include a data transfer on the SD_DAT line(s).
- **Addressed (Point-to-Point) Data Transfer Command (ADTC)**—Sent to the card that is selected; includes data transfer on the SD_DAT line(s).

18.7.8.3 Command Formats

All commands are sent over the SD_CMD line, are a fixed length of 48 bits, and are in the format shown in Table 18-24.

Table 18-24. Command Format

	Bit 47	Bit 46	Bits 45–40	Bits 39–8	Bits 7–1	Bit 0
Description	Start Bit	Transmission Bit	Command Index	Argument	CRC7	End Bit
Value	0	1	x	x	x	1

- **Transmission Bit**—A 1 indicates a transmission from the MMC/SD module to a card.
- **Command Index**—Contains the binary-coded number of the command index.
- **Argument**—Some commands require an address that is coded in 32 bits.

18.7.8.4 Commands for the MMC/SD Module

Table 18-25 list all the MMC/SD module's commands, their type, argument, format, and description.

Table 18-25. Commands for MMC/SD Module

CMD INDEX	Type	Argument	Response Format	Abbreviation	Description
CMD0	BC	[31:0] stuff bits	–	GO_IDLE _STATE	Resets all MMC and SD memory cards to idle state.
CMD1	BCR	[31:0] OCR without busy	R3	SEND_OP _COND	Asks all MMC and SD memory cards in idle state to send their operation conditions register contents on the SD_CMD line.
CMD2	BCR	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the SD_CMD line.
CMD3	AC	[31:16] RCA [15:0] stuff bits	R1 R6 (SD I/O)	SET_RELATIVE _ADDR	Assigns the argument as the relative address of the card.
CMD4	BC	[31:16] DSR [15:0] stuff bits	–	SET_DSR	Programs the DSR of all cards.
CMD5	BC	[31:0] OCR without busy	R4	IO_SEND_OP _COND	Asks all SD I/O cards in idle state to send their operation conditions register contents on the SD_CMD line.
CMD6	Reserved				

Table 18-25. Commands for MMC/SD Module (continued)

CMD INDEX	Type	Argument	Response Format	Abbreviation	Description
CMD7	AC	[31:16] RCA [15:0] stuff bits	R1b	SELECT/ DESELECT _CARD	Toggles a card between stand-by and transfer states or between programming and disconnect states. The card is selected by its own relative address and is deselected by any other address. Address 0 deselects all cards.
CMD8	Reserved				
CMD9	AC	[31:16] RCA [15:0] stuff bits	R2	SEND_CSD	Asks addressed card to sends its card-specific data (CSD) on the SD_CMD line.
CMD10	AC	[31:16] RCA [15:0] stuff bits	R2	SEND_CID	Asks addressed card to send its card-identification (CID) on the SD_CMD line.
CMD11	ADTC	[31:0] data address	R1	READ_DAT _UNTIL_STOP	Reads data stream from the card, starting at the given address, until a STOP_TRANSMISSION command.
CMD12	AC	[31:0] stuff bits	R1b	STOP_ TRANSMISSION	Forces the card to stop transmission.
CMD13	AC	[31:16] RCA [15:0] stuff bits	R1	SEND_STATUS	Asks addressed card to send its status register.
CMD14	Reserved				
CMD15	AC	[31:16] RCA [15:0] stuff bits	–	GO_INACTIVE _STATE	Sets the card to inactive state to protect the card stack against communication breakdowns.
CMD16	AC	[31:0] block length	R1	SET_ BLOCKLEN	Sets the block length (in bytes) for all future block commands (read and write). Default block length is specified in the CSD.
CMD17	ADTC	[31:0] data address	R1	READ_SINGLE _BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.
CMD18	ADTC	[31:0] data address	R1	READ_ MULTIPLE _BLOCK	Transfers data blocks from the card to the MMC/SD module continuously until interrupted by a STOP_TRANSMISSION command.
CMD19	Reserved				
CMD20	ADTC	[31:0] data address	R1	WRITE_DAT_ UNTIL_STOP	Writes data stream from the MMC/SD module, starting at the given address, until a STOP_TRANSMISION command is issued.

Table 18-25. Commands for MMC/SD Module (continued)

CMD INDEX	Type	Argument	Response Format	Abbreviation	Description
CMD21:23	Reserved				
CMD24	ADTC	[31:0] data address	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.
CMD25	ADTC	[31:0] data address	R1	WRITE_ MULTIPLE_ BLOCK	Writes blocks of data continuously until a STOP_TRANSMISSION command is issued.
CMD26	ADTC	[31:0] stuff bits	R1	PROGRAM_CID	Programs the card identification register. CMD26 is issued only once per card. The card contains hardware to prevent this operation after the first programming. Normally CMD26 is reserved for the manufacturer.
CMD27	ADTC	[31:0] stuff bits	R1	PROGRAM _CSD	Programs the programmable bits of the CSD.
CMD28	AC	[31:0] data address	R1b	SET_WRITE _PROT	Sets the write protection bit of the addressed group (when the card provides write protection features). The properties of write protection are coded in the CSD Register (WP_GRP_SIZE).
CMD29	AC	[31:0] data address	R1b	CLR_WRITE _PROT	Clears the write protection bit of the addressed group (when the card provides write protection features).
CMD30	ADTC	[31:0] write protect data address	R1	SEND_WRITE _PROT	Asks the card to send the status of the write protection bits (when the card provides write protection features).
CMD31	Reserved				
CMD32	AC	[31:0] data address	R1	TAG_SECTOR _START	Selects the start address of the range to be erased for sector erasing.
CMD33	AC	[31:0] data address	R1	TAG_SECTOR _END	Sets the address of the last sector in a continuous range within the selected erase group, or the address of a single sector to be selected for erase.
CMD34	AC	[31:0] data address	R1	UNTAG_ SECTOR	Removes one previously selected sector from the erase selection.
CMD35	AC	[31:0] data address	R1	TAG_ERASE _GROUP _START	Selects the start address of the range to be erased for group erasing.
CMD36	AC	[31:0] data address	R1	TAG_ERASE _GROUP_END	Sets the address of the last erase group within a continuous range to be selected for erase.

Table 18-25. Commands for MMC/SD Module (continued)

CMD INDEX	Type	Argument	Response Format	Abbreviation	Description
CMD37	AC	[31:0] data address	R1	UNTAG_ERASE _GROUP	Removes one previously selected erase group from the erase selection.
CMD38	AC	[31:0] stuff bits	R1b	ERASE	Erases all selected sectors.
CMD39	AC	[31:16] RCA [15] register write flag [14:8] register address [7:0] register data	R4	FAST_IO	Writes and reads 8-bit (register) data fields. Addresses a card and a register and provides the data for writing when the write flag is set. The R4 response contains data read from the address register. Accesses application dependent registers that are not defined in MMC and SD standards.
CMD40	BCR	[31:0] stuff bits	R5	GO_IRQ_STATE	Sets the system into interrupt state.
CMD41	reserved				
CMD42	ADTC	[31:0] stuff bits	R1b	LOCK_UNLOCK	Sets/resets the password or lock/unlock the card. The size of the data block is set by the SET_BLOCKLEN command.
CMD43:51	reserved				
CMD52	–	[31:0] stuff bits	R5	IO_RW_DIRECT	Access a single register within the total 128k of register space in any I/O function.
CMD53	–	[31:0] stuff bits	R5	IO_RW_ EXTENDED	Access multiple I/O registers with a single command. It allows the reading or writing of a large number of I/O registers.
CMD54	reserved				
CMD55	AC	[31:16] RCA [15:0] stuff bits	R1	APP_CMD	Indicates to the card that the next command is an application specific command rather that a standard command.
CMD56	ADTC	[31:1] stuff bits [0]: RD/WR[1]	R1b	GEN_CMD	Transfers a data block to the card or gets a data block from the card for general purpose or application specific commands. The size of the data block is set by the SET_BLOCKLEN command.
CMD57:63	reserved				
ACMDs are preceded by the APP_CMD (CMD55) command. Commands listed below are used for SD only. SD commands not listed below are not supported in this module.					

Table 18-25. Commands for MMC/SD Module (continued)

CMD INDEX	Type	Argument	Response Format	Abbreviation	Description
ACMD6	AC	[31:2] stuff bits [1:0] bus width	R1	SET_BUS _WIDTH	Defines the data bus width to be used for data transfer. 00 = 1-bit and 10 = 4-bit bus (The allowed data bus widths are given in the SCR Register.)
ACMD13	ADTC	[31:0] stuff bits	R1	SD_STATUS	Sends the SD memory card status.
ACMD22	ADTC	[31:0] stuff bits	R1	SEND_NUM_ WR_SECTORS	Sends the number of written (without errors) sectors. Responds with 32 bit + CRC data block.
ACMD23	AC	[31:23] stuff bits [22:0] number of blocks	R1	SET_WR_BLK _ERASE _COUNT	Sets the number of write blocks to be pre-erased before writing (used for faster execution of the WRITE_MULTIPLE_BLOCK (CMD 25). 1 = default (one write block).
ACMD41	BCR	[31:0] OCR	R3	SD_APP_OP _COND	Asks the accessed card to send its operating condition register (OCR) content on the SD_CMD line.
ACMD42	AC	[31:1] stuff bits [0] SET_CD	R1	SET_CLR_ CARD_DETECT	Connects (1) or disconnects (0) the 50 kOhm pull-up resistor on the CD/SD_DAT [3] (pin 1) of the card. The pull-up is used for card detection.
ACMD51	ADTC	[31:0] stuff bits	R1	SEND_SCR	Reads the SD Configuration Register.

1. **1**—The MMC/SD module receives a block of data from the card
   0—The MMC/SD module sends a block of data to the card

18.7.8.5 Response Formats

Table 18-25 on page 18-58 lists all commands supported by the MMC/SD module and their response formats. All responses are sent via the command line SD_CMD. The response transmission always starts with the left bit of the bit stream corresponding to the response codeword. The code length depends on the response type.

The first two bits of the response are the start bit and the direction bit. The second bit indicates the direction of transmission. Other response bits are dependent on the type of response.

Table 18-26 through Table 18-32 describe the command response formats. In the tables a value denoted by x indicates a variable entry. All responses except for type R3 are protected by a CRC. Every command codeword is terminated by the end bit.

18.7.8.5.1 R1—Normal Response

- Response length is 48 bits
- Bits 45:40 contain the binary-coded number of the command index generating the response
- The status of the card is coded in 32 bits (Card Status Register)

NOTE:

When data transfer to the card is involved, a busy signal can appear on the data line after the transmission of each block of data. The MMC/SD module must check for busy after data block transmission.

Table 18-26. R1 Response

	Bit 47	Bit 46	Bits 45-40	Bits 39-8	Bits 7-1	Bit 0
Description	Start Bit	Transmission Bit	Command Index	Card Status	CRC7	End Bit
Value	0	0	x	x	x	1

18.7.8.5.2 R1b—Normal Response with Busy

- Identical to R1 with an optional busy signal transmitted on the data line
- Card can become busy after receiving these commands based on its state prior to the command reception. The MMC/SD module must check for busy at the response.

18.7.8.5.3 R2—CID, CSD Register

- Response length is 136 bits
- CID contents sent as response to ALL_SEND_CID (CMD 2) and SEND_CID (CMD10)
- CSD contents sent as response to SEND_CSD (CMD9)
- Only the bits [127…1] of the CID and CSD are transferred (the reserved bit [0] of these registers is replaced by the end bit of the response)

Table 18-27. R2 Response

	Bit 135	Bit 134	Bits 133-128	Bits 127-1	Bit 0
Description	Start Bit	Transmission Bit	Reserved	CID/CSD including CRC7	End Bit
Value	0	0	111111	x	1

18.7.8.5.4 R3—OCR Register

- Response length is 48 bits
- OCR contents sent as response to SEND_OP_COND (CMD1, MMC) or SD_APP_OP_COND (ACMD41, SD)

Table 18-28. R3 Response

	Bit 47	Bit 46	Bits 45-40	Bits 39-8	Bits 7-1	Bit 0
Description	Start Bit	Transmission Bit	Reserved	OCR	Reserved	End Bit
Value	0	0	111111	x	1111111	1

18.7.8.5.5 R4—Fast I/O for MMC Only

- Response length is 48 bits
- Argument field contains the RCA of the addressed card, the register address to be read-out or written-to, and the registers' contents

Table 18-29. R4 Response

	Bit 47	Bit 46	Bits 45-40	Argument Bits 39-8			Bits 7-1	Bit 0
				Bits 39-24	Bits 23-16	Bits 15-8		
Description	Start Bit	Transmission Bit	CMD39	RCA	Register Address	Register Content	CRC7	End Bit
Value	0	0	100111	x	x	x	x	1

18.7.8.5.6 R4b—SD I/O Only

Table 18-30. R4b Response

	Bit 47	Bit 46	Bits 45-40	Argument Bits 39-8					Bits 7-1	Bit 0
				Bit 39	Bits 38-36	Bit 35	Bits 34-32	Bits 31-8		
Description	Start Bit	Direction Bit	Reserved	Card is Ready	Number of I/O Functions	Memory Present	Stuff Bits	I/O ORC	Reserved	End Bit
Value	0	0	x	x	x	x	x	x	x	1

18.7.8.5.7 R5—Interrupt Request (for MMC Only)

- Response length is 48 bits
- Argument field contains the RCA of the addressed card

Table 18-31. R5 Response

	Bit 47	Bit 46	Bits 45-40	Argument Bits 39-8		Bits 7-1	Bit 0
				Bits 39-24	Bits 23-8		
Description	Start Bit	Transmission Bit	CMD40	RCA	Reserved	CRC7	End Bit
Value	0	0	101000	x	x	x	1

18.7.8.5.8 R6—SD I/O Only

Card status bits change when SET_RELATIVE_ADDR (CMD3) is sent to an I/O only card. In this case, the 16 bits of response are the SD I/O-only values:

- Bit [15]—COM_CRC_ERROR

- Bit [14]—ILLEGAL_COMMAND
- Bit [13]—ERROR
- Bit [12:0]—Reserved

Table 18-32. R6 Response

	Bit 47	Bit 46	Bits 45-40	Argument Bits 39-8		Bits 7-1	Bit 0
				Bits 39-24	Bits 23-8		
Description	Start Bit	Direction Bit	CMD3	RCA	Card Status	CRC7	End Bit
Value	0	0	000011	x	x	x	1

Chapter 19
Memory Stick Host Controller (MSHC) Module

19.1 Overview

This chapter describes how data is transferred to a Memory Stick device and discusses how to configure and program the Memory Stick Host Controller (MSHC) module.

19.2 Features

The MSHC module provides the following features:

- Integrated 8-byte (4-half-word) FIFO buffers for transmit and receive
- Integrated CRC circuit
- Host bus clock supports HCLK maximum setting (96 MHz)
- DMA support with selectable DMA request condition based on FIFO status
- Automatic command execution (can be toggled on/off) when an interrupt from the Memory Stick is detected
- Built-in Serial Clock Divider: maximum 25 MHz serial data transfer rate
- Protocol is started by writing to the Memory Stick Command Register from the ARM920T core
- Data is requested by DMA or interrupt requests to the ARM920T core on entering the data period
- RDY time-out period can be set by the number of serial clock cycles
- Interrupt can be output to the ARM920T core when a time-out occurs
- CRC can be turned off during test mode
- Two integrated general purpose input ports
- 16-bit host bus access (byte access not supported)

19.3 Block Diagram and Description

Figure 19-1 on page 19-2 shows a high-level block diagram of the MSHC module.

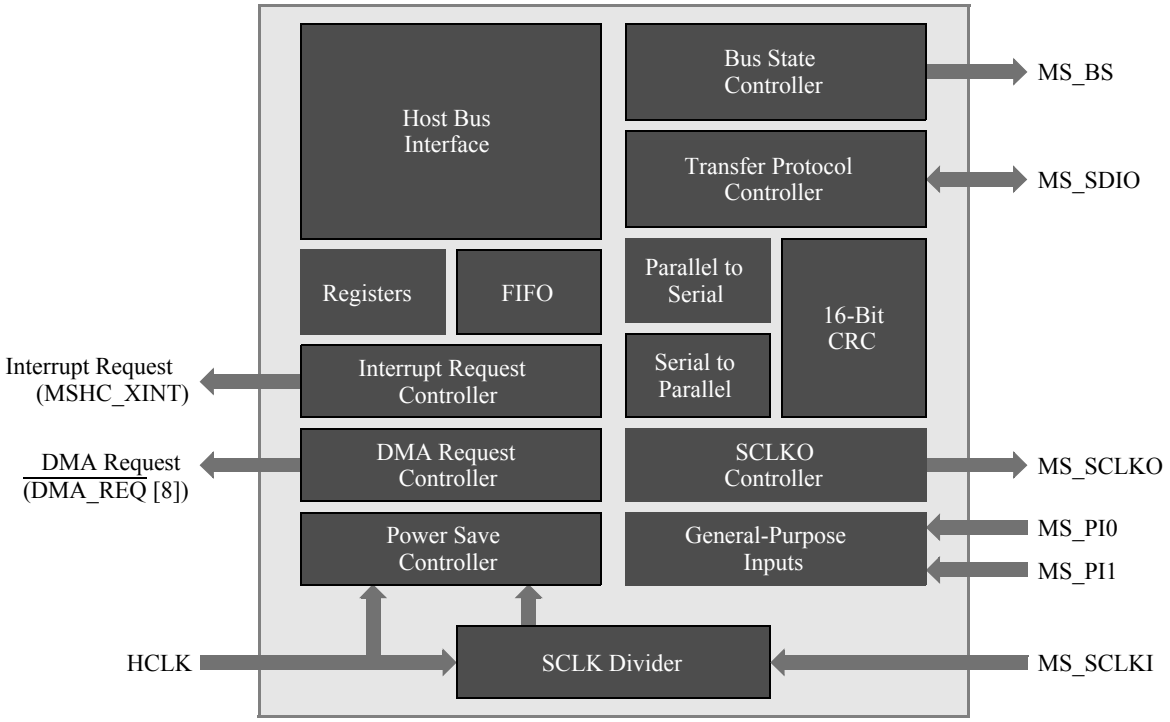


Figure 19-1. Memory Stick Host Controller Simplified Block Diagram

19.4 Memory Stick Interface

The MC9328MXL provides support for the standard Memory Stick interface. Devices that conform to both the Memory Stick form factor and protocol are supported. Figure 19-2 on page 19-3 shows the interface signals required by the Memory Stick hardware.

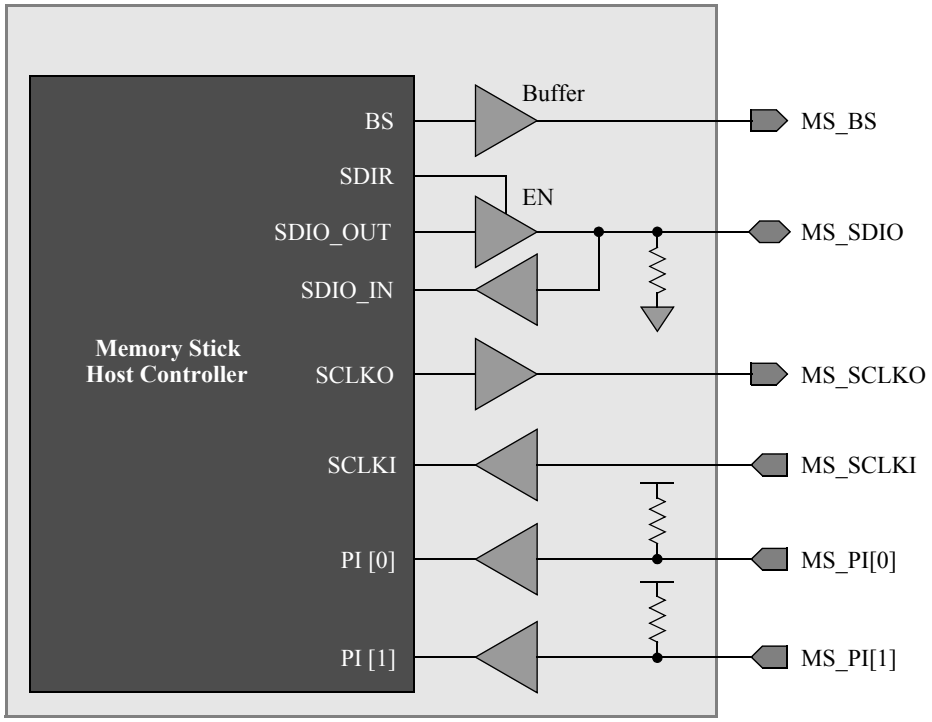


Figure 19-2. Memory Stick Interface

NOTE:

The Memory Stick interface signals are multiplexed with GPIO signals. For detailed information, see Section 29.5.1, “Data Direction Registers,” on page 29-8.

19.4.1 Signal Description

The MSHC module uses the following six signals to interface with the external Memory Stick device:

- **MS_BS**—Memory Stick Bus State (Output): Serial bus control signal.
- **MS_SDIO**—Memory Stick Serial Data Input/Output.
- **MS_SCLKO**—Memory Stick Serial Clock Output: Serial protocol clock signal.
- **MS_SCLKI**—Memory Stick External Clock Input: External clock source for the SCLK divider.
- **MS_PI0**—General Purpose Input 0. Supports Memory Stick insertion/extraction detection.
- **MS_PI1**—General Purpose Input 1. Supports Memory Stick insertion/extraction detection.

19.4.2 Pin Configuration for the MSHC Module

Section 19.4.1, “Signal Description,” includes the pins used by the MSHC module. These pins are multiplexed with other functions on the device, and must be configured for Memory Stick operation.

NOTE:

The user must ensure that the data direction bits in the GPIO are set to the correct direction for proper operation. See Section 29.5.1, “Data Direction Registers,” on

page 29-8 for details.

Table 19-1. Pin Configuration

Pin	Setting	Configuration Procedure
MS_BS	Alternate function of GPIO Port B [13]	1. Clear bit 13 of Port B GPIO In Use register (GIUS_B) 2. Set bit 13 of Port B General Purpose register (GPR_B)
MS_SCLKO	Alternate function of GPIO Port B [12]	1. Clear bit 12 of Port B GPIO In Use register (GIUS_B) 2. Set bit 12 of Port B General Purpose register (GPR_B)
MS_SDIO	Alternate function of GPIO Port B [11]	1. Clear bit 11 of Port B GPIO In Use register (GIUS_B) 2. Set bit 11 of Port B General Purpose register (GPR_B)
MS_SCLKI	Alternate function of GPIO Port B [10]	1. Clear bit 10 of Port B GPIO In Use register (GIUS_B) 2. Set bit 10 of Port B General Purpose register (GPR_B)
MS_PI1	Alternate function of GPIO Port B [9]	1. Clear bit 9 of Port B GPIO In Use register (GIUS_B) 2. Set bit 9 of Port B General Purpose register (GPR_B)
MS_PI0	Alternate function of GPIO Port B [8]	1. Clear bit 8 of Port B GPIO In Use register (GIUS_B) 2. Set bit 8 of Port B General Purpose register (GPR_B)

19.5 Memory Stick Host Controller Operation

The MSHC module consists of the following functions:

- DATA FIFO Operation
- Bus State Control Operation
- SDIO Interrupt Operation
- Reset Operation
- Power Save Mode Operation
- Auto Command Function Operation
- Serial Clock Divider Operation
- MC9328MXL System-level DMA Transfer Operation

19.5.1 Data FIFO Operation

The MSHC module features an integrated 8-byte (4-half-word) data FIFO to transmit and receive data. FIFO access is accomplished through reads from the Memory Stick Receive FIFO Data Register and writes to the Memory Stick Transmit FIFO Data Register.

For receiving data, the receive buffer empty (RBE) and receive buffer full (RBF) status flags are provided to determine the type of accesses allowed or expected. When RBE = 0, data reads from the buffer are valid, and when RBE = 1 data reads from the buffer are invalid. When the receiver FIFO is not full, RBF = 0, and when the receiver FIFO is full and requires service, RBF = 1.

For transmitting data, the transmit buffer empty (TBE) and transmit buffer full (TBF) status flags are provided. When TBE = 0, the transmit buffer contains data that is pending transmission and when TBE = 1 the buffer is empty. When TBF = 1, the transmit buffer is full and data writes to the buffer are ignored. When TBF = 0, there is room for data in the transmit buffer.

19.5.2 Bus State Control Operation

The Memory Stick protocol requires three interface signal line connections for data transfers: MS_BS, MS_SDIO, and MS_SCLKO (or MS_SCLKI). Communication is always initiated by the MSHC module and operates the bus in either four-state or two-state access mode.

The MS_BS signal classifies data on the SDIO into one of four states (BS0, BS1, BS2, or BS3) according to its attribute and transfer direction. BS0 is the INT transfer state, and during this state no packet transmissions occur. During the BS1, BS2, and BS3 states, packet communications are executed. The BS1, BS2, and BS3 states are regarded as one packet length and one communication transfer is always completed within one packet length (in four-state access mode).

The Memory Stick usually operates in four state access mode and in BS1, BS2, and BS3 bus states. When an error occurs during packet communication, the mode is shifted to two-state access mode, and the BS0 and BS1 bus states are automatically repeated to avoid a bus collision on the SDIO.

19.5.3 MSHC Module Interrupt Operation

The MSHC module provides a single interrupt to the interrupt controller. For interrupt pin assignments, see Chapter 10, “Interrupt Controller (AITC).”

19.5.3.1 Interrupt Sources

The MSHC module provides interrupt source and status flags. Generally, after $\overline{\text{MSIRQ}}$ assertion (when an interrupt event occurs), there is distinction in the MSHC module about how to clear the interrupt to the ARM920T processor (MSIRQ negate) and clearing the interrupt condition in the MSHC module (Interrupt Flag Clear). Table 19-2 summarizes the interrupt sources that assert $\overline{\text{MSIRQ}}$ to the interrupt controller.

Table 19-2. MSHC Module Interrupt Sources Summary

Interrupt Flag Name (Register)	$\overline{\text{MSIRQ}}$ Interrupt Enable Setting(s)	$\overline{\text{MSIRQ}}$ Interrupt Disable Setting(s)	Interrupt Flag Clear	$\overline{\text{MSIRQ}}$ Negate
INT (MSCS)	MSICS [INTEN] = 1	MSICS [INTEN] = 0	Read MSICS	Depends on interrupt source
RDY (MSICS)	MSICS [INTEN] = 1 MSCS [SIEN] = 1	MSICS [INTEN] = 0	Write MSCMD	Read MSICS
SIF (MSICS)	MSICS [INTEN] = 1 MSCS [SIEN] = 1 MSC2 [ACD] = 0	MSICS [INTEN] = 0 or MSC2 [ACD] = 1	Write MSCMD	Read MSICS

Table 19-2. MSHC Module Interrupt Sources Summary (continued)

Interrupt Flag Name (Register)	$\overline{\text{MSIRQ}}$ Interrupt Enable Setting(s)	$\overline{\text{MSIRQ}}$ Interrupt Disable Setting(s)	Interrupt Flag Clear	$\overline{\text{MSIRQ}}$ Negate
DRQ (MSICS)[1]	MSICS [DRQSL] = 1 MSICS [INTEN] = 1	MSICS [INTEN] = 0 or MSICS [DRQSL] = 0	Write MSTDATA for Write TPC[2] Read MSRDATA for Read TPC	Read MSICS or Write MSTDATA for Write TPC Read MSRDATA for Read TPC
PIN (MSICS)	MSICS [INTEN] = 1 MSICS [PINEN] = 1 MSPPCD [PIENx][3] = 1	MSICS [INTEN] = 0	Read MSPPCD	Read MSICS
FAE (MSICS)	MSICS [INTEN] = 1 MSFAECS [FAEEN] = 1	MSICS [INTEN] = 0	Read MSFAECS	Read MSICS
CRC (MSICS)	MSICS [INTEN] = 1 MSCS [SIEN] = 1	MSICS [INTEN] = 0	Write MSCMD	Read MSICS
TOE (MSICS)	MSICS [INTEN] = 1 MSCS [SIEN] = 1 MSCS [BSYCNT > 0	MSICS [INTEN] = 0	Write MSCMD	Read MSICS

1. DRQ (MSICS):
   When DAKEN (MSCS) = 0
   DRQ (MSICS) = 1 when Rx FIFO receives at least 1 half-word (RFF = don't care) for receive
   DRQ (MSICS) = 1 when Tx FIFO has at least 1 empty slot available (TFE = don't care) for transmit
   When DAKEN (MSCS) = 1
   DRQ (MSICS) = 1 when Rx FIFO is full (RFF = 1case) for receive
   DRQ (MSICS) = 1 when Tx FIFO is empty (TFE = 1 case) for transmit
   or
   DRQ (MSICS) = 1 when Rx FIFO receives at least 1 half-word (RFF = 0 case) for receive
   DRQ (MSICS) = 1 when Tx FIFO has at least 1 empty slot available (TFE = 0 case) for transmit
2. TPC, means Transfer Protocol Command
3. PIENx is used for either PIEN0 or PIEN1 bit of the MSPPCD register

19.5.3.2 SDIO Interrupt Operation

An interrupt transfer (INT) state from the Memory Stick to the MSHC module can occur during BS0, as shown in Figure 19-3.

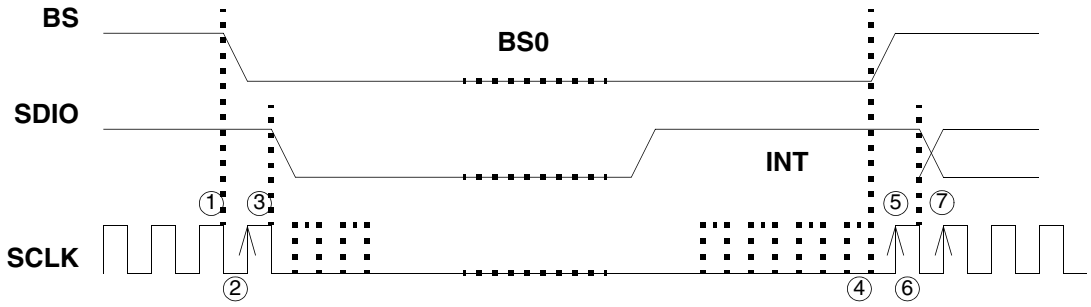


Figure 19-3. Memory Stick Interrupt Transfer State (BS0) Operation

When the Memory Stick detects BS0 at the timing indicated by point 2 in Figure 19-3, the INT transfer state is started at timing indicated at point 3. The MSHC module can terminate MS_SCLKO after the timing transition indicated by point 3 (MS_SCLKO = low). When an interrupt occurs, it is reflected in the Memory Stick Interrupt Control/Status Register and SDIO is asserted high (interrupt) by the Memory Stick. When SDIO = HIGH (INT) is detected during BS0 the INT bit of the Memory Stick Interrupt Control/Status Register (MSCR) is read and the interrupt status is checked.

The detection of INT from the Memory Stick occurs in following conditions:

1. MC9328MXL is in Normal operation mode or Doze mode and the MSHC module is enabled (MSCEN bit is set).
2. Under condition 1, the MSHC module is in Normal operation mode (PWS bit = 0).
3. The MSHC module recognizes INT when the SDIO remains high for 3 SCLK cycles during BS0.

19.5.4 Reset Operation

The Memory Stick Control/Status Register's RST bit (MSCS [RST]) provides a mechanism for software resets. When 1 is written to the RST bit, the MSHC module is reset and an associated I/O reset is initiated.

A value of 1 is maintained for the RST bit for more than SCLK 2 clocks and then must be returned to 0 to perform reset in synchronization with the clock.

A reset of the MSHC module results in:

1. Register operation (Status after RST = 1 and immediately after RST = 0)
   - Memory Stick Command Register (MSCMD) = 0x0000
   - Memory Stick Control/Status Register (MSCS) = 0x050A
   - Memory Stick Receive FIFO Data Register (MSRDATA) and Memory Stick Transmit FIFO Data Register (MSTDATA) = 0x0000
   - Memory Stick Interrupt Control/Status Register (MSICS) = 0x0080
   - Memory Stick Parallel Port Control/Data Register (MSPPCD) = 0x0000
   - Memory Stick Control 2 Register (MSC2) = ACD, RED and LEND bit = 0. MSCEN bit is not changed
   - Memory Stick Auto Command Register (MSACD) = 0x7001
   - Memory Stick FIFO Access Error Control/Status Register (MSFAECS) = 0x0000
   - Memory Stick Serial Clock Divider Register (MSCLKD) = no change

— Memory Stick DMA Request Control Register (MSDRQC) = 0x0000

2. Output signal status
   — MS_BS —> Low level
   — MS_SDIO_OUT —> Low level
   — MS_SCLK —> Low level
3. Internal operation
   — Internal Interrupt Request signal ($\overline{\text{MSIRQ}}$) —> High level (Negated)
   — Internal DMA Request signal —> High level (Negated)
   — The Transmit/Receive FIFOs are cleared
4. The executing protocol is terminated

19.5.5 Power Save Mode Operation

Figure 19-4, “Power Save Mode,” on page 19-8 depicts the Power Save Mode of the MSHC module.

NOTE:

The separate “Power Save Mode” feature of the MSHC module must not be confused with the power save mode features of the i.MX system as described in Chapter 12, “Phase-Locked Loop and Clock Controller.”

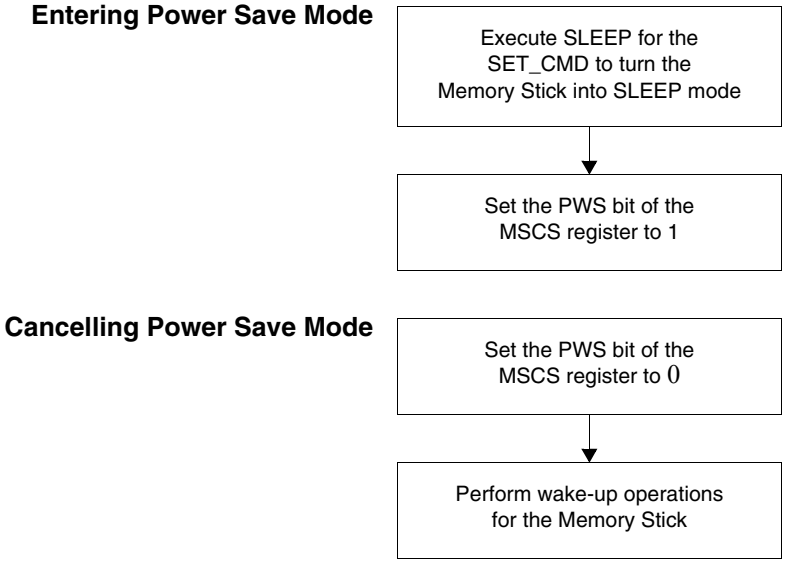


Figure 19-4. Power Save Mode

After the MSHC module is placed in power save mode (PWS = 1), the Memory Stick cannot be placed in SLEEP mode because the protocol cannot be started. The user must place the Memory Stick into SLEEP mode before placing the MSHC module in Power Save Mode (PWS = 1). Also, first cancel Power Save mode (PWS = 0) before waking up the Memory Stick.

In Power save mode, the MSHC module can detect and MS_PI [1:0] input status interrupt change. Table 19-3 shows i.MX and MSHC module Power Save Mode combination and whether or not MSHC module can detect them.

Table 19-3. Interrupt Detect Capability on Power Save Mode

MC9328MXL Power Save Mode	MSHC Module PWS	MS_PI [1:0] Interrupt	MS_SDIO Interrupt
Doze	0 (No PWS)	Detectable	Detectable
Doze	1 (PWS)	Detectable	Not detectable
Sleep	X	Not detectable	Not detectable

19.5.5.1 Register Access During Power Save Mode

Note that the following registers cannot be written while the MSHC module is in Power Save Mode (PWS bit = 1).

- MSCMD, MSTDATA, MSC2 (except MSCEN bit), MSACD, MSFAECS, and MSDRQC register

19.5.5.2 Register Access while MSHC Module is Disabled

The following register must only be written in MSHC module disable mode (MSCEN bit = 0).

- SRC bit and DIV [1:0] bits of MSCLKD register

Setting the MSCEN bit from 1 to 0 causes all of MSHC module's registers to initialize except the MSCEN bit of MSC2 register and the MSCLKD register.

The following procedure is used to initialize the MSHC module registers for operation:

1. Set the MSCEN bit of MSC2 to 1.
2. Write the value to other registers or other bits of MSC2
3. ..... etc.

19.5.6 Auto Command Function

The MSHC module supports an Auto Command function. Auto Command automatically executes GET_INT or READ_REG on the host interface for checking status after SET_CMD ends.

With this function, the INT signal from the Memory Stick is detected and the command set in the Memory Stick Auto Command Register is executed.

The result of an automatically executed command (the value of the read register) is put in the receive data buffer.

The time required and ARM920T processor load is lower with this function than when the ARM920T processor executes SET_CMD and then GET_INT (or READ_REG).

NOTE:

Be sure that READ_SIZE is set to 4 half-words or less when executing READ_REG using the ACD.

Figure 19-5 on page 19-10 indicates the ARM920T core processing and host interface operations when the ACD is used. This figure illustrates BLOCK_READ in SET_CMD execution.

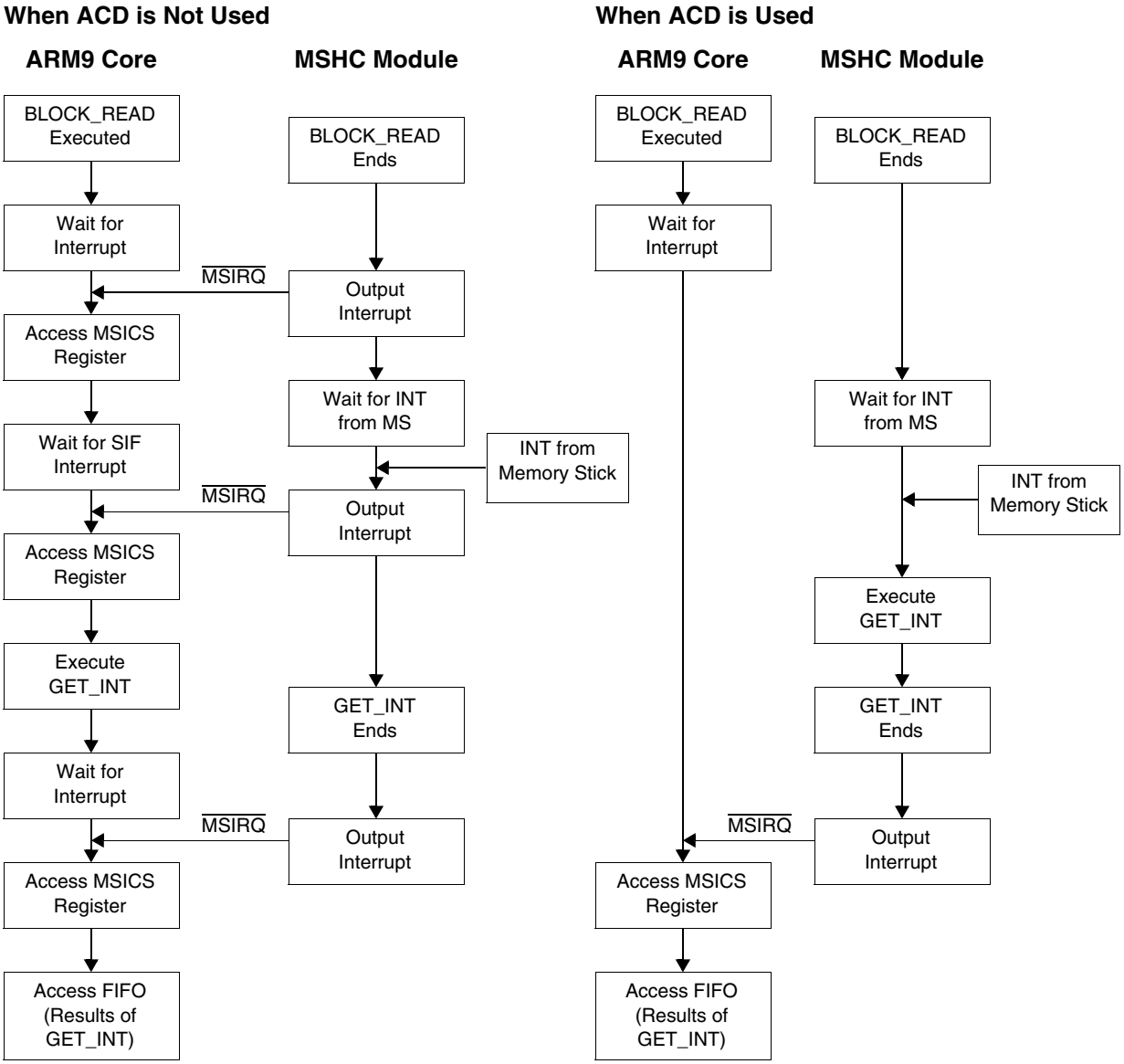


Figure 19-5. Auto Command Function Operation

NOTE:

When a CRC error or TOE occurs processing terminates without performing ACD and an interrupt signal $\overline{\text{MSIRQ}}$ is output.

Be sure to set the ACD bit of Control register 2 to 1 immediately before executing ACD (in Figure 19-5 on page 19-10, immediately before executing BLOCK_READ).

The ACD bit of Memory Stick Control register 2 is automatically set to 0 after ACD ends (in Figure 19-5 on page 19-10, after GET_INT ends).

19.5.7 Serial Clock Divider Operation

The MC9328MXL MSHC module provides for flexible transfer rate control with a configurable Serial Clock Divider. Four settings are supported by the Divider: /1, /2, /4, and /8. When internal HCLK is used, the setting /1 is provided for manufacturer testing and should not be used.

Figure 19-6 illustrates the MSHC module serial clock divider and Table 19-4 provides the settings.

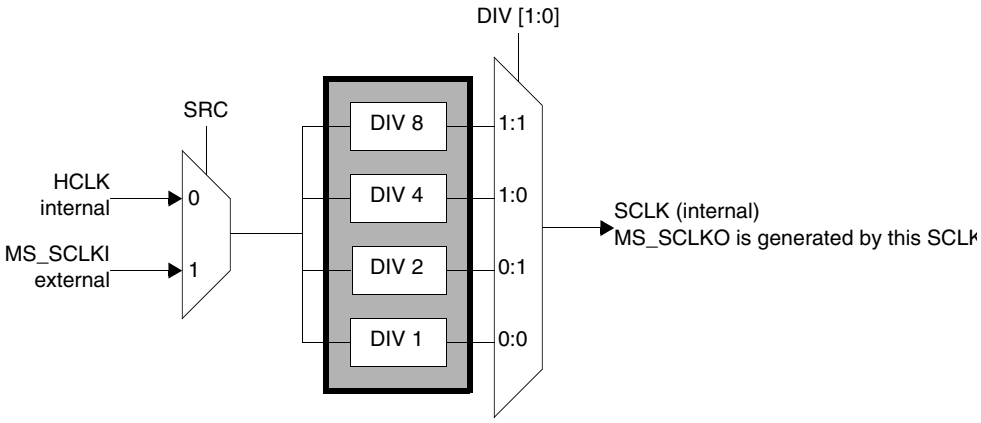


Figure 19-6. MSHC Module Serial Clock Divider

Table 19-4. Serial Clock Divider Settings

Source Select Bit (SRC Bit of MSCLKD)	Divide Bits (DIV [1:0] of MSCLKD)	Serial Clock Source	Divide Ratio	SCLKO Output
0	0:0	—	—	Not supported
0	0:1	HCLK	Divide by 2	1/2 HCLK
0	1:0	HCLK	Divide by 4	1/4 HCLK
0	1:1	HCLK	Divide by 8	1/8 HCLK
1	0:0	SCLKI	Divide by 1	SCLKI
1	0:1	SCLKI	Divide by 2	1/2 SCLKI
1	1:0	SCLKI	Divide by 4	1/4 SCLKI
1	1:1	SCLKI	Divide by 8	1/8 SCLKI

19.5.8 System-Level DMA Transfer Operation

The MSHC module DMA request bit is assigned to $\overline{\text{DMA_REQ}}$ [8] in the MC9328MXL DMA request signals. Please refer to the Chapter 13, “DMA Controller,” more detail information. Table 19-5 on page 19-12 summarizes the important data of the MSHC module to configure the DMA general and DMA I/O registers. Because the Rx and Tx share the same memory address, only 1 DMA channel can be used for DMA transfer—that is, it cannot use 2 DMA channels to handle Rx and Tx separately. This implies that the every time the MSHC module is switched from the transmit to the receive operation or the vice versa, the DMA registers need to be reconfigured.

Table 19-5. MSHC Module DMA Configuration Options

Parameter	MSHC Module RX	MSHC Module TX
FIFO SIZE	16-bit	16-bit
Memory size	32-bit	32-bit
DMA Burst length setting	2 bytes or 8 bytes	2 bytes or 8 bytes
DMA source select setting	8 dma_req [8]	8 dma_req [8]
DMA channels available for use	Channel 0–10 (FIFO to Memory)	Channel 0–10 (Memory to FIFO)
Memory address	User specified	User specified
Peripheral address	0x0021A0004 (MSRDATA)	0x0021A004 (MSTDATA)
Byte count	User specified	User specified
Request Time-out	Supported	Supported
DMA interrupt	Supported	Supported

19.6 Programming Model

The MSHC module includes 11 user-accessible 16-bit registers. All registers are 16 bits wide and 16-bit aligned. Because the MSHC module does not support byte access, the user must access on a word basis. Table 19-6 summarizes the MSHC module registers and addresses.

Table 19-6. MSHC Module Register Memory Map

Description	Name	Address
Memory Stick Command Register	MSCMD	0x0021A000
Memory Stick Control/Status Register	MSCS	0x0021A002
Memory Stick Transmit FIFO Data Register	MSTDATA	0x0021A004
Memory Stick Receive FIFO Data Register	MSRDATA	0x0021A004
Memory Stick Interrupt Control/Status Register	MSICS	0x0021A006
Memory Stick Parallel Port Control/Data Register	MSPPCD	0x0021A008
Memory Stick Control 2 Register	MSC2	0x0021A00A
Memory Stick Auto Command Register	MSACD	0x0021A00C
Memory Stick FIFO Access Error Control/Status Register	MSFAECS	0x0021A00E

Table 19-6. MSHC Module Register Memory Map (continued)

Description	Name	Address
Memory Stick Serial Clock Divider Register	MSCLKD	0x0021A010
Memory Stick DMA Request Control Register	MSDRQC	0x0021A012

19.7 Memory Stick Command Register

The Memory Stick Command Register. The bit position assignments for this register are shown in the following register display. The settings for this register are described in Table 19-7.

Protocol is started by writing to the Memory Stick Command Register. The data transfer direction is extracted from the PID code. The CRC16 bit is transferred during the data period even when the data size is 0. Disabled when the data size is 0 and the NOCRC bit of Control Register 1 is 1. Data cannot be written to the Memory Stick Command Register when the RDY bit of the MSICS register is 0 (during protocol execution).

MSCMD Memory Stick Command Register **Addr 0x0021A000**

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PID						DATA SIZE									
TYPE	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 19-7. Memory Stick Command Register Description

Name	Description	Setting
PID Bits 15–12	**Packet ID**—Holds the PID code.	0000 = Reserved 0001 = Reserved 0010 = READ_PAGE_DATA 0100 = READ_REG 0111 = GET_INT 1000 = SET_R/W_REG_ADRS 1011 = WRITE_REG 1101 = WRITE_PAGE_DATA 1110 = SET_CMD 1111 = Reserved
Reserved Bits 11–10	Reserved—These bits are reserved and should read 0.	
DATA SIZE Bits 9–0	**Data Size**—Sets the data size, in bytes, based on the PID code.	

19.7.1 Memory Stick Control/Status Register

The bit position assignments for the Memory Stick Control/Status Register are shown in the following register display. The settings for this register are described in Table 19-8.

This register is initialized on power up or when RST bit of Memory Stick Control/Status Register is 1.

MSCS Memory Stick Control/Status Register **Addr 0x0021A002**

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RST	PWS	SIEN	DAKEN	NOCRC	BSYCNT			INT	DRQ			RBE	RBF	TBE	TBF
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0
	0x050A															

Table 19-8. Memory Stick Control/Status Register Description

Name	Description	Setting
RST Bit 15	**Reset**—Resets the MSHC module.	0 = No Reset 1 = Reset MSHC module
PWS Bit 14	**Power Save**—Enables/Disables power save mode. Data can only be written to MSCS register and the MSICS register when PWS is 1. It is not possible to write to PWS while the protocol is executing.	0 = Power save disabled 1 = Power save enabled
SIEN Bit 13	**Serial Interface Enable**—Enables/Disables serial Interface output enabled. Normally set to 1 during operation.	0 = Serial interface disabled 1 = Serial interface enabled
DAKEN Bit 12	**XDAK Enable**—Configures the internal DMA transfer protocol by enabling the DMA acknowledge signal XDAK. This XDAK signal supports 4-half-word burst DMA transfer. Therefore, when the user needs to configure the module for 4-half-word DMA burst transfer mode, DAKEN bit must be set to 1. When the user needs to configure the module for 1-half-word burst DMA transfer mode, DAKEN bit can be set to either value. See NOTE of the RFF and TFE bits of the MSDRQC register.	0 = XDAK input disabled 1 = XDAK input enabled
NOCRC Bit 11	**No CRC**—Controls whether a CRC will be added to the end of the data array. Normally, this bit remains at 0 during operation.	0 = CRC on 1 = CRC off
BSYCNT Bits 10–8	**Busy Count**—Sets the maximum BSY time-out time to wait until the RDY signal is output from the card. RDY time-out error detection is not performed when BSYCNT = 0 and exceeding 5 × 4 + 2 = 22 SCLK cycles causes a RDY time-out error.	000 = No RDY time-out error detection performed 001 = 1 × 4 + 2 = 6 SCLK 010 = 2 × 4 + 2 = 10 SCLK 011 = 3 × 4 + 2 = 14 SCLK 100 = 4 × 4 + 2 = 18 SCLK 101 = 5 × 4 + 2 = 22 SCLK 110 = 6 × 4 + 2 = 26 SCLK 111 = 7 × 4 + 2 = 30 SCLK

Table 19-8. Memory Stick Control/Status Register Description (continued)

Name	Description	Setting
INT Bit 7	**Interrupt Status**—Indicates whether an interrupt condition was generated. The status will change even when the interrupt itself is disabled via the INTEN bit of the MSICS register.	0 = No interrupt condition occurred 1 = Interrupt condition occurred
DRQ Bit 6	**DMA Request**—Indicates that data was requested. The status will change even when the interrupt itself is disabled via the DRQEN bit of the MSICS register.	0 = No DMA request occurred 1 = DMA request occurred
Reserved Bits 5–4	Reserved—These bits are reserved and should read 0.	
RBE Bit 3	**Receive Buffer Empty Flag**—Indicates whether there is data in the receive buffer or not.	0 = Data available in receiver data buffer 1 = Receiver data buffer EMPTY
RBF Bit 2	**Receive Buffer Full Flag**—Indicates whether the receive buffer is full or not.	0 = Receiver data buffer NOT FULL 1 = Receiver data buffer FULL
TBE Bit 1	**Transmit Buffer Empty Flag**—Indicates whether there is data in the transmit buffer or not.	0 = Data in the transmit data buffer 1 = Transmit data buffer EMPTY
TBF Bit 0	**Transmit Buffer Full Flag**—Indicates whether the transmit buffer is full or not.	0 = Transmit data buffer NOT FULL 1 = Transmit data buffer FULL

19.7.2 Memory Stick Transmit FIFO Data Register

The write-only Memory Stick Transmit FIFO Data Register is a 16-bit register. The bit position assignments for this register are shown in the following register display. The settings for this register are described in Table 19-9.

This register's value and the FIFO pointers are initialized on power up or when RST bit of Memory Stick Control/Status Register is 1.

Big/little endian mode of the FIFO DATA register can be set by the LEND bit of the MSC2 register. The default setting is big-endian. When the LEND bit is 0, the MSHC module handles the FIFO data in big-endian. In big-endian mode, to send only one byte of data, the data byte must be written in bits 15 through 8. When the LEND bit is 1, the MSHC module handles the FIFO data in little-endian. In little-endian mode, to send only one byte of data, the data byte must be written in bits 7 through 0.

When TBF is 1, write data is ignored and it is not stored to the FIFO. The Transmit FIFO DATA register must be written only when the MSCS register's DRQ bit or MSICS register's DRQ bit is 1, and must not be written before setting a write command to the Memory Stick Command Register.

MSTDATA Memory Stick Transmit FIFO Data Register **Addr 0x0021A004**

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TX DATA BUFFER															
TYPE	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 19-9. Memory Stick Transmit FIFO Data Register Description

Name	Description
TX DATA BUFFER Bits 15–0	**Transmit FIFO Data Buffer**—Holds data to transmit. If the buffer is full (MSCS bit TBF=1), the new write data is ignored.

19.7.3 Memory Stick Receive FIFO Data Register

The read-only Memory Stick Receive FIFO Data Register is a 16-bit register. The bit position assignments for this register are shown in the following register display. The settings for this register are described in Table 19-10.

This register's value and the FIFO pointers are initialized on power up or when RST bit of Memory Stick Control/Status Register is 1.

Big/little endian mode of the FIFO DATA register can be set by the LEND bit of the MSC2 register. The default setting is big-endian. When the LEND bit is 0, the MSHC module handles the FIFO data in big-endian. In big-endian mode, one data byte incoming via the MS_SDIO pin is received to bits 15 through 8 and next data byte is received to bits 7 through 0 in MSRDATA. Therefore, when only one byte of data is received from the Memory Stick, the valid data byte is put into bits 15 through 8 in MSRDATA.

When the LEND bit is 1, the MSHC module handles the FIFO data in little-endian. In little-endian mode, one data byte incoming via the MS_SDIO pin is received to bits 7 through 0 and next byte data is received to bits 15 through 8 in MSRDATA. Therefore, when only one data byte is received from the Memory Stick, the valid data byte is put into bits 7 through 0 in MSRDATA.

When RBE is 1, invalid data is read and the FIFO read operation is ignored. The receive FIFO DATA register must be read only when the MSCS register's DRQ bit or MSICS register's DRQ bit is 1.

MSRDATA Memory Stick Receive FIFO Data Register **Addr 0x0021A004**

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RX DATA BUFFER															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 19-10. Memory Stick Receive FIFO Data Register Description

Name	Description
RX DATA BUFFER Bits 15–0	**Receiver FIFO Data Buffer**—Holds received data. If the buffer is empty (MSCS bit RBE=1), the read data is invalid.

19.7.4 Memory Stick Interrupt Control/Status Register

Memory Stick Interrupt Control/Status Register is a 16-bit register. The bit position assignments for this register are shown in the following register display. The settings for this register are described in Table 19-11.

This register is initialized on power up or when RST bit of Memory Stick Control/Status Register is 1. When the MSICS register is read by the host, the internal interrupt signal $\overline{\text{MSIRQ}}$ is set to high level (negated).

MSICS Memory Stick Interrupt Control/Status Register **Addr 0x0021A006**

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	INTEN	DRQSL	PINEN						RDY	SIF	DRQ	PIN		FAE	CRC	TOE
TYPE	rw	rw	rw	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
							0x0080									

Table 19-11. Memory Stick Interrupt Control/Status Register Description

Name	Description	Setting
INTEN Bit 15	**$\overline{\text{MSIRQ}}$ Enable**—Enables/Disables the internal interrupt request $\overline{\text{MSIRQ}}$ signal output. The $\overline{\text{MSIRQ}}$ interrupt signal is generated when an interrupt condition occurs after INTEN has been set to 1.	0 = Interrupt disabled 1 = Interrupt enabled
DRQSL Bit 14	**Data Transfer Request $\overline{\text{MSIRQ}}$ Enable**—This bit is set to 1 when MSIRQ output is enabled during data transfer request and it is set to 0 when disabled (initial value). Also, this bit enables operation of the DRQ bit of MSICS register. **Note:** The DRQSL bit must be disabled if a DMA transfer is used to/from the MSHC module's FIFO. For data transfers to the FIFO by CPU (non-DMA operation), the DRQSL bit may be enabled. Setting DRQSL to 1 must be used with DAKEN = 1. Upon detection of a Data Transfer Request interrupt (DRQ bit in the MSICS register is set) during the interrupt service routine, the MSICS register must be read after either writing to MSTDATA (for Write TPC) or reading from MSRDATA (for Read TPC). This operation is needed to accurately detect the status of the interrupt status bits.	0 = Disabled 1 = Enabled
PINEN Bit 13	**MS_PI [1:0] Level Change $\overline{\text{MSIRQ}}$ Enable**—Enables/Disables the $\overline{\text{MSIRQ}}$ output due to a change in the level at the MS_PI [1:0] pins. When a level change on MS_PI [1:0] occurs while PINEN = `0`, $\overline{\text{MSIRQ}}$ can be output when the user sets PINEN = 1. To avoid this, the user must wait more than 32 HCLKs before setting PINEN to `1` after setting the PIENx bit of MSPPCD register to `1`.	0 = Disabled 1 = Enabled
Reserved Bits 12–8	Reserved—These bits are reserved and should read 0.	
RDY Bit 7	**Ready**—Indicates whether communications with the memory stick are in progress or have ended. Clear by writing to the Memory Stick Command Register. $\overline{\text{MSIRQ}}$ asserts when RDY transitions from `0` to `1` to signal that the protocol has ended. An internal interrupt request ($\overline{\text{MSIRQ}}$) for this bit is negated by reading the MSICS register (when INTEN = 1). **Note:** Data cannot be written to the Memory Stick Command Register while the protocol is executing.	0 = Protocol in progress 1 = Protocol ended

Table 19-11. Memory Stick Interrupt Control/Status Register Description (continued)

Name	Description	Setting
SIF Bit 6	**Serial I/F Interrupt**—Indicates that a Serial I/F Interrupt has been generated. For SIF, an interrupt signal is output separately from RDY. (See Figure 19-3, "Memory Stick Interrupt Transfer State (BS0) Operation," on page 19-7). Cleared by writing to the Memory Stick Command Register. An internal interrupt request ($\overline{\text{MSIRQ}}$) for this bit is negated by reading the MSICS register (when INTEN = 1).	0 = No serial I/F Interrupt 1 = Serial I/F Interrupt generated
DRQ Bit 5	**Data Transfer Request**—Indicates that a data transfer request condition occurred. The DRQ bit can be changed only when the MSICS bit DRQSL is 1. Cleared by writing to the FIFO (when PID is a write command) or reading the FIFO (when PID is a read command), and then an internal interrupt request ($\overline{\text{MSIRQ}}$) is negated (when DRQSL = 1). Also the interrupt request ($\overline{\text{MSIRQ}}$) for this bit is negated by reading the MSICS register. **NOTE:** When the DRQEN bit of the MSDRQC register is set to 0, the internal DMA request signal is not generated even when this DRQ bit is 1.	0 = No data transfer request condition occurs 1 = Data transfer request condition occurs
PIN Bit 4	**Parallel Input**—Indicates whether the parallel input level has changed on pins MS_PI [1:0]. This bit is cleared by reading the MSPPCD. An internal interrupt request ($\overline{\text{MSIRQ}}$) for this bit is negated by reading the MSICS register (when INTEN = 1).	0 = Parallel Input level unchanged 1 = Parallel Input level change
Reserved Bit 3	Reserved—This bit is reserved and should read 0.	
FAE Bit 2	**FIFO Access Error**—Indicates that a FIFO access error occurred. Cleared when the MSFAECS register is read. This status bit is enabled/disabled with the FAEEN bit of the MSFAECS register. An internal interrupt request ($\overline{\text{MSIRQ}}$) for this bit is negated by reading the MSICS register (when INTEN = 1).	0 = No FIFO access error 1 = FIFO access error occurred
CRC Bit 1	**CRC Error**—Indicates that a CRC error occurred. Cleared when data is written to the Memory Stick Command Register. BS output is set to Low level when a CRC error occurs. Also, RDY becomes 1 and an interrupt signal is output. An internal interrupt request ($\overline{\text{MSIRQ}}$) for this bit is negated by reading the MSICS register (when INTEN = 1).	0 = No CRC error 1 = CRC error occurred
TOE Bit 0	**Time-Out Error**—Indicates that a BSY time-out error occurred. Cleared when data is written to the Memory Stick Command Register. Exceeding the number of clocks set using BSYCNT of the Control register is taken as a card malfunction and an RDY time-out error (TOE) is sent out. Also, RDY becomes 1 and an interrupt signal is output. An internal interrupt request ($\overline{\text{MSIRQ}}$) for this bit is negated by reading the MSICS register (when INTEN = 1).	0 = No BSY time-out error 1 = BSY time-out error

19.7.5 Memory Stick Parallel Port Control/Data Register

The Memory Stick Parallel Port Control/Data Register is a 16-bit register. The bit position assignments for this register are shown in the following register display. The settings for this register are described in Table 19-12.

This register is initialized on power up or when RST bit of Memory Stick Control/Status Register is 1.

Parallel input MS_PIN [1:0] is configured using two flip-flops each running at 1/16 clock.

- The parallel input pin MS_PI [1:0] is pulled up internally.
- The XPIN [1:0] bit is 1 when the MS_PI [1:0] pin is Low level, and 0 when High level.

It takes 30 HCLK cycles for a value from the parallel input pin PI [1:0] to be reflected on bits XPIN [1:0].This detection is available in following conditions:

- MC9328MXL is in Normal operation mode or Doze mode and the MSHC module must be enabled (MSCEN bit is set).
- Under the above condition, MSHC module is in Normal operation mode (PWS bit = 0) or Power Save Mode (PWS bit = 1).

MSPPCD Memory Stick Parallel Port Control/Data Register **Addr 0x0021A008**

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			PIEN1	PIEN0							XPIN1	XPIN0				
TYPE	r	r	rw	rw	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 19-12. Memory Stick Parallel Port Control/Data Register Description

Name	Description	Setting
Reserved Bits 15–14	Reserved—These bits are reserved and should read 0.	
PIEN1 Bit 13	**PIEN1**—Enables/Disables parallel port data input on MS_PI1.	0 = Parallel input port disabled 1 = Parallel input port enabled
PIEN0 Bit 12	**PIEN0**—Enables/Disables parallel port data input on MS_PI0.	0 = Parallel input port disabled 1 = Parallel input port enabled
Reserved Bits 11–6	Reserved—These bits are reserved and should read 0.	
XPIN1 Bit 5	**XPIN1**—Indicates status of the MS_PI1 pin.	0 = Parallel input port is high level 1 = Parallel input port is low level
XPIN0 Bit 4	**XPIN0**—Indicates status of the MS_PI0 pin.	0 = Parallel input port is high level 1 = Parallel input port is low level
Reserved Bits 3–0	Reserved—These bits are reserved and should read 0.	

19.7.6 Memory Stick Control 2 Register

The Memory Stick Control 2 Register is a 16-bit register. The bit position assignments for this register are shown in the following register display. The settings for this register are described in Table 19-13. For other operation description see Section 19.5.6, “Auto Command Function,” on page 19-9.

This register is initialized on power up or when RST bit of Memory Stick Control/Status Register is 1.

MSC2 Memory Stick Control 2 Register **Addr 0x0021A00A**

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ACD	RED													LEND	MSCEN
TYPE	rw	rw	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 19-13. Memory Stick Control 2 Register Description

Name	Description	Setting
ACD Bit 15	**Auto Command**—Enables/Disables auto command. When set, a command is automatically executed after an INT is detected from Memory Stick.	0 = Auto command disabled 1 = Auto command enabled
RED Bit 14	**Rise Edge Data**—Sets the edge at which serial data is loaded into the module.	0 = Serial data loaded at rising edge of the clock. 1 = Serial data loaded at falling edge of the clock
Reserved Bits 13–2	Reserved—These bits are reserved and should read 0.	
LEND Bit 1	**Little Endian Enable**—Sets the FIFO data to Big or Little Endian.	0 = Big endian 1 = Little endian
MSCEN Bit 0	**MSHC Enable**—Enables/Disables the MSHC module. **Note:** MSCEN bit is NOT reset by setting RST bit of MSCS register.	0 = MSHC module is disabled 1 = MSHC module is enabled

19.7.7 Memory Stick Auto Command Register

The Memory Stick Auto Command Register is a 16-bit register. The bit position assignments for this register are shown in the following register display. The settings for this register are described in Table 19.7.7.

MSACD Memory Stick Auto Command Register **Addr 0x0021A00C**

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	APID						ADATASIZE									
TYPE	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1
	0x7001															

Table 19-14. Memory Stick Auto Command Register Description

Name	Description	Setting
APID Bits 15–12	**Auto Command PID**—Sets the PID to be automatically executed.	0100 = READ_REG 0111 = GET_INT All Others Reserved
Reserved Bits 11–10	Reserved—These bits are reserved and should read 0.	
ADATASIZE Bits 9–0	**Auto Command Data Size**—Sets the data size in bytes.	

19.7.8 Memory Stick FIFO Access Error Control/Status Register

The Memory Stick FIFO Access Error Control/Status Register is a 16-bit register. This register's purpose is to detect an invalid FIFO access from Host bus side. For example, when Rx FIFO is empty, when Host or DMAC reads the FIFO, the access means that underrun operation is caused. When Tx FIFO is full, when Host or DMAC writes the FIFO, the access means that overrun operation is caused. However, because the FIFO's pointer does not advance in spite of these invalid accesses, user does not need to clear the FIFO in such cases. This register would be useful for debugging Host's and DMAC's FIFO access operation.

This register is initialized on power up or when RST bit of Memory Stick Control/Status Register is 1.

The bit position assignments for this register are shown in the following register display. The settings for this register are described in Table 19-15.

MSFAECS Memory Stick FIFO Access Error Control/Status Register **Addr 0x0021A00E**

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								FAEEN							RUN	TOV
TYPE	r	r	r	r	r	r	r	rw	r	r	r	r	r	r	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 19-15. Memory Stick FIFO Access Error Control/Status Register Description

Name	Description	Setting
Reserved Bits 15–9	Reserved—These bits are reserved and should read 0.	
FAEEN Bit 8	**FIFO Access Error Detection Enable**—Enables/Disables the detection of the host's invalid FIFO access on bits RUN and TOV. This bit also enables the FIFO Access Error (FAE) interrupt status bit (MSICS register).	0 = FIFO access error detection disabled 1 = FIFO access error detection enabled
Reserved Bits 7–2	Reserved—These bits are reserved and should read 0.	

Table 19-15. Memory Stick FIFO Access Error Control/Status Register Description (continued)

Name	Description	Setting
RUN Bit 1	**Rx FIFO Underrun Access**—Indicates that the host attempted a read access when the Rx FIFO is empty (RBE = 1). When this bit is set, the FAE interrupt status bit is set in MSICS register. This detection is available in case of FAEEN = 1. Clear by writing 1.	0 = No Rx FIFO access error 1 = Rx FIFO access error
TOV Bit 0	**Tx FIFO Overrun Access**—Indicates that the host attempted a write access when the Tx FIFO is full (TBF = 1). When this bit is set, the FAE interrupt status bit is also set in MSICS register. This detection is available in case of FAEEN = 1. Cleared by writing 1.	0 = No Tx FIFO access error 1 = Tx FIFO access error

19.7.9 Memory Stick Serial Clock Divider Register

The Memory Stick Serial Clock Divider Register is a 16-bit register. The bit position assignments for this register are shown in the following register display. The settings for this register are described in Table 19-16.

This register is initialized on power up.

NOTE:

This register is not initialized by the RST bit of the Memory Stick Control/Status Register.

MSCLKD Memory Stick Serial Clock Divider Register **Addr 0x0021A010**

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SRC														DIV	
TYPE	rw	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
								0x0002								

Table 19-16. Memory Stick Serial Clock Divider Register Description

Name	Description	Setting
SRC Bit 15	**Source Clock of Divider**—Selects whether the SCLKI pin or the internal HCLK will be the source clock of the divider. This bit must not be set to 1 during normal operation. **Note:** The SRC bit is NOT reset by setting the RST bit of the MSCS register. A DIV setting of 00 is not supported when SRC = 0.	0 = Select HCLK as source clock 1 = Reserved
Reserved Bits 14–2	Reserved—These bits are reserved and should read 0.	

Table 19-16. Memory Stick Serial Clock Divider Register Description (continued)

Name	Description	Setting
DIV Bits 1–0	**Divide Ratio**—Selects the divide ratio. The divider supports 2^n divide ratio, where N = 0,1,2,3. This bit must not be written after set MSCEN bit to 1. This bit must be modified only when MSCEN bit is 0(disabled). **Note:** The DIV bits are NOT reset by setting the RST bit of the MSCS register. A DIV setting of 00 is not supported when SRC = 0. **Note:** SRC = 0 and N = 0 (divide by 1) are used only for debugging purposes.	00 = Divide by 1 01 = Divide by 2 10 = Divide by 4 11 = Divide by 8

19.7.10 Memory Stick DMA Request Control Register

The Memory Stick DMA Request Control Register (MSDRQC) is a 16-bit register. The bit position assignments for MSDRQC are shown in the following register display. The settings for this register are described in Table 19-17 on page 19-23.

This register is initialized on power up or when RST bit of Memory Stick Control/Status Register is 1.

When the DMA controller is used with the MSHC module, the DMAC Burst Length register value must be either 2 bytes or 8 bytes, because that MSHC module's FIFO depth is 4 half-words (8 bytes) and the user can configure a DMA request condition either 1 half-word or 4 half-words. The following describes the MSHC module's DMA request operation in a special case.

When MSRDATA is transferred out by DMA Controller and the last burst data is less than the DMAC Burst Length register's value, then MSHC module generates a DMA request signal for the last burst transfer when receive a last byte data on the Read type TPC. This DMA request capability is needed to communicate with DMA Controller.

MSDRQC Memory Stick DMA Request Control Register **Addr 0x0021A012**

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DRQEN											RFF				TFE
TYPE	rw	r	r	r	r	r	r	r	r	r	r	rw	r	r	r	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 19-17. Memory Stick DMA Request Control Register Description

Name	Description	Setting
DRQEN Bit 15	**DMA Request Enable**—Enables/Disables operation of the DMA request signal. **Note:** This bit must be set to 1 before initiating DMA transfer.	0 = Disable DMA transfer requests 1 = Enable DMA transfer requests
Reserved Bits 14–5	Reserved—These bits are reserved and should read 0.	

Table 19-17. Memory Stick DMA Request Control Register Description (continued)

Name	Description	Setting
RFF Bit 4	**Rx FIFO Full DMA Request**—Controls the DMA request signal in case PID/APID is a read command. **Note:** This bit is effective only in case of DAKEN bit of MSCS register is 1. When DAKEN is 0, MSHC module generates DMA request in condition of RFF = 0. Therefore, when user need to use in condition of RFF = 1, DAKEN bit must be set to 1.	0 = Generate DMA request when Rx FIFO has received at least one half-word (use for DMA transfer of 1 half-word (16 bits)) 1 = Generate DMA request when Rx FIFO is full (4 half-words)
Reserved Bits 3–1	Reserved—These bits are reserved and should read 0.	
TFE Bit 0	**Tx FIFO Empty DMA Request**—Controls the DMA request signal in case of PID is a write command. **Note:** This bit is effective only in case of DAKEN bit of MSCS register is 1. When DAKEN is 0, MSHC module generates DMA request in condition of TFE = 0. Therefore, when user need to use in condition of TFE = 1, DAKEN bit must be set to 1.	0 = Generate DMA request when at least 1 empty slot is available in Tx FIFO (use for DMA transfer of 1 half-word (16 bits)) 1 = Generate DMA request when Tx FIFO is empty

19.8 Programmer's Reference

19.8.1 Memory Stick Serial Interface Overview

The Memory Stick Protocol requires 3 interface signal line connections for data transfers: MS_BS, MS_SDIO, and MS_SCLKO (or MS_SCLKI). MS_PI [1:0] pin inputs detect insertion/removal of Memory Stick. Communication is always started from the MSHC module and operates the bus in either Four State Access or Two State Access mode.

MS_BS classifies data on SDIO into one of four states (BS0, BS1, BS2 or BS3) according to the attribute and transfer direction. BS0 state has no packet communication going on while three states (BS1, BS2, and BS3) have packet communication being executed. BS1 through BS3 are regarded as one packet and one communication transfer is always completed within one packet (in Four State Access Mode).

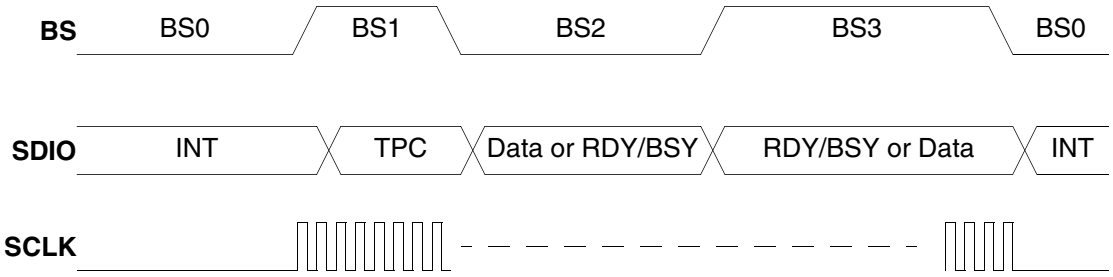


Figure 19-7. Memory Stick Bus Four State Access Protocol

Table 19-18. Serial Interface Signal Specifications

Signal	I/O Host (MSHC) Side	Description
MS_SDIO	Input/Output	Serial Data Bus. The direction of the data and the data itself changes at each Bus State. Data is 8 bit, MSB first.
MS_PI0	Input	Parallel Port Data Input. Memory Stick Insertion/Extraction detect 0.
MS_BS	Output	Indicates Bus State (0:3) on SDIO and its timing of starting transfer.
MS_PI1	Input	Parallel Port Data Input. Memory Stick Insertion/Extraction detect 1.
MS_SCLKI	Input	External Clock input to the serial clock generation circuit.
MS_SCLKO (Serial Clock)	Output	Signal on MS_BS and MS_SDIO is output on trailing edge and input (latched) at leading edge. It is always output except during BS0 period.

Table 19-19. Four State Access Mode

State	MS_BS	State Name	Description
BS0	LOW	INT Transfer State	A state in which packet communication is not active, and (MS_SDIO) is used as a transmission line for INT signals (interruption)
BS1	HIGH	TPC State	Packet Starts and transfers Transfer Protocol Command (TPC) from MSHC module to Memory Stick
BS2	LOW	Hand Shake State (Read Protocol)	Waiting for RDY signal
		Data Transfer State (Write Protocol)	Transferring data to Memory Stick
BS3	HIGH	Data Transfer State (Read Protocol)	Reading data from Memory Stick
		Handshake State (Write Protocol)	Waiting for RDY signal

Memory Stick usually operates in Four State Access Mode from BS0 through BS3. However, when an error occurs during packet communication, the mode is shifted to Two State Access Mode in which states BS0 and BS1 are automatically repeated to avoid bus collision on SDIO. See Section 19.8.3, "Transfer Protocol Command (TPC)."

19.8.2 Protocol

Bus State sequences of write packet which transfer data from the MSHC module to the Memory Stick differ from those of read packet which transfer data from the Memory Stick to the MSHC module.

19.8.2.1 Write Packet

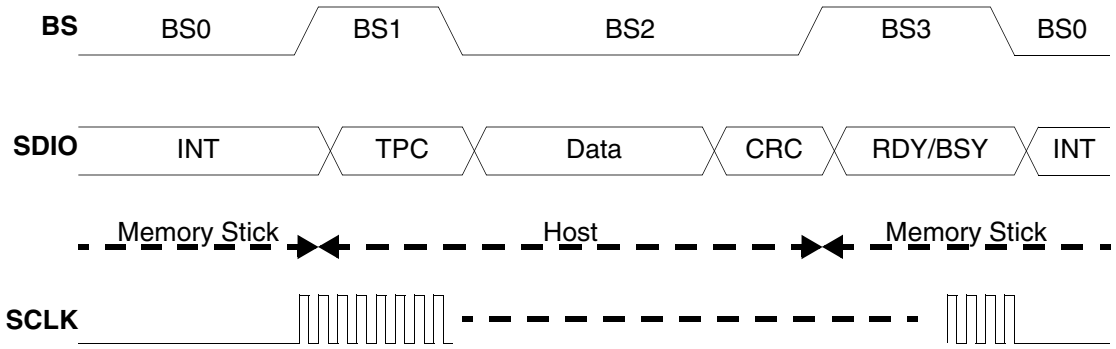


Figure 19-8. Write Packet

Table 19-20. Write Packet

Bus State	Direction	Description
BS1 (TPC)	MSHC module to Memory Stick	Transfers Write TPC
BS2 (Data)	MSHC module to Memory Stick	Transfer Data + CRC to SDIO from HOST
BS3 (Handshake)	Memory Stick to MSHC module	During BSY output (High/Low signal) on SDIO, Memory Stick decides whether packet can be terminated normally or not, and reflects the result to the corresponding register, then outputs RDY (a signal inverting at every 1SCLK) on SDIO.
BS0 (INT)	Memory Stick to MSHC module	when some interruption factors occur as a result of Memory Stick inner operation, INT (HIGH signal) is output on SDIO. During BS0 period, SDIO signal line is used as INT signal line which does not synchronize with SCLK.

19.8.2.2 Read Packet

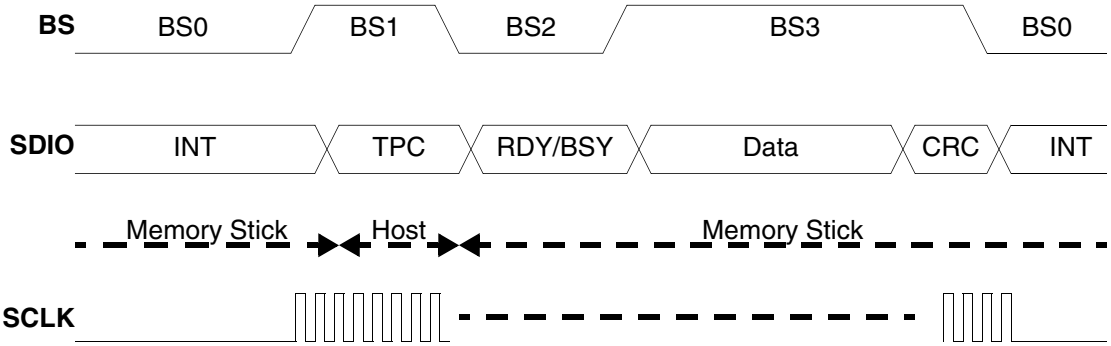


Figure 19-9. Read Packet

Table 19-21. Read Packet

Bus State	Direction	Description
BS1 (TPC)	MSHC module to Memory Stick	Transfers Read TPC.
BS2 (Handshake)	Memory Stick to MSHC module	Memory Stick outputs BSY (High/Low signal) on SDIO until reading data is ready to transfer. When ready, it outputs RDY (inverting signal at every 1SCLK).
BS3 (Data)	Memory Stick to MSHC module	Data + CRC are output on MS_SDIO from Memory Stick.
BS0 (INT)	Memory Stick to MSHC module	When some interruption factors occur as a result of Memory Stick inner operation, INT (High) signal is output on SDIO. During BS0 period, SDIO signal line is used as INT signal line which does not synchronize with SCLK.

19.8.3 Transfer Protocol Command (TPC)

The MSHC module can directly access registers and PageBuffer on the Memory Stick by TPC. The TPC code is 8-bit data which is coded by TPC 4-bit data and one's complement TPC 4-bit data for error check. TPC 4-bit code described as follows:

Table 19-22. TPC Code Specification

Name	TPC [3:0]				Operation	Description
READ_PAGE _DATA	0	0	1	0	Transfer from Page Buffer	TPC for reading from PageBuffer in units of page (= 512 bytes). Data is fixed length of 512 bytes + CRC (16-bit).
READ_REG	0	1	0	0	Read register	TPC for reading from the register which address was set. Address and Data length are set by SET_R/W_REG_ADRS. Actual Data length: the value which was set + CRC (16bit).

Table 19-22. TPC Code Specification (continued)

Name	TPC [3:0]				Operation	Description
GET_INT	0	1	1	1	Read INT register	Only INT register: 1 byte is read. Setting by SET_R/W_REG_ADRS TPC is not necessary. Read INT register operation is provided as an independent TPC, as INT register is accessed frequently. Data is fixed length of 1 byte + CRC (16-bit).
WRITE_PAGE _DATA	1	1	0	1	Transfer to PageBuffer	TPC for writing to PageBuffer in units of page (= 512 bytes). Address and Data are fixed length of 512 bytes + CRC (16-bit).
WRITE_REG	1	0	1	1	Write register	TPC for writing to the register whose address was set. Address and Data length are set by SET_R/W_REG_ADRS. Actual Data length: the value which was set + CRC (16-bit).
SET_R/W_ REG_ADRS	1	0	0	0	Address setting of READ_REG WRITE_REG	TPC for setting values which determine the register accessed by WRITE_REG and READ_REG. Values to be set are the following 4 bytes (fixed length). Data is fixed length of 4 bytes + CRC (16-bit). READ_REG: - Starting address for READ_REG: Starting address of the register to be read - Consecutive size for READ_REG: The number of registers to be read consecutively. WRITE_REG: - Starting address for WRITE_REG: Starting address of the register to be written. - Consecutive size for WRITE_REG: The number of registers to be written consecutively.
SET_CMD	1	1	1	0	Set CMD	CMD to be executed by Flash Memory Controller, such as operation for Flash Memory is transferred. Data is fixed length of CMD (8bit) + CRC (16-bit). Flash memory controller starts operation when CMD is set by this TPC, and posts the result by INT.
Reserved	0	0	0	1	—	—

19.8.4 Protocol Error

19.8.4.1 Overview

Because High and Low MS_BS signals express Bus States respectively, bus collision occurs when a difference in Bus State between Host and Memory Stick arises for some cause. To avoid this, Memory Stick shifts to Two State Access Mode automatically when an error occurs in a packet.

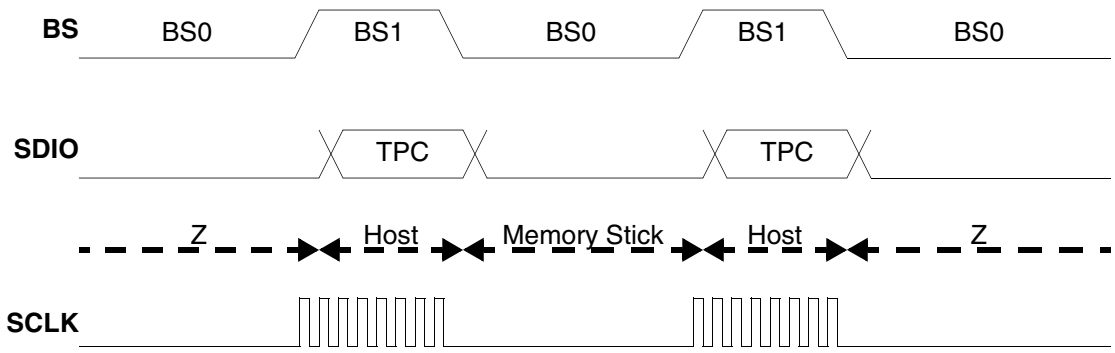


Figure 19-10. Two State Access Mode

In Two State Access Mode, operation is performed with recognition that MS_BS = LOW is BS0 and MS_BS = HIGH is BS1.

Table 19-23. Bus State in Two State Access Mode

Bus State	Direction	Description
BS0	—	Under normal conditions, BS0 is regarded as high impedance state, regardless of INT signal output period. There is no output even when INT signal is active.
BS1 (TPC)	MSHC module to Memory Stick	Memory Stick accepts TPC.

When Memory Stick shifts to Two State Access Mode, time-out occurs during the Handshake State of the packet on Host, and the failure of communicating packet is detected.

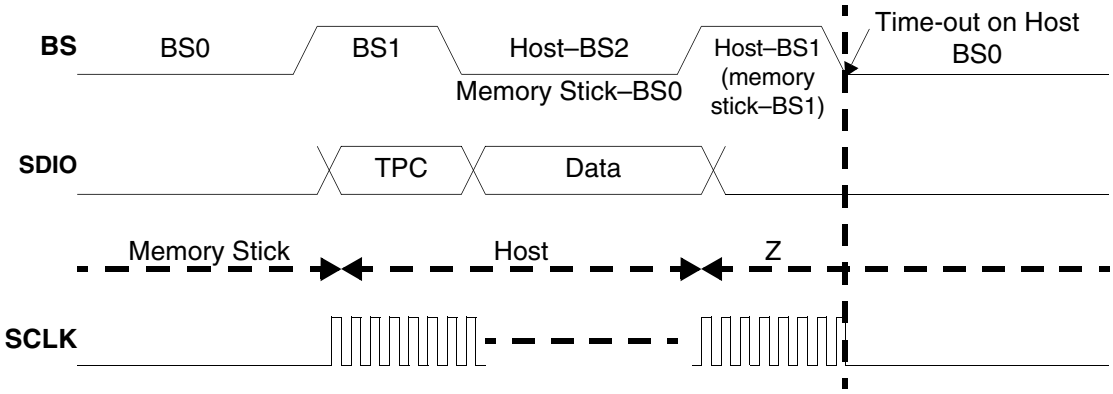


Figure 19-11. Write Packet Time-Out

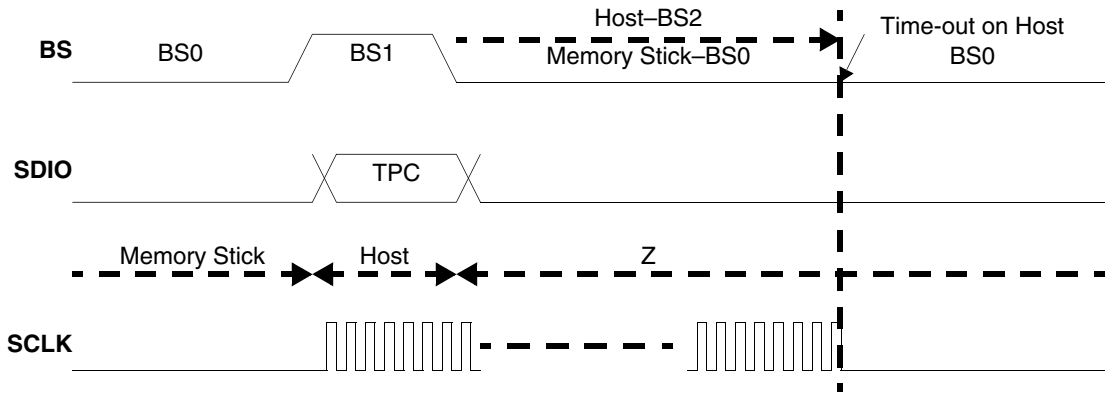


Figure 19-12. Read Packet Time-Out

NOTE:

When a time-out occurs in BS2 of a read packet, the bus state does not shift to BS3.

19.8.4.2 Two State Access Mode Factor

Table 19-24. Two State Access Mode Factor

Error	Description	
TPC Code Error	4 Bit Error Check Code Error	—
	Undefined TPC	—
	Unacceptable TPC	TPC is received, however the Memory Stick is not capable of executing it due to internal status.
	Short TPC State	When BS1 is under 8 SCLKO.
Data Error	Write Packet CRC Error	CRC error occurred in the data transferred from MSHC module.
	Short Data State	Not all data are accepted because data state of BS is shorter than the setting on Memory Stick.
Handshake Error	Short Handshake State	BS is switched before the output of RDY, though Memory Stick is operating normally.
Power Supply On	—	—

When no error occurs in BS1, the bus state shifts to BS2, BS3 and enters Four State Access Mode. When an error described above occurs, it returns to Two State Access Mode again.

19.8.5 Signal Timing

19.8.5.1 Timing

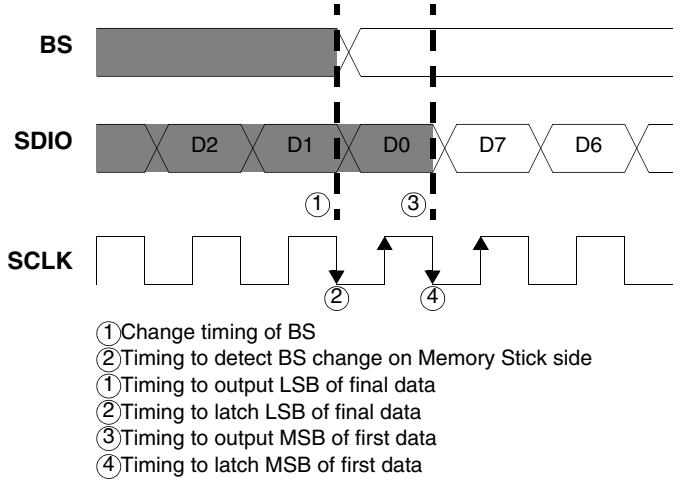


Figure 19-13. Signal Timing

- Timing of SCLK, SDIO and BS
  - Sender outputs SDIO signal at SCLK fall (output side), and latches it at SCLK rise (input side).
- BS signal is output synchronizing with SCLK fall
- Relation between BS change and data
  - When BS changes to shift to the next state, and Bus State is not extended, new BS is output synchronizing with the output timing of final data LSB on SDIO in the previous state.
- TPC, data and CRC are MSB first

19.8.5.2 Bus State Extension

When it is difficult to switch BS signal in the same timing as the final data, in TPC State and Data State, it is possible to continue that Bus State without switching the BS signal even after the final data transfer. However, in Data State, HIGH must be output on SDIO during the period when Bus State is continued without switching after the transfer of the last bit. In TPC State, signals on SDIO in this period are not prescribed.

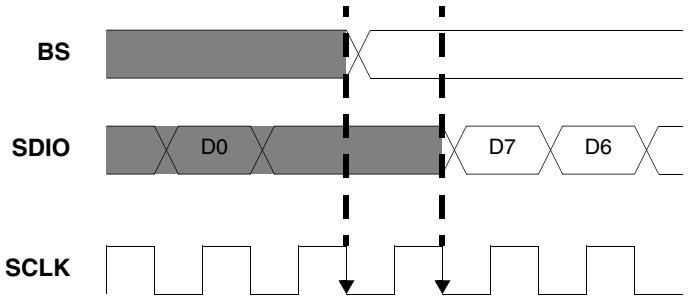


Figure 19-14. Bus State Extension

19.8.5.3 Data Transfer Extension

When data transferred from the MSHC module cannot be output to catch the next fall of SCLK, or data transferred from the Memory Stick to the MSHC module cannot be received from the rise of next SCLK because the MSHC module buffer is full, the next data transfer can be delayed by keeping SCLK high.

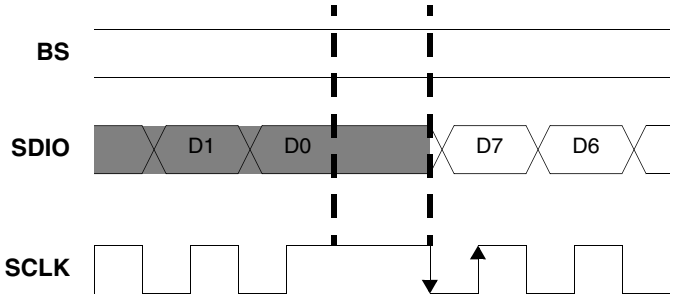


Figure 19-15. SCLK Extension for Data Wait

Chapter 20
Pulse-Width Modulator (PWM)

The pulse-width modulator (PWM) of the MC9328MXL is optimized to use 16-bit resolution and a 4×16 data FIFO to generate high quality sound from stored audio files and to generate tones. Figure 20-1 illustrates the block diagram of the pulse-width modulator.

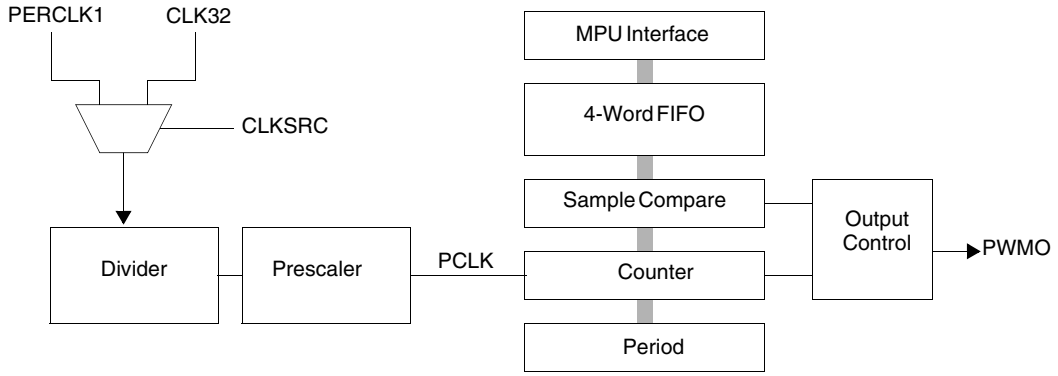


Figure 20-1. Pulse-Width Modulator Block Diagram

20.1 Introduction

The PWM can be programmed to select one of two clock signals as its source frequency. The selected clock signal is passed through a divider and a prescaler before being input to the counter. The output is available at the pulse-width modulator output (PWMO) external pin.

20.2 PWM Signals

20.2.1 Clock Signals

As shown in Figure 20-1, the clock source (CLKSRC) bit in the PWM control (PWMC) register selects the source clock—PERCLK1 (the default) or CLK32—to be used by the PWM. The selected clock signal is then sent through a divider and a prescaler to produce the PCLK signal.

The clock selection (CLKSEL) field in the PWMC selects the frequency of the output of the divider chain. The incoming clock source is divided by a binary value between 2 and 16.

For 16 kHz audio applications, CLKSEL = 01, divide by 4. For DC-level applications, CLKSEL = 11, divide by 16. See Table 20-7 on page 20-8 for a complete list of settings for the PWMC register.

Adjust the 7-bit prescaler to achieve lower sampling rates by programming the PRESCALER field in the PWMC register with any number between 0 and 127 to scale down the incoming clock source by a corresponding factor of 1 to 128.

20.2.2 Pin Configuration for PWM

Figure 20-1 shows the signals used for the PWM module. The PWMO pin is multiplexed with other functions on the device, and must be configured for PWM operation. Table 20-1 describes the procedure for the PWM pin configuration.

NOTE:

The user must ensure that the data direction bit in the GPIO is set to the correct direction for proper operation. See Section 29.5.1, “Data Direction Registers,” on page 29-8 for details.

Table 20-1. Pin Configuration

Pin	Setting	Configuration Procedure
PWMO	Primary function of GPIO Port A [2]	1.Clear bit 2 of Port A GPIO In Use register (GIUS_A) 2. Clear bit 2 of Port A General Purpose register (GPR_A)

20.3 PWM Operation

The pulse-width modulator has three modes of operation—playback, tone, and digital-to-analog (D/A) converter.

20.3.1 Playback Mode

In playback mode, the pulse-width modulator uses the data from a sound file to output the resulting audio through an external speaker. To reproduce the best quality of sound from a sound file, it is necessary to use a sampling frequency that is equal to or an even multiple of the sampling frequency used to record the sound.

The PWM produces variable-width pulses at a constant frequency. The width of the pulse is proportional to the analog voltage of a particular audio sample. At the beginning of a sample period cycle, the PWMO pin is set to 1 and the counter begins counting up from 0x0000. The sample value is compared on each count of the prescaler clock. When the sample and count values match, the PWMO signal is cleared to 0. The counter continues counting, and when it overflows from 0xFFFF to 0x0000, another sample period cycle begins. The prescaler clock (PCLK) runs 256 times faster than the sampling rate when the PERIOD field of the PWM period (PWMP) register is at its maximum value.

Figure 20-2 illustrates how variable width pulses affect an audio waveform.

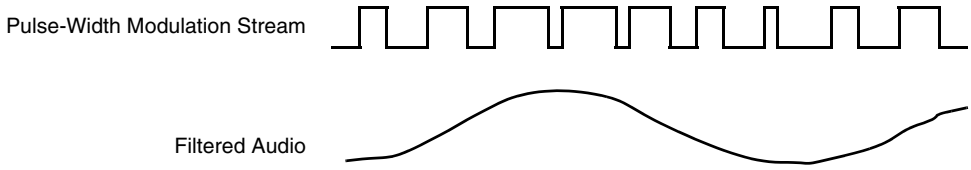


Figure 20-2. Audio Waveform Generation

Digital sample values are loaded into the pulse-width modulator as 16-bit words (big endian format). A 4-word FIFO minimizes interrupt overhead. A maskable interrupt is generated when there are 1 or 0 words in the FIFO, in which case the software can write two 16-bit samples into the FIFO. Interrupts occur every 50 µs, if the REPEAT field of the PWMC register is set to 0, when a 16 kHz sampling frequency is being used to play back sampled data, when writing two 16-bit data at each interrupt.

20.3.2 Tone Mode

When the value stored in the PWMP register < 0xFFFE, the PWM operates in tone mode and generates a continuous tone at a single frequency which is determined by the settings in the PWM registers.

20.3.3 Digital-to-Analog Converter (D/A) Mode

The pulse-width modulator outputs a frequency with a different pulse width if a low-pass filter is added at the PWMO signal. It produces a different DC level when programmed using the sample fields in the PWM sample (PWMS) register. When used in this manner, the PWM becomes a D/A converter.

20.4 Programming Model

The PWM module includes 4 user-accessible 32-bit registers. Table 20-2 summarizes these registers and their addresses.

Table 20-2. PWM Module Register Memory Map

Description	Name	Address
PWM Control Register	PWMC	0x00208000
PWM Sample Register	PWMS	0x00208004
PWM Period Register	PWMP	0x00208008
PWM Counter Register	PWMCNT	0x0020800C

20.4.1 PWM Control Register

The PWM Control Register controls the operation of the pulse-width modulator, and it also contains the status of the PWM FIFO. The register bit assignments are shown in the following register display. The register settings are described in Table 20-3 on page 20-4.

PWMC PWM Control Register **Addr 0x00208000**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
														HCTR	BCTR	SWR
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CLK SRC	PRESCALER							IRQ	IRQ EN	FIFO AV	EN	REPEAT		CLKSEL	
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	r	rw	r	rw	w		rw	
RESET	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
	0x0020*															

Table 20-3. PWM Control Register Description

Name	Description	Settings
Reserved Bits 31–19	Reserved—These bits are reserved and should read 0.	
HCTR Bit 18	**Halfword FIFO Data Swapping**—Swapping upper and lower write data which to PWMS.	0 = Do not swap 1 = Swap
BCTR Bit 17	**Byte FIFO Data Swapping**—Swapping bits[15:8] and bits[7:0] data which from PWMS to FIFO.	0 = Do not swap 1 = Swap
SWR Bit 16	**Software Reset**—Enables a reset of the PWM. After five system clock cycles, the SWR bit releases automatically.	0 = No action taken 1 = Reset the PWM
CLKSRC Bit 15	**Clock Source**—Selects the clock source for the pulse-width modulator.	0 = Selects PERCLK1 source 1 = Selects CLK32 source The CLK32 frequency is determined by the frequency of the reference crystal.
PRESCALER Bits 14–8	**Prescaler**—Scales down the incoming clock by dividing the incoming clock signal by the value contained in the PRESCALER field+1. The prescaler is normally used to generate a low single-tone PWMO signal. For voice modulation, these bits are set to 0 (divide by 1).	
IRQ Bit 7	**Interrupt Request**—Indicates that the FIFO contains 1 or 0 words, which signals the need to write no more than three 16-bit words into the PWMS register. The IRQ bit automatically clears itself after this register is read, eliminating an extra write cycle in the interrupt service routine. If the IRQEN bit is 0, the IRQ bit can be polled to indicate the status of the period comparator. For debugging purposes, the IRQ bit can be set to immediately post a PWM interrupt.	0 = FIFO contains more than 1 sample word 1 = FIFO contains 1 or 0 sample words
IRQEN Bit 6	**Interrupt Request Enable**—Enables/Disables the pulse-width modulator interrupt. When IRQEN is low, the interrupt is disabled.	0 = PWM interrupt is disabled 1 = PWM interrupt is enabled

Table 20-3. PWM Control Register Description (continued)

Name	Description	Settings
FIFOAV Bit 5	**FIFO Available**—Indicates that the FIFO is available for at least one word of sample data. Data words can be written to the FIFO as long as the FIFOAV bit is set. If the FIFO is written to while the FIFOAV bit is cleared, the write is ignored.	
EN Bit 4	**Enable**—Enables/Disables the pulse-width modulator. If the EN bit is not enabled, writes to any register in the PWM are ignored. When the pulse-width modulator is disabled, it is in low-power mode, the output pin is forced to 0, and the following events occur: • The clock prescaler is reset and frozen. • The counter is reset and frozen. • The FIFO is flushed. When the pulse-width modulator is enabled, it begins a new period, and the following events occur: • The output pin is set to start a new period. • The prescaler and counter are released and begin counting. • The IRQ bit is set, therefore indicating that the FIFO is empty.	0 = PWM is disabled 1 = PWM is enabled
REPEAT Bits 3–2	**Sample Repeats**—Selects the number of times each sample is repeated. The repeat feature reduces the interrupt overhead, reduces CPU loading when audio data is played back at a higher rate, and allows the use of a lower-cost low-pass filter. For example, if the audio data is sampled at 8 kHz and the data is played back at 8 kHz again, an 8 kHz hum (carrier) is generated during playback. To filter this carrier, a high-quality low-pass filter is required. For a higher playback rate, it is possible to reconstruct samples at 16 kHz by using the sample twice. This method shifts the carrier from an audible 8 kHz to a less sensitive 16 kHz frequency range, resulting in better output sound quality.	00 = No repeat (play the sample once) 01 = Repeat one time (play the sample twice) 10 = Repeat three times (play the sample four times) 11 = Repeat seven times (play the sample eight times)
CLKSEL Bits 1–0	**Clock Selection**—Selects the output of the sampling clock. The approximate sampling rates are calculated using a 16 MHz clock source (PRESCALER = 0 and PERIOD = default).	00 = Divide by 2—produces a 32 kHz sampling rate 01 = Divide by 4—produces a 16 kHz sampling rate 10 = Divide by 8—produces an 8 kHz sampling rate 11 = Divide by 16—provides a 4 kHz sampling rate

20.4.1.1 HCTR and BCTR Bit Description

When the endian format of the wave data stored in external memory is not compatible to the system endian format, these two bits control the swapping of the data to the PWM FIFO. The data port size used by the external memory must be used in conjunction with these two bits.

Table 20-4. HCTR and BCTR Bit Swapping

HCTR	BCTR	Swapping
0	0	No
0	1	Swapping bits[15:8] to bits[7:0] and bits[7:0] to bits[15:8]
1	0	Swapping bits[31:16] to bits[15:0] and bits[15:0] to bits[31:16]
1	1	Swapping bit[s31:16] to bits[15:0] and bits[15:0] to bits[15:0] to bits[31:16], and after that swapping bits[15:8] to bits[7:0] and bits[7:0] to bits[15:8]

Note: Only the lower 16 bits are passed to PWM 16-bit FIFO after swapping.

20.4.2 PWM Sample Register

The PWM Sample Register is the input to the FIFO. Writing successive audio sample values to this register automatically loads the values into the FIFO. The pulse-width modulator free-runs at the last set duty-cycle setting until the FIFO is reloaded or the pulse-width modulator is disabled. If the value in this register is higher than the PERIOD + 1, the output is never reset, which results in a 100% duty-cycle.

The register bit assignments are shown in the following register display. The register settings are described in Table 20-5.

PWMS PWM Sample Register **Addr 0x00208004**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SAMPLE															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	0xXXXX															

Table 20-5. PWM Sample Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
SAMPLE Bits 15–0	**Sample**—Contains a two-sample word. This word is written to the pulse-width modulator.

20.4.3 PWM Period Register

This register controls the pulse-width modulator period. When the counter value matches PERIOD + 1, the counter is reset to start another period. The following equation applies:

$$PWMO\ (Hz) = PCLK\ (Hz) \div (PERIOD + 2) \qquad \textit{Eqn. 20-1}$$

Writing 0xFFFF to this register achieves the same result as writing 0xFFFE.

The register bit assignments are shown in the following register display. The register settings are described in Table 20-6.

PWMP PWM Period Register **Addr 0x00208008**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PERIOD															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	0xFFFE															

Table 20-6. PWM Period Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
PERIOD Bits 15–0	**Period**—Represents the pulse-width modulator's period control value.

20.4.4 PWM Counter Register

The read-only PWM Counter Register contains the current count value and can be read at any time without disturbing the counter. The register bit assignments are shown in the following register display. The register settings are described in Table 20-7.

PWMCNT PWM Counter Register **Addr 0x0020800C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	COUNT															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 20-7. PWM Counter Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
COUNT Bits 15–0	**Count**—Represents the current count value.

Chapter 21
Real-Time Clock (RTC)

This chapter discusses how to operate and program the real-time clock (RTC) module that maintains the system clock, provides stopwatch, alarm, and interrupt functions, and supports the following features:

- Full clock—days, hours, minutes, seconds
- Minute countdown timer with interrupt
- Programmable daily alarm with interrupt
- Sampling timer with interrupt
- Once-per-day, once-per-hour, once-per-minute, and once-per-second interrupts
- Operation at 32.768 kHz or 32 kHz (determined by reference clock crystal)

As shown in the RTC block diagram (Figure 21-1), the real-time clock module consists of the following blocks:

- Prescaler
- Time-of-day (TOD) clock counter
- Alarm
- Sampling timer
- Minute stopwatch
- Associated control and bus interface hardware

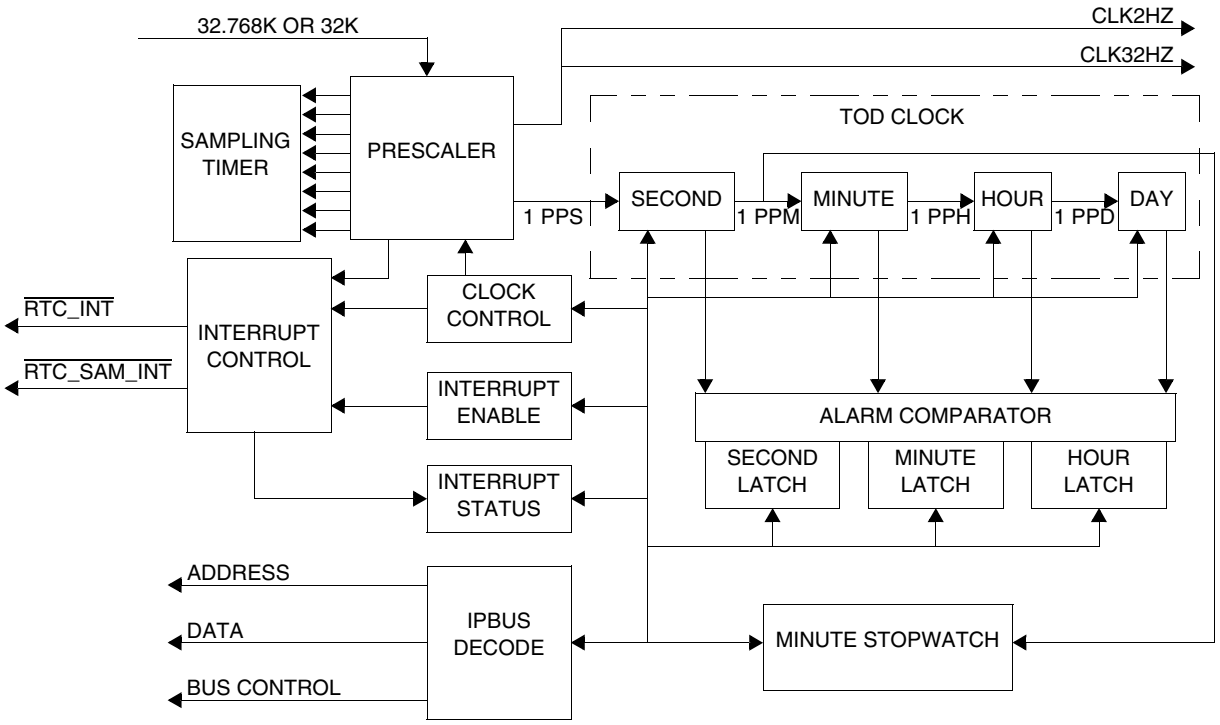


Figure 21-1. Real-Time Clock Block Diagram

21.1 Operation

The prescaler converts the incoming crystal reference clock to a 1 Hz signal which is used to increment the seconds, minutes, hours, and days TOD counters. The alarm functions, when enabled, generate RTC interrupts when the TOD settings reach programmed values. The sampling timer generates fixed-frequency interrupts, and the minute stopwatch allows for efficient interrupts on minute boundaries.

21.1.1 Prescaler and Counter

The prescaler divides the reference clock down to 1 Hz. The reference frequencies of 32.768 kHz and 32 kHz are supported. The prescaler stages are tapped to support the sampling timer.

The counter portion of the RTC module consists of four groups of counters that are physically located in three registers:

- The 6-bit seconds counter is located in the SECONDS register
- The 6-bit minutes counter and the 5-bit hours counter are located in the HOURMIN register
- The 9-bit day counter is located in the DAYR register

These counters cover a 24-hour clock over 512 days. All three registers can be read or written at any time.

Interrupts signal when each of the four counters increments, and can be used to indicate when a counter rolls over. For example, each tick of the seconds counter causes the 1 Hz interrupt flag to be set. When the seconds counter rolls from 59 to 00, the minute counter increments and the MIN interrupt flag is set. The same is true for the minute counter with the HR signal, and the hour counter with the DAY signal.

21.1.2 Alarm

There are three alarm registers that mirror the three counter registers. An alarm is set by accessing the real-time clock alarm registers (ALRM_HM, ALRM_SEC, and DAYALARM) and loading the exact time that the alarm should generate an interrupt. When the TOD clock value and the alarm value coincide, if the ALM bit in the real-time clock interrupt enable register (RTCIENR) is set, an interrupt occurs.

NOTE:

If the alarm is not disabled, it will reoccur every 512 days. If a single alarm is desired, the alarm function must be disabled through the RTC Interrupt Enable register (RTCIENR).

21.1.3 Sampling Timer

The sampling timer is designed to support application software. The sampling timer generates a periodic interrupt with the frequency specified by the SAMx bits of the RTCIENR register. This timer can be used for digitizer sampling, keyboard debouncing, or communication polling. The sampling timer operates only if the real-time clock is enabled. Table 21-1 lists the interrupt frequencies of the sampling timer for the possible reference clocks.

Multiple SAMx bits may be set in the RTC Interrupt Enable Register (RTCIENR). The corresponding bits in the RTC Interrupt Status Register (RTCISR) will be set at the noted frequencies.

Table 21-1. Sampling Timer Frequencies

Sampling Frequency	32.768 kHz Reference Clock	32 kHz Reference Clock
SAM7	512 Hz	500 Hz
SAM6	256 Hz	250 Hz
SAM5	128 Hz	125 Hz
SAM4	64 Hz	62.5 Hz
SAM3	32 Hz	31.25 Hz
SAM2	16 Hz	15.625 Hz
SAM1	8 Hz	7.8125 Hz
SAM0	4 Hz	3.90625 Hz

21.1.4 Minute Stopwatch

The minute stopwatch performs a countdown with a one minute resolution. It can be used to generate an interrupt on a minute boundary—for example, to turn off the LCD controller after five minutes of inactivity, program a value of 0x04 into the Stopwatch Count (CNT) field of the Stopwatch Minutes (STPWCH) register (see Table 21-12 on page 21-16 for a complete list of settings for the STPWCH register). At each minute, the value in the stopwatch is decremented. When the stopwatch value reaches -1, the interrupt occurs. The value of the register does not change until it is reprogrammed. Note that the actual delay includes the seconds from setting the stopwatch to the next minute tick.

21.2 Programming Model

The RTC module includes ten 32-bit registers. Table 21-2 summarizes these registers and their addresses.

Table 21-2. RTC Module Register Memory Map

Description	Name	Address
RTC Days Counter Register	DAYR	0x00204020
RTC Hours and Minutes Counter Register	HOURMIN	0x00204000
RTC Seconds Counter Register	SECONDS	0x00204004
RTC Day Alarm Register	DAYALARM	0x00204024
RTC Hours and Minutes Alarm Register	ALRM_HM	0x00204008
RTC Seconds Alarm Register	ALRM_SEC	0x0020400C
RTC Control Register	RCCTL	0x00204010
RTC Interrupt Status Register	RTCISR	0x00204014
RTC Interrupt Enable Register	RTCIENR	0x00204018
Stopwatch Minutes Register	STPWCH	0x0020401C

21.2.1 RTC Days Counter Register

The real-time clock days counter register (DAYR) is used to program the day for the TOD clock. When the HOUR field of the HOURMIN register rolls over from 23 to 00, the day counter increments. It can be read or written at any time. After a write, the time changes to the new value. This register cannot be reset because the real-time clock is always enabled at reset.

NOTE:

This day counter only supports halfword and word write operations. That means that all 9 bits must be set simultaneously.

DAYR RTC Days Counter Register **Addr 0x00204020**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								DAYS								
TYPE	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?
								0x0???								

Table 21-3. RTC Days Counter Register Description

Name	Description	Settings
Reserved Bits 31–9	Reserved—These bits are reserved and should read 0.	
DAYS Bits 8–0	**Day Setting**—Indicates the current day count.	DAYS can be set to any value between 0 and 511.

21.2.2 RTC Hours and Minutes Counter Register

The real-time clock hours and minutes counter register (HOURMIN) is used to program the hours and minutes for the TOD clock. It can be read or written at any time. After a write, the time changes to the new value. This register cannot be reset because the real-time clock is always enabled at reset.

HOURMIN RTC Hours and Minutes Counter Register **Addr 0x00204000**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				HOURS							MINUTES					
TYPE	r	r	r	rw	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	?	?	?	?	?	0	0	?	?	?	?	?	?
								0x????								

Table 21-4. RTC Hours and Minutes Counter Register Description

Name	Description	Settings
Reserved Bits 31–13	Reserved—These bits are reserved and should read 0.	
HOURS Bits 12–8	**Hour Setting**—Indicates the current hour.	HOURS can be set to any value between 0 and 23.
Reserved Bits 7–6	Reserved—These bits are reserved and should read 0.	
MINUTES Bits 5–0	**Minute Setting**—Indicates the current minute.	MINUTES can be set to any value between 0 and 59.

21.2.3 RTC Seconds Counter Register

The real-time clock seconds register (SECONDS) is used to program the seconds for the TOD clock. It can be read or written at any time. After a write, the time changes to the new value. This register cannot be reset because the real-time clock is always enabled at reset.

SECONDS RTC Seconds Counter Register **Addr 0x00204004**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											SECONDS					
TYPE	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?
								0X00??								

Table 21-5. RTC Seconds Counter Register Description

Name	Description	Settings
Reserved Bits 31–6	Reserved—These bits are reserved and should read 0.	
SECONDS Bits 5–0	**Seconds Setting**—Indicates the current second.	SECONDS can be set to any value between 0 and 59.

21.2.4 RTC Day Alarm Register

The real-time clock day alarm (DAYALARM) register is used to configure the day for the alarm. The alarm settings can be read or written at any time.

DAYALARM RTC Day Alarm Register **Addr 0x00204024**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								DAYSAL								
TYPE	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 21-6. RTC Day Alarm Register Description

Name	Description	Settings
Reserved Bits 31–9	Reserved—These bits are reserved and should read 0.	
DAYSAL Bits 8–0	**Day Setting of the Alarm**—Indicates the current day setting of the alarm.	DAYSAL can be set to any value between 0 and 511.

21.2.5 RTC Hours and Minutes Alarm Register

The real-time clock hours and minutes alarm (ALRM_HM) register is used to configure the hours and minutes setting for the alarm. The alarm settings can be read or written at any time.

ALRM_HM RTC Hours and Minutes Alarm Register **Addr 0x00204008**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				HOURS							MINUTES					
TYPE	r	r	r	rw	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 21-7. RTC Hours and Minutes Alarm Register Description

Name	Description	Settings
Reserved Bits 31–13	Reserved—These bits are reserved and should read 0.	
HOURS Bits 12–8	**Hour Setting of the Alarm**—Indicates the current hour setting of the alarm.	HOURS can be set to any value between 0 and 23.
Reserved Bits 7–6	Reserved—These bits are reserved and should read 0.	
MINUTES Bits 5–0	**Minute Setting of the Alarm**—Indicates the current minute setting of the alarm.	MINUTES can be set to any value between 0 and 59.

21.2.6 RTC Seconds Alarm Register

The real-time clock seconds alarm (ALRM_SEC) register is used to configure the seconds setting for the alarm. The alarm settings can be read or written at any time.

ALRM_SEC RTC Seconds Alarm Register **Addr 0x0020400C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											SECONDS					
TYPE	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 21-8. RTC Seconds Alarm Register Description

Name	Description	Settings
Reserved Bits 31–6	Reserved—These bits are reserved and should read 0.	
SECONDS Bits 6–0	**Seconds Setting of the Alarm**—Indicates the current seconds setting of the alarm.	SECONDS can be set to any value between 0 and 59.

21.2.7 RTC Control Register

The real-time clock control (RTCCTL) register is used to enable the real-time clock module and specify the reference frequency information for the prescaler.

RCCTL RTC Control Register **Addr 0x00204010**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									EN	XTL						SWR
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	r	r	r	r	rw
RESET	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	0x0080															

Table 21-9. RTC Control Register Description

Name	Description	Settings
Reserved Bits 31–8	Reserved—These bits are reserved and should read 0.	
EN Bit 7	**Enable**—Enables/Disables the real-time clock. The software reset bit (SWR) has no effect on this bit.	0 = Disable the real-time clock 1 = Enable the real-time clock
XTL Bits 6–5	**Crystal Selection**—Selects the proper input crystal frequency. It is important to set these bits correctly or the real-time clock will be inaccurate.	00 = 32.768 kHz 01 = 32 kHz 11 = 32.768 kHz
Reserved Bits 4–1	Reserved—These bits are reserved and should read 0.	
SWR Bit 0	**Software Reset**—Resets the module to its default state. However, a software reset will have no effect on the clock enable (EN) bit.	0 = No effect. 1 = Reset the module to its default state.

21.2.8 RTC Interrupt Status Register

The real-time clock interrupt status register (RTCISR) indicates the status of the various real-time clock interrupts, except for the 2Hz bit. When an event of the types included in this register occurs, if the corresponding bit in the RTC Interrupt Enable Register (RTCIENR) is set, then the bit will be set in this register. These bits are cleared by writing a value of 1, which also clears the interrupt. Interrupts may occur when the system clock is idle or in sleep mode. For more information about the frequency of the sampling timer interrupts (SAM7-SAM0), refer to Table 21-1.

RTCISR RTC Interrupt Status Register **Addr 0x00204014**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SAM7	SAM6	SAM5	SAM4	SAM3	SAM2	SAM1	SAM0	2HZ		HR	1HZ	DAY	ALM	MIN	SW
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 21-10. RTC Interrupt Status Register Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
SAM7 Bit 15	**Sampling Timer Interrupt Flag at SAM7 Frequency**—Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 512, 500, or 600 Hz. The actual rate of the interrupt depends on the input clock value. See Table 21-1.	0 = No SAM7 interrupt occurred 1 = A SAM7 interrupt occurred
SAM6 Bit 14	**Sampling Timer Interrupt Flag at SAM6 Frequency**—Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 256, 250, or 300 Hz. The actual rate of the interrupt depends on the input clock value. See Table 21-1.	0 = No SAM6 interrupt occurred 1 = A SAM6 interrupt occurred
SAM5 Bit 13	**Sampling Timer Interrupt Flag at SAM5 Frequency**—Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 128, 125, or 150 Hz. The actual rate of the interrupt depends on the input clock value. See Table 21-1.	0 = No SAM5 interrupt occurred 1 = A SAM5 interrupt occurred
SAM4 Bit 12	**Sampling Timer Interrupt Flag at SAM4 Frequency**—Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 64, 62.5, or 75 Hz. The actual rate of the interrupt depends on the input clock value. See Table 21-1.	0 = No SAM4 interrupt occurred 1 = A SAM4 interrupt occurred

Table 21-10. RTC Interrupt Status Register Description (continued)

Name	Description	Settings
SAM3 Bit 11	**Sampling Timer Interrupt Flag at SAM3 Frequency**—Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 32, 31.25, or 37.5 Hz. The actual rate of the interrupt depends on the input clock value. See Table 21-1.	0 = No SAM3 interrupt occurred 1 = A SAM3 interrupt occurred
SAM2 Bit 10	**Sampling Timer Interrupt Flag at SAM2 Frequency**—Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 16, 15.625, or 18.75 Hz. The actual rate of the interrupt depends on the input clock value. See Table 21-1.	0 = No SAM2 interrupt occurred 1 = A SAM2 interrupt occurred
SAM1 Bit 9	**Sampling Timer Interrupt Flag at SAM1 Frequency**—Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 8, 7.8125, or 9.375 Hz. The actual rate of the interrupt depends on the input clock value. See Table 21-1.	0 = No SAM1 interrupt occurred 1 = A SAM1 interrupt occurred
SAM0 Bit 8	**Sampling Timer Interrupt Flag at SAM0 Frequency**—Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 4, 3.90625, or 4.6875 Hz. The actual rate of the interrupt depends on the input clock value. See Table 21-1.	0 = No SAM0 interrupt occurred 1 = A SAM0 interrupt occurred
2HZ Bit 7	**2 Hz Flag**—Indicates that a 2Hz status event has occurred. If enabled, this bit is set at intervals of every 2 Hz.	0 = No 2 Hz event occurred 1 = A 2 Hz interval event occurred
Reserved Bit 6	Reserved—This bit is reserved and should read 0.	
HR Bit 5	**Hour Flag**—Indicates that the hour counter has incremented. If enabled, this bit is set on every increment of the hour counter in the time-of-day clock.	0 = No 1-hour interrupt occurred 1 = A 1-hour interrupt occurred
1HZ Bit 4	**1 Hz Flag**—Indicates that the second counter has incremented. If enabled, this bit is set on every increment of the second counter of the time-of-day clock.	0 = No 1 Hz interrupt occurred 1 = A 1 Hz interrupt occurred
DAY Bit 3	**Day Flag**—Indicates that the day counter has incremented. If enabled, this bit is set on every increment of the day counter of the time-of-day clock.	0 = No 24-hour rollover interrupt occurred 1 = A 24-hour rollover interrupt occurred
ALM Bit 2	**Alarm Flag**—Indicates that the real-time clock matches the value in the alarm registers. Note that the alarm will reoccur every 512 days. For a single alarm, clear the interrupt enable for this bit in the interrupt service routine.	0 = No alarm interrupt occurred 1 = An alarm interrupt occurred
MIN Bit 1	**Minute Flag**—Indicates that the minute counter has incremented. If enabled, this bit is set on every increment of the minute counter in the time-of-day clock.	0 = No 1-minute interrupt occurred 1 = A 1-minute interrupt occurred
SW Bit 0	**Stopwatch Flag**—Indicates that the stopwatch countdown timed out.	0 = The stopwatch did not time-out. 1 = The stopwatch timed out.

21.2.9 RTC Interrupt Enable Register

The real-time clock interrupt enable register (RTCIENR) is used to enable/disable the various real-time clock interrupts except the 2HZ bit (RTCIENR[7]). When an event of the types included in this register occurs, if that bit in this register is set, then the corresponding bit will be set in the RTC Interrupt Status Register (RTCISR). For more information about the frequency of the sampling timer interrupts (SAM7-SAM0), refer to Table 21-1.

RTCIENR RTC Interrupt Enable Register **Addr 0x00204018**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SAM7	SAM6	SAM5	SAM4	SAM3	SAM2	SAM1	SAM0	2HZ		HR	1HZ	DAY	ALM	MIN	SW
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 21-11. RTC Interrupt Enable Register Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
SAM7 Bit 15	**Sampling Timer Interrupt Flag at SAM7 Interrupt Enable**—Enables/Disables the real-time sampling timer interrupt 7. The frequency of this interrupt is shown in Table 21-1.	0 = SAM7 interrupt is disabled 1 = SAM7 interrupt is enabled
SAM6 Bit 14	**Sampling Timer Interrupt Flag at SAM6 Interrupt Enable**—Enables/Disables the real-time sampling timer interrupt 6. The frequency of this interrupt is shown in Table 21-1.	0 = SAM6 interrupt is disabled 1 = SAM6 interrupt is enabled
SAM5 Bit 13	**Sampling Timer Interrupt Flag at SAM5 Interrupt Enable**—Enables/Disables the real-time sampling timer interrupt 5. The frequency of this interrupt is shown in Table 21-1.	0 = SAM5 interrupt is disabled 1 = SAM5 interrupt is enabled
SAM4 Bit 12	**Sampling Timer Interrupt Flag at SAM4 Interrupt Enable**—Enables/Disables the real-time sampling timer interrupt 4. The frequency of this interrupt is shown in Table 21-1.	0 = SAM4 interrupt is disabled 1 = SAM4 interrupt is enabled
SAM3 Bit 11	**Sampling Timer Interrupt Flag at SAM3 Interrupt Enable**—Enables/Disables the real-time sampling timer interrupt 3. The frequency of this interrupt is shown in Table 21-1.	0 = SAM3 interrupt is disabled 1 = SAM3 interrupt is enabled
SAM2 Bit 10	**Sampling Timer Interrupt Flag at SAM2 Interrupt Enable**—Enables/Disables the real-time sampling timer interrupt 2. The frequency of this interrupt is shown in Table 21-1.	0 = SAM2 interrupt is disabled 1 = SAM2 interrupt is enabled

Table 21-11. RTC Interrupt Enable Register Description (continued)

Name	Description	Settings
SAM1 Bit 9	**Sampling Timer Interrupt Flag at SAM1 Interrupt Enable**—Enables/Disables the real-time sampling timer interrupt 1. The frequency of this interrupt is shown in Table 21-1.	0 = SAM1 interrupt is disabled 1 = SAM1 interrupt is enabled
SAM0 Bit 8	**Sampling Timer Interrupt Flag at SAM0 Interrupt Enable**—Enables/Disables the real-time sampling timer interrupt 0. The frequency of this interrupt is shown in Table 21-1.	0 = SAM0 interrupt is disabled 1 = SAM0 interrupt is enabled
2HZ Bit 7	**2 Hz Interrupt Enable**—Enables/Disables the 2 Hz bit at a 2 Hz rate.	0 = The 2 Hz clock is disabled 1 = The 2 Hz clock is enabled
Reserved Bit 6	Reserved—This bit is reserved and should read 0.	
HR Bit 5	**Hour Interrupt Enable**—Enables/Disables an interrupt whenever the hour counter of the real-time clock increments.	0 = The 1-hour interrupt is disabled 1 = The 1-hour interrupt is enabled
1HZ Bit 4	**1 Hz Interrupt Enable**—Enables/Disables an interrupt whenever the second counter of the real-time clock increments.	0 = The 1 Hz interrupt is disabled 1 = The 1 Hz interrupt is enabled
DAY Bit 3	**Day Interrupt Enable**—Enables/Disables an interrupt whenever the hours counter rolls over from 23 to 0 (midnight rollover).	0 = The 24-hour interrupt is disabled 1 = The 24-hour interrupt is enabled
ALM Bit 2	**Alarm Interrupt Enable**—Enables/Disables the alarm interrupt.	0 = The alarm interrupt is disabled 1 = The alarm interrupt is enabled
MIN Bit 1	**Minute Interrupt Enable**—Enables/Disables an interrupt whenever the minute counter of the real-time clock increments.	0 = The 1-minute interrupt is disabled 1 = The 1-minute interrupt is enabled
SW Bit 0	**Stopwatch Interrupt Enable**—Enables/Disables the stopwatch interrupt. **Note:** The stopwatch counts down and remains at decimal -1 until it is reprogrammed. If this bit is enabled with -1 (decimal) in the STPWCH register, an interrupt will be posted on the next minute tick.	0 = Stopwatch interrupt is disabled 1 = Stopwatch interrupt is enabled

21.2.10 Stopwatch Minutes Register

The stopwatch minutes (STPWCH) register contains the current stopwatch countdown value. When the minute counter of the TOD clock increments, the value in this register decrements.

STPWCH Stopwatch Minutes Register **Addr 0x0020401C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											CNT					
TYPE	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
	0x003F															

Table 21-12. Stopwatch Minutes Register Description

Name	Description	Settings
Reserved Bits 31–6	Reserved—These bits are reserved and should read 0.	
CNT Bits 5–0	**Stopwatch Count**—Contains the stopwatch countdown value. **Note:** The stopwatch counter is decremented by the minute (MIN) tick output from the real-time clock, so the average tolerance of the count is 0.5 minutes. For better accuracy, enable the stopwatch by polling the MIN bit of the RTCISR register or by polling the minute interrupt service routine.	These bits can be set to any value between 0 and 62. When the countdown has completed, the value will not change until a nonzero value (1–62) is written.

Chapter 22
SDRAM Memory Controller

This chapter describes the SDRAM controller (SDRAMC) on the MC9328MXL.

22.1 Features

The SDRAM Controller includes these distinctive features:

- Supports 4 banks of 64-, 128-, or 256-Mbit synchronous DRAMs
- Includes 2 independent chip-selects
  - — Up to 64 Mbyte per chip-select
  - — Up to four banks simultaneously active per chip-select
  - — JEDEC standard pinout and operation
- Supports Micron SyncFlash® SDRAM-interface burst flash memory
  - — Boot capability from $\overline{\text{CSD1}}$
- Supports burst reads of word (32-bit) data types
- PC100 compliant interface
  - — 100 MHz system clock achievable with “-8” option PC100 compliant memories
  - — single and fixed-length (8-word) word access
  - — Typical access time of 8-1-1-1 at 100 MHz
- Software configurable bus width, row and column sizes, and delays for differing system requirements
- Built in auto-refresh timer and state machine
- Hardware supported self-refresh entry and exit which keeps data valid during system reset and low-power modes
- Auto-powerdown (clock suspend) timer

22.2 Block Diagram

Figure 22-1 is a block diagram of the SDRAM Controller.

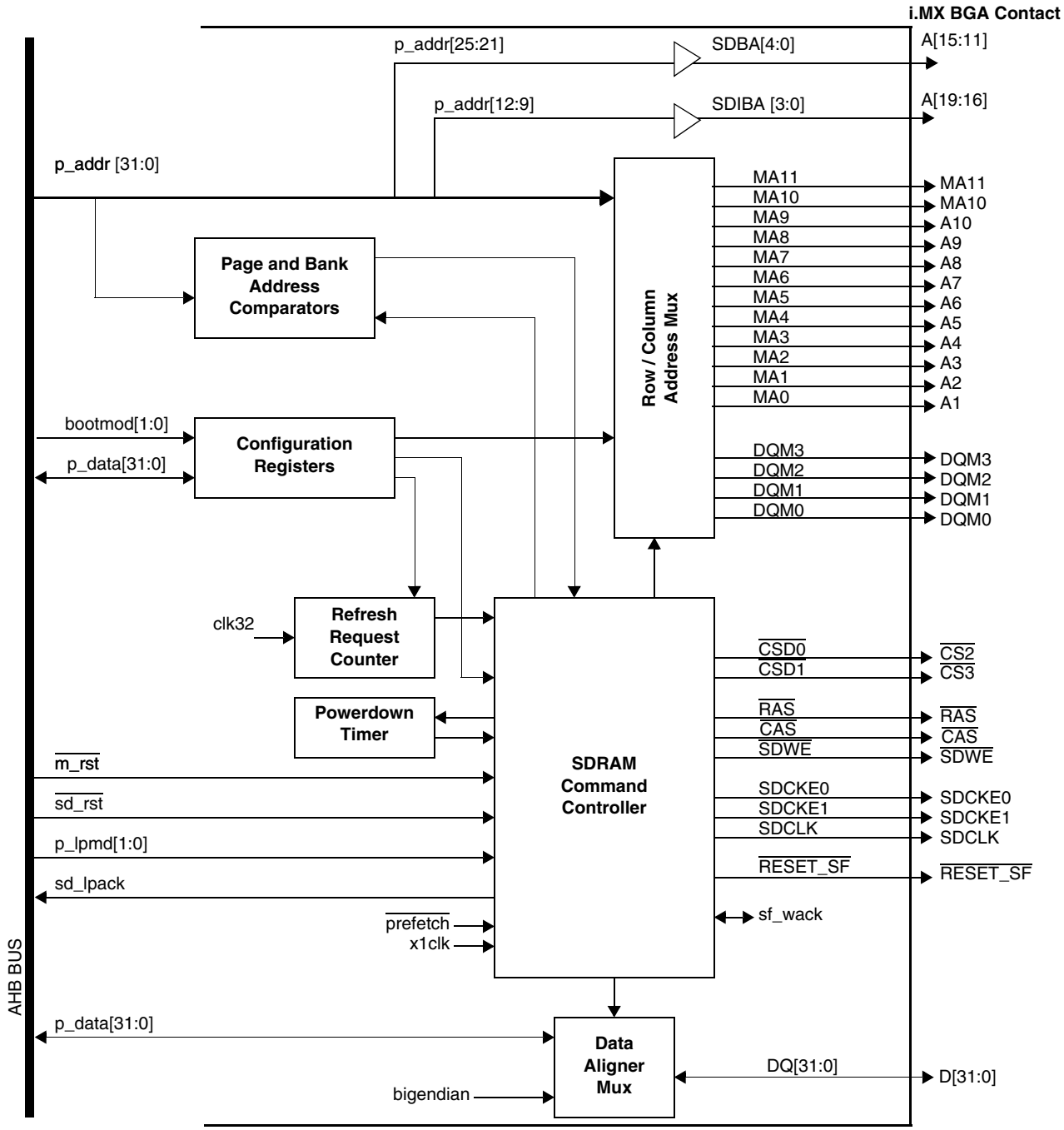


Figure 22-1. SDRAM Controller Block Diagram

22.3 Functional Overview

The SDRAM Controller consists of 7 major blocks, including the SDRAM command controller, page and bank address comparators, row and column address multiplexer, data aligner–multiplexer, configuration registers, refresh request counter, and the powerdown timer.

22.3.1 SDRAM Command Controller

The command controller handles the majority of the actions within the SDRAM controller including sequencing accesses to the memories, initializing the DRAM, keeping track of active banks within each memory region, scheduling refresh operations, transitioning into and out of low-power modes, and controlling the address and data multiplexers.

22.3.2 Page and Bank Address Comparators

There are a total of 8 address comparators. Each chip-select has a unique comparator for each of its four banks. The comparators are used to determine if a requested access falls within the address range of a currently active DRAM page.

22.3.3 Row and Column Address Multiplexer

All synchronous DRAMs incorporate a multiplexed address bus, although the address folding points vary according to memory density, data I/O size, and processor data bus width. The address folding point is described as the point where the column address bits end and the row (or bank) address begin. The SDRAM Controller takes these variables into account and provides the proper alignment of the multiplexed address through the row and column address multiplexer, non-multiplexed address pins, and the connections between the controller and the memory devices.

22.3.4 Data Aligner and Multiplexer

The data alignment block is responsible for aligning the data between the internal AHB bus and the external memory device(s) including little endian byte swapping.

22.3.5 Configuration Registers

Configuration registers determine the operating mode of the SDRAM Controller by selecting memory device density and bus width, the number of memory devices, CAS latency, row-to-column delay, and the burst length. Enable bits are provided for refresh and the auto-powerdown timer. Control bits provide a mechanism for software-initiated SDRAM initialization, SDRAM mode register settings, and all bank precharge and auto-refresh cycles.

22.3.6 Refresh Request Counter

SDRAM memories require a periodic refresh to retain data. The refresh request counter generates requests to the SDRAM Command Controller to perform these refresh cycles. Requests are scheduled according to a 32 kHz clock input with 1, 2, or 4 refresh cycles generated per clock.

22.3.7 Powerdown Timer

The powerdown timer detects periods of inactivity to the SDRAM and disables the clock when the inactive period surpasses the selected time-out. Data is retained during the powerdown state. Subsequent requests to the SDRAM incur only a minimal added start-up delay (beyond the normal access time). The powerdown timer may be programmed to expire anytime the controller is not actively reading or writing the memory, after 64 or 128 clocks of inactivity, or may be disabled entirely.

22.3.8 DMA Operation with the SDRAM Controller

The DMA controller has the capability to perform burst transfers (reads and writes) of byte and half word data types while the SDRAM controller support is restricted to internal AHB burst transfers of word (32-bit) data types. Therefore, when using the DMA in conjunction with the SDRAM controller, ensure that all burst transfers to or from the SDRAM controller are of word data types. This is configured in the DMA Channel Control register. When choosing SDRAM memory as the source address, set the Source Size bits as a 32-bit port, and likewise when setting up the destination address. Refer to the “Chapter 13, “DMA Controller,” for more details.

This section discusses input and output signals between the SDRAM Controller and the external memory devices. Other than the chip-select outputs ($\overline{CSD0}$ / $\overline{CSD1}$) and clock enables (SDCKE0 / SDCKE1), all signals are shared between the two chip-select regions. The interface signals are summarized in Table 22-1 and detailed in Section 22.3.9 through Section 22.3.19. Interconnect and timing diagrams are included as part of the detailed discussion on controller operation in Section 22.5, “Operating Modes.”

All external interface signals are referenced to the SDRAM clock, SDCLK.

Table 22-1. AHB Bus and Internal Interface Signals

Name	Function	Direction
clk32	32.0 kHz Clock Reference	Input
sf_wack	WMIMI bus time-out suppression for SyncFlash low-power mode wake-up	Output
DQ [31:0]	Internal data I/O bus	I/O

Table 22-2. SDRAM Interface Characteristics

SDRAMC Signal Name	BGA Contact Name	Function	Direction	Reset State
SDCLK	SDCLK	Clock to SDRAM	Output	Enabled
SDCKE0	SDCLKE0	Clock enable to SDRAM 0	Output	High
SDCKE1	SDCLKE1	Clock enable to SDRAM 1	Output	High
$\overline{CSD0}$	$\overline{CS2}$	Chip-select to SDRAM array 0	Output	High
$\overline{CSD1}$	$\overline{CS3}$	Chip-select to SDRAM array 1	Output	High
MA [11:10]	MA [11:10]	Multiplexed Address	Output	Low
MA [9:0]	A [10:1]	Multiplexed Address	Output	Low
SDBA [4:0] / A [25:21]	A [15:11]	Non-multiplexed Address	Output	Low
SDIBA [3:0] / A [12:9]	A [19:16]	Non-multiplexed Address	Output	Low
DQM3	DQM3	Data Qualifier Mask byte 3 (D [31:24])	Output	Low

Table 22-2. SDRAM Interface Characteristics (continued)

SDRAMC Signal Name	BGA Contact Name	Function	Direction	Reset State
DQM2	DQM2	Data Qualifier Mask byte 2 (D [23:16])	Output	Low
DQM1	DQM1	Data Qualifier Mask byte 1 (D [15:8])	Output	Low
DQM0	DQM0	Data Qualifier Mask byte 0 (D [7:0])	Output	Low
DQ[31:0]	D[31:0]	Data bus	I/O	High
$\overline{\text{SDWE}}$	$\overline{\text{SDWE}}$	Write Enable	Output	High
$\overline{\text{RAS}}$	$\overline{\text{RAS}}$	Row Address Strobe	Output	High
$\overline{\text{CAS}}$	$\overline{\text{CAS}}$	Column Address Strobe	Output	High
$\overline{\text{RESET_SF}}$	$\overline{\text{RESET_SF}}$	SyncFlash Reset/Powerdown	Output	Low

22.3.9 SDCLK—SDRAM Clock

The SDCLK output provides the timing reference for the memory devices. All other SDRAM interface signals are referenced to this clock. SDCLK is synchronous to the system clock, however it is gated off during low-power operating modes when both SDCKE0 and SDCKE1 are negated.

22.3.10 SDCKE0, SDCKE1—SDRAM Clock Enables

The SDCKE0 and SDCKE1 pins are clock enable outputs to the SDRAM memory devices. SDCKE0 corresponds to SDRAM array 0 and SDCKE1 to SDRAM array 1. When these pins are asserted high, the memory's clock input is active, which means that a stable clock is being supplied. The low assertion deactivates the memory's clock input. A low assertion of SDCKEx initiates Powerdown, Self Refresh, and Suspend modes to the SDRAM.

22.3.11 $\overline{\text{CSD0}}$, $\overline{\text{CSD1}}$—SDRAM Chip-Select

$\overline{\text{CSD0}}$ and $\overline{\text{CSD1}}$ are used to select SDRAM array 0 and SDRAM array 1, respectively. When a valid command is present on the other control signals, the chip-select signals are used to indicate which device the command is directed towards.

22.3.12 DQ [31:0]—Data Bus (Internal)

The 32 data lines are used to transfer data between the SDRAM Controller and memory. Data bit 31 is the most significant bit and bit 0 is the least significant.

22.3.13 MA [11:0]—Multiplexed Address Bus

The multiplexed address bus specifies the SDRAM page and the location within the page targeted by the current access. The multiplexed address pins are used in conjunction with some of the non-multiplexed ARM920T processor address signals to comprise the complete SDRAM address. Connections between the SDRAM Controller and memory vary depending on the SDRAM device density. See Section 22.6.1, "Address Multiplexing," on page 22-29 and specifically Table 22-20 and Table 22-22 for details on supported SDRAM configurations.

22.3.14 SDBA [4:0], SDIBA [3:0]—Non-Multiplexed Address Bus

The non-multiplexed address pins specify the SDRAM bank to which the current command is targeted. In some density or width configurations, these pins also supply the most significant bits of the row address. Table 22-20 on page 22-46 and Table 22-21 on page 22-47 document which address pins are used for any given configuration.

22.3.15 DQM3, DQM2, DQM1, DQM0—Data Qualifier Mask

During read cycles, the DQMx pins control the SDRAM data output buffers. DQMx asserted high disables the output buffers leaving them in a high-impedance state. DQMx asserted low allows the data buffers to drive normally.

During write cycles, DQMx controls which bytes are written in the SDRAM. DQMx asserted low enables a write to the corresponding byte, whereas DQMx asserted high leaves the byte unchanged.

DQM3 corresponds to the most significant byte and DQM0 to the least significant. Sixteen bit memories require only two DQM connections. Memories aligned to the upper data bus (D [31:16]) connect to DQM3 and DQM2, while memories aligned to the lower data bus (D [15:0]) connect to DQM1 and DQM0. Memory alignment is selected in the SDCTLx Registers.

22.3.16 $\overline{\text{SDWE}}$—Write Enable

Write enable is part of the three bit command field ($\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ make up the other two bits) used by the SDRAM. Generally, $\overline{\text{SDWE}}$ will be asserted low if a command transfers data to the memory. A detailed summary of the supported SDRAM commands is provided in Table 22-50 on page 22-71.

22.3.17 $\overline{\text{RAS}}$—Row Address Strobe

Row address strobe is also part of the SDRAM command field. It is generally used to indicate an operation affecting an entire bank or row. When $\overline{\text{RAS}}$ is asserted (low), a new SDRAM row address must be latched. Table 22-50 on page 22-71 provides details on SDRAM command encoding.

22.3.18 $\overline{\text{CAS}}$—Column Address Strobe

The column address strobe is the third signal comprised in the command field. It generally signifies a column oriented command. When $\overline{\text{CAS}}$ is asserted (low), the column address has changed. Table 22-50 on page 22-71 provides details on SDRAM command encoding.

22.3.19 $\overline{\text{RESET_SF}}$—Reset or Powerdown

This output signal is only used for SyncFlash memories. When asserted low, the SyncFlash memory state machines are reset and the device is placed in the lowest power operating mode. Bringing this signal high returns the device to a normal operating condition, ready to accept commands following a stabilization delay.

Three conditions force the $\overline{\text{RESET_SF}}$ pin to be asserted: SDRAM reset ($\overline{\text{SD_RST}}$), disabling the chip-select via the SDE bit in the SDCTL1 control register, and entering one of the low-power modes. In each case, the system integrator must guarantee that the SyncFlash stabilization period between $\overline{\text{RESET_SF}}$ negation and the first access to the device has been met.

NOTE:

Programming hardware-protected blocks within the SyncFlash requires RESET_SF to be raised to 5V +/-10%. This feature is not supported by the SDRAM Controller.

22.3.20 Pin Configuration for SDRAMC

Table 22-3 lists the pins used for the SDRAM controller. These pins are multiplexed with other functions on the device, and must be configured for SDRAM operation.

NOTE:

The user must ensure that the data direction bits in the GPIO are set to the correct direction for proper operation. See Section 29.5.1, “Data Direction Registers,” on page 29-8 for details.

Table 22-3. Pin Configuration

Pin	Setting	Set-Up Procedure
SDCLK	Not multiplexed	
SDCKE0	Not multiplexed	
SDCKE1	Not multiplexed	
$\overline{\text{CSD0}}$	Alternate function of $\overline{\text{CS2}}$/$\overline{\text{CSD0}}$ pin	Set bit 0 (SDCS0_SEL) in the Function Multiplexing Control register of the System Control module.
$\overline{\text{CSD1}}$	Alternate function of $\overline{\text{CS3}}$/$\overline{\text{CSD1}}$ pin	Set bit 1 (SDCS1_SEL) in the Function Multiplexing Control register of the System Control module.
MA [11:10]	Not multiplexed	
MA [9:0]	Multiplexed with A [10:1]	Internal signal from SDRAMC, asserted for SDRAM accesses
SDBA [4:0]	Multiplexed with A [15:11]	Internal signal from SDRAMC, asserted for SDRAM accesses
SDIBA [3:0]	Multiplexed with A [19:16]	Internal signal from SDRAMC, asserted for SDRAM accesses
DQM3	Not multiplexed	
DQM2	Not multiplexed	
DQM1	Not multiplexed	
DQM0	Not multiplexed	
$\overline{\text{SDWE}}$	Not multiplexed	
$\overline{\text{RAS}}$	Not multiplexed	
$\overline{\text{CAS}}$	Not multiplexed	
$\overline{\text{RESET_SF}}$	Not multiplexed	

22.4 Programming Model

The SDRAM controller includes four 32-bit registers. Table 22-4 summarizes these registers and their addresses. Registers are accessible in supervisor mode only. Attempts to access the registers in user mode will result in a transfer error (TEA) being returned on the ARM920T processor's local bus.

Table 22-4. SDRAM Module Register Memory Map

Description	Name	Address
SDRAM 0 Control Register	SDCTL0	0x00221000
SDRAM 1 Control Register	SDCTL1	0x00221004
SDRAM Reset Register	SDRST	0x00221018
Miscellaneous Register	MISCELLANEOUS	0x00221014

The SDRAM arrays are mapped according to Table 22-5. A 64 Mbyte space is allocated to each chip-select. Memories smaller than the allocated region are redundantly mapped throughout the remainder of the region. Attempted accesses to a disabled chip-select region (SDE bit of the SDCTLx = 0) and User accesses to a protected (SP bit of the SDCTL = 1) region will result in a transfer error.

Table 22-5. SDRAM Array Memory Map

Address	Use	Access
0x0800 0000 – 0x0BFF FFFF	SDRAM 0 Memory array	R/W
0x0C00 0000 – 0x0FFF FFFF	SDRAM 1 Memory array	R/W

22.4.1 SDRAM Control Registers

There are two SDRAM Control Registers, one for each of the two memory regions. SDCTL0 defines the operating characteristics for the SDRAM 0 region (selected by $\overline{CSD0}$), while SDCTL1 does the same for the SDRAM 1 region (selected by $\overline{CSD1}$). Bit and field assignments within the registers are identical.

		Addr
SDCTL0	SDRAM 0 Control Register	**0x00221000**
SDCTL1	SDRAM 1 Control Register	**0x00221004**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SDE	SMODE			SP		ROW				COL		IAM		DSIZ	
TYPE	rw	rw	rw	rw	rw	r	rw	rw	r	r	rw	rw	rw	r	rw	rw
RESET	0*	0	0	0	0	0	0	1	0	0	0	0	0	0	0*	0*
	0x0100*															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SREFR		CLKST				SCL			SRP	SRCD			SRC		
TYPE	rw	rw	rw	rw	r	r	rw	rw	r	rw	rw	rw	r	rw	rw	rw
RESET	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
	0x0300															

* For SDCTL1, the reset state is affected by bootmod [1:0]

Table 22-6. SDRAM 0 Control Register and SDRAM 1 Control Register Description

Name	Description	Settings
SDE Bit 31	**SDRAM Controller Enable**—Enables/Disables the SDRAM controller. The module is disabled on reset. For SDRAM 1, if the selected boot mode is the SyncFlash memory, the module will be enabled on reset. Disabling the module shuts off all clocks within the module with the exception of register accesses.	0 = Disabled 1 = Enabled
SMODE Bits 30–28	**SDRAM Controller Operating Mode**—Determines the operating mode of the SDRAM controller. The controller is capable of operating in six different modes. These modes are primarily used for SDRAM initialization and programming of the SyncFlash memory. Any access to the SDRAM memory space, while in one of the alternate modes, will result in the corresponding special cycle being run. Moving from Normal to any other mode does not close (precharge) any banks that may be open (activated). Under most circumstances, software must run a precharge-all cycle when transitioning out of Normal Read/Write mode. SyncFlash command register read/write sequences are the one exception that does not require a software initiated precharge. Operating mode details are provided in Section 22.7, “SDRAM Operation,” and Section 22.8, “SyncFlash Operation.” Reset initializes the operating mode to “Normal Read/Write”.	000 = Normal Read/Write 001 = Precharge Command 010 = Auto-Refresh Command 011 = Set Mode Register Command 100 = Reserved 101 = Reserved 110 = SyncFlash Load Command Register 111 = SyncFlash Program Read/Write

Table 22-6. SDRAM 0 Control Register and SDRAM 1 Control Register Description (continued)

Name	Description	Settings
SP Bit 27	**Supervisor Protect**—Restricts user accesses within the chip-select region.	0 = User mode accesses are allowed to this chip-select region. 1 = User mode accesses are prohibited. An attempted access to this chip-select region while in user mode will result in a BUS ERROR being returned back to the CPU. The chip-select will not be asserted. Read Accesses are not affected.
Reserved Bit 26	Reserved—This bit is reserved and should read 0.	
ROW Bits 25–24	**Row Address Width**—Specifies the number of row addresses used by the memory array. This number does not include the bank, column, or data qualifier addresses. Parameters affected by the programming of this field include the page-hit address comparators and the bank address bit locations (non-interleaved mode only).	00 = 11 01 = 12 10 = 13 11 = Reserved
Reserved Bits 23–22	Reserved—These bits are reserved and should read 0.	
COL Bits 21–20	**Column Address Width**—Specifies the number of column addresses in the memory array and will determine the break point in the address multiplexer. Column width is the number of multiplexed column addresses and does not include bank and row addresses, or addresses used to generate the DQM signals.	00 = 8 01 = 9 10 = 10 11 = 11

Table 22-6. SDRAM 0 Control Register and SDRAM 1 Control Register Description (continued)

Name	Description	Settings
IAM Bit 19	**(Bank[1]) Interleaved Address Mode**—Controls bank address alignment. Bank addresses fall between the row and column addresses, resulting in an interleaved memory map with the banks alternately striped through the memory region. They are more significant than row and column addresses and result in a linear addressing of the banks through the memory map. Bank address bit placement has a significant effect on how well the SDRAM page buffers are used, with a corresponding impact on system performance. The SDRAM Controller supports two bank address alignments which will satisfy most system requirements. See Figure 22-2 for memory bank interleaving options. **Note:** Memory address linearity is of little concern to the user when the memory is comprised of RAM devices, however, it is of utmost importance when the memory is not-volatile and erase/program is block oriented. Flash memories are an example of a block oriented memory. Choosing the interleaved (IAM = 1) address option will generally require the user to erase the memory in multiples of 4 times the block size (for example, 256k x 4, or 1 Mbyte for the 64 Mbit Micron SyncFlash). Programming can still occur on a page by page basis, however executing out of the device being programmed may not be possible because the code is likely to cross a page boundary into the bank being programmed. If the device is programmed outside the system, the data must be shuffled to account for the interleaved blocks. For these reasons, it is recommended that the user choose linear addressing (IAM = 0) for block oriented devices.	0 = Linear Address Map. Addresses flow through each page in the first bank, into the second bank, and so on. See Figure 22-2. Linear Addressing is best suited to applications with large continuous blocks of linear accessed data such as an LCD display buffer. 1 = Interleaved Address Map. Addresses flow through one page in the first bank, to one page in the second, to the third, etc. The banks alternate at each SDRAM page boundary. See Figure 22-2. Interleaved Mapping is better suited for ARM9 code space. The interleaving of the banks eliminates the need to continually open and close pages when loops and LRW constants cross page boundaries, resulting in higher system throughput.
Reserved Bit 18	Reserved—This bit is reserved and should read 0.	
DSIZ Bits 17–16	**SDRAM Memory Data Width**—Defines the width of the SDRAM memory and its alignment on the external data bus. 16-bit ports may be aligned to either the high or low half-word to equalize capacitive loading on the bus. A hardware reset loads this control field from the boot source selection (bootmod[1:0]) to the SDRAM1 control register.	00 = 16-bit aligned to D[31:16] 01 = 16-bit aligned to D[15:0] 1x = 32-bit memory
SREFR Bits 15–14	**SDRAM Refresh Rate**—Enables/Disables SDRAM refresh cycles and controls the refresh rate. Refresh cycles are referenced to a 32 kHz clock. At each rising edge, 1, 2, or 4 rows will be refreshed. Multiple refresh cycles are separated by the row cycle delay specified in the SRC control field.	See Table 22-7 on page 22-13 for bit and field settings.

Table 22-6. SDRAM 0 Control Register and SDRAM 1 Control Register Description (continued)

Name	Description	Settings
CLKST Bits 13–12	**Clock Suspend Time-out**—Determines if and when the SDRAM will be placed in a clock suspend condition. The suspend time-out can be triggered either on the absence of an active bank also called Precharge Powerdown (CLKST = 01) or on a clock count from the last access also called Active Powerdown (CLKST = 10 or 11). Count-based time-outs do not force the SDRAM into an idle condition (for example, any active banks remain open). Section 22.6.4, “Clock Suspend Low-Power Mode,” provides a comprehensive description of this operating mode.	00 = Disabled 01 = Anytime all banks are inactive 10 = 64 clocks after completion of last access (Precharge Powerdown). 11 = 128 clocks after completion of last access (Active Powerdown).
Reserved Bits 11–10	Reserved—These bits are reserved and should read 0	
SCL Bits 9–8	**SDRAM CAS Latency**—Determines the latency between a read command and the availability of data on the bus. This field does not affect the second and subsequent data words in a burst, and has no effect on write cycles.	00 = Reserved 01 = 1 clock 10 = 2 clocks 11 = 3 clocks **Note:** See Figure 22-3 on page 22-14 CAS Latency Timing diagram.
Reserved Bit 7	Reserved—This bit is reserved and should read 0.	
SRP Bit 6	**SDRAM Row Precharge Delay**—Determines the number of idle clocks inserted between a precharge command and the next-row activate command to the same bank.	0 = 3 clocks inserted 1 = 2 clocks inserted
SRCD Bits 5–4	**SDRAM Row-to-Column Delay**—Determines the number of clocks inserted between a row activate command and a subsequent read or write command to the same bank. See Figure 22-5.	00 = 4 clocks inserted 01 = 1 clock inserted 10 = 2 clocks inserted 11 = 3 clocks inserted
Reserved Bit 3	Reserved—This bit is reserved and should read 0.	
SRC Bits 2–0	**SDRAM Row Cycle Delay**—Determines the minimum delay (in number of clocks) between a refresh and any subsequent refresh or read/write access. This delay corresponds to the minimum row cycle time captured in the tRC/tRFC memory timing spec. An example timing diagram for SRC = 3 can be found in Figure 22-6. **Note:** The SRC control field is not used to enforce t_{RC} timing for row activate to row activate within the same bank. The sum of t_{RCD} + CAS latency + t_{RP} (reads) and t_{RCD} + t_{WR} + t_{RP} (writes) must be greater than t_{RC} (SRC).	000 = 8 clocks 001 = 1 clock 010 = 2 clocks 011 = 3 clocks 100 = 4 clocks 101 = 5 clocks 110 = 6 clocks 111 = 7 clocks

1. Placing the MC9328MXL SDRAM controller in Bank Interleaved mode is not the same as SDRAM Interleaved mode. For Bank Interleaved mode, the SDRAM memory must be programmed to sequential or linear mode in the SDRAM mode register.

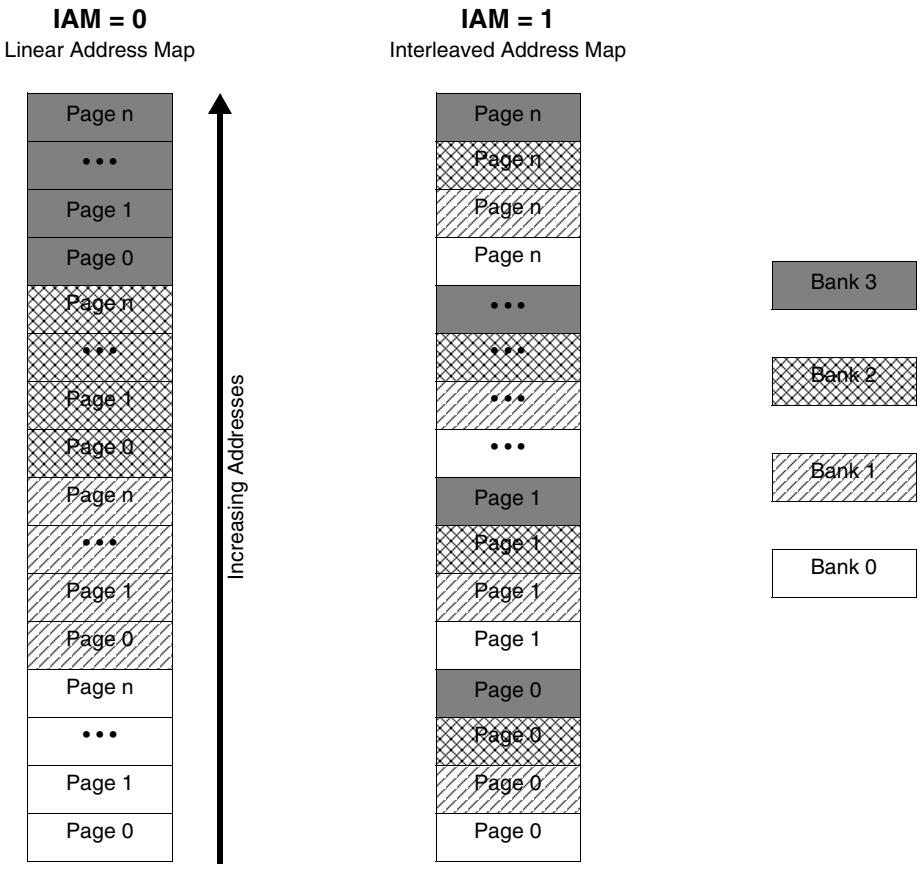


Figure 22-2. Memory Bank Interleaving Options

Table 22-7. Settings for SREFR Field

SREFR	Rows Each Refresh Clock	~Row/64ms @ 32 kHz	Row Rate @ 32 kHz	~Row/64ms @ 32.768 kHz	Row Rate @ 32.768 kHz
00	Refresh disabled				
01	1	2048	31.25 µs	2097	30.52 µs
10	2	4096	15.62 µs	4194	15.26 µs
11	4	8192	7.81 µs	8388	7.63 µs

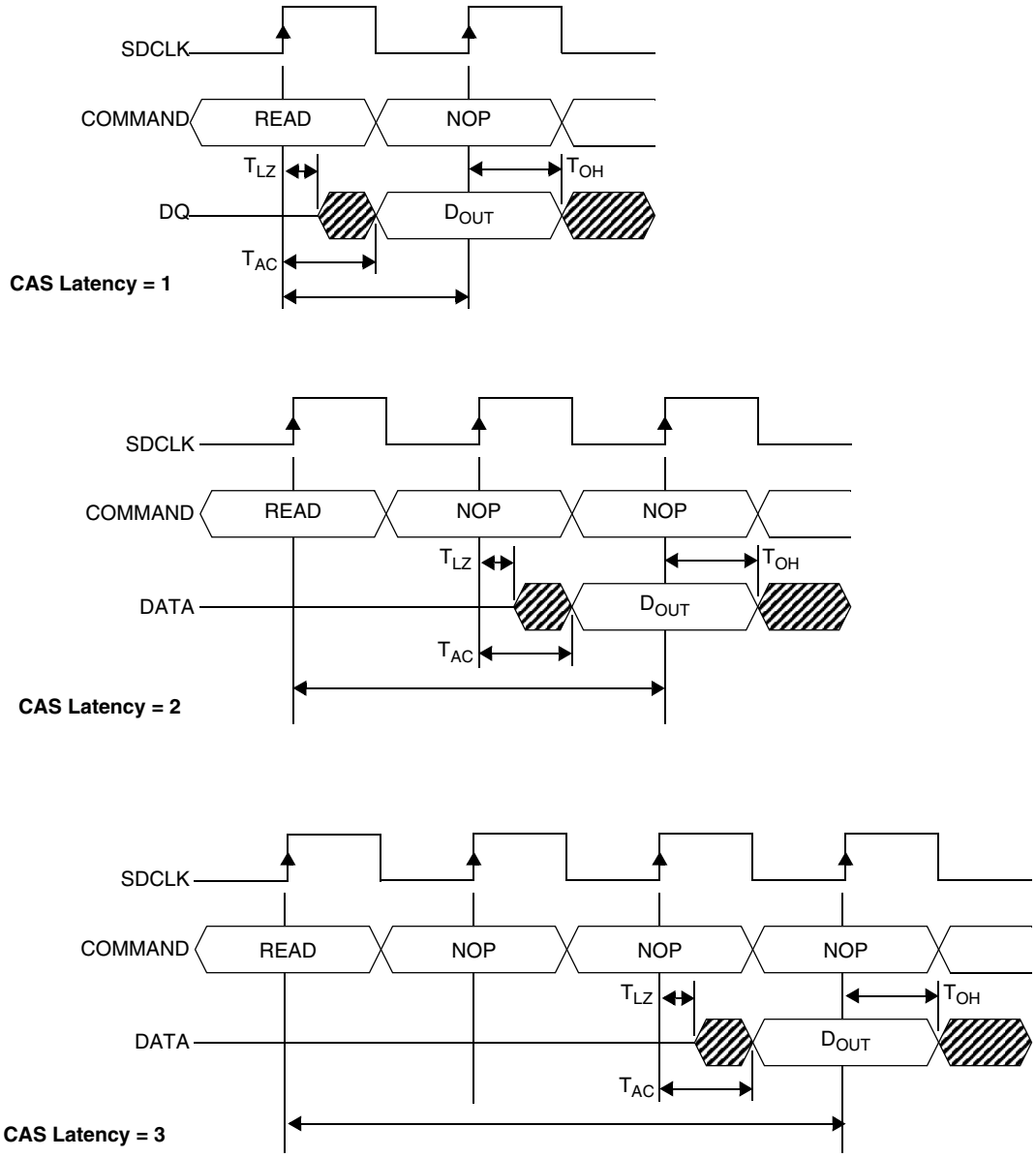


Figure 22-3. CAS Latency Timing

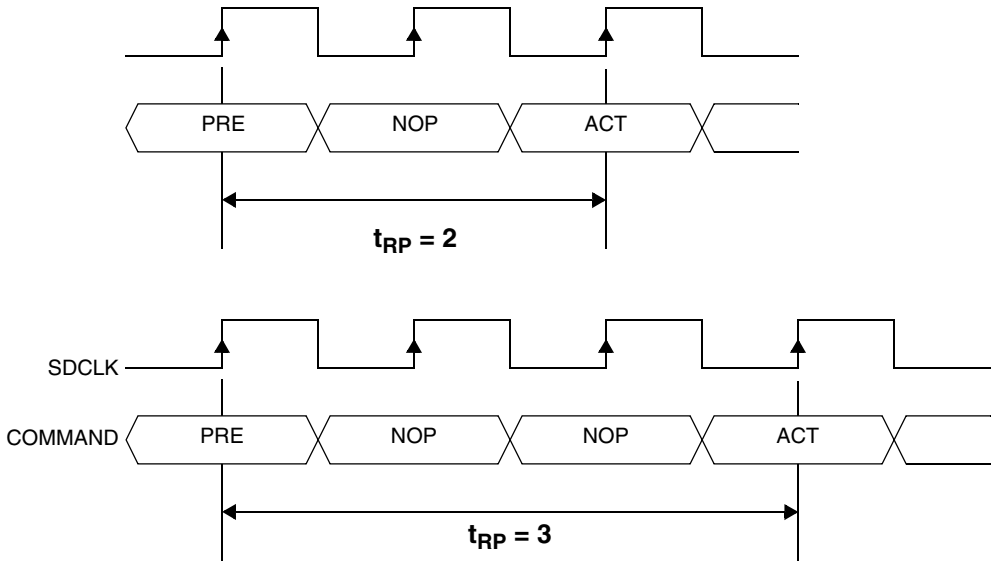


Figure 22-4. Precharge Delay Timing

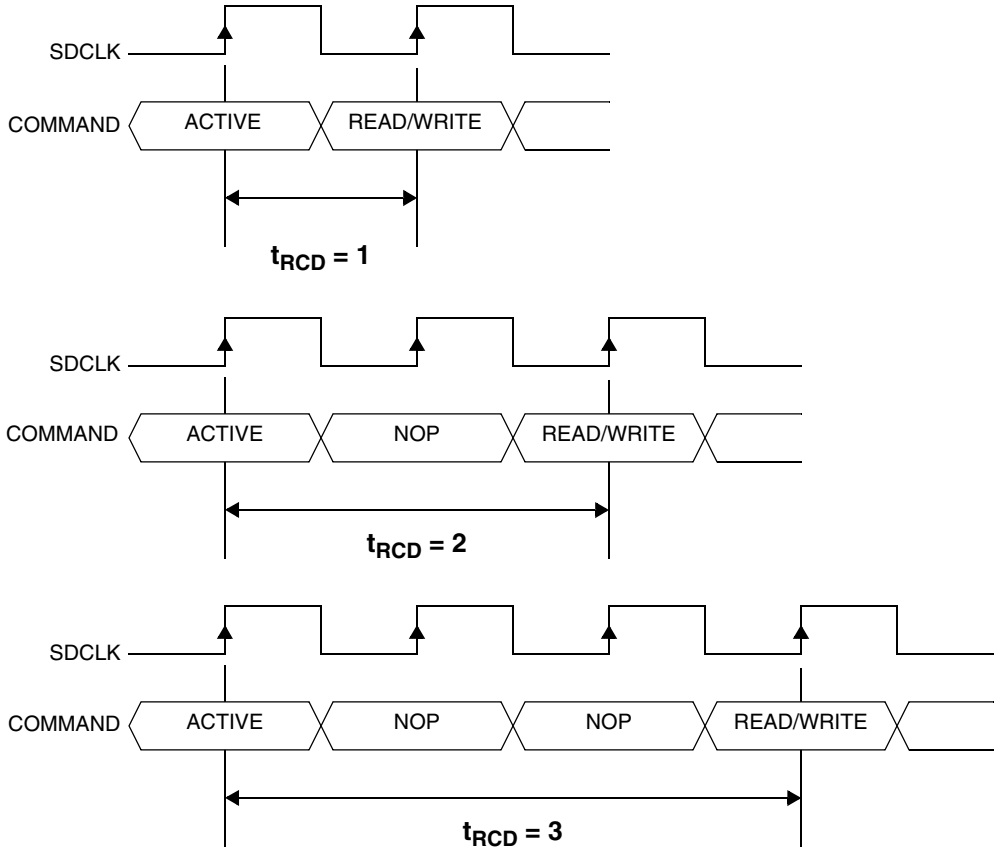


Figure 22-5. Row-to-Column Delay Timing

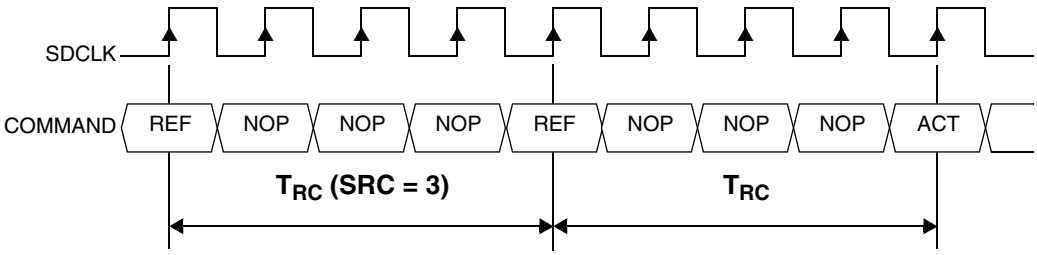


Figure 22-6. Row Cycle Timing

22.4.2 SDRAM Reset Register

The write-only SDRAM Reset Register controls the reset pulse timing.

SDRST SDRAM Reset Register **Addr 0x00221018**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RST															
TYPE	w	w	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 22-8. SDRAM Reset Register Description

Name	Description	Settings
RST Bits 31–30	**Software Initiated Local Module Reset Bits**—Generates local module reset to SDRAM/SyncFlash controller.	00 = No effect to the SDRAMC 01 = One HCLK cycle reset pulse 10 = One HCLK cycle reset pulse 11 = Two HCLK cycle reset pulse
Reserved Bits 29–0	Reserved—These bits are reserved and should read 0.	

22.4.3 Miscellaneous Register

MISCELLANEOUS Miscellaneous Register **Addr 0x00221014**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	OMA															
TYPE	rw	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																RMA0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 22-9. Miscellaneous Register Description

Name	Description	Settings
OMA Bit 31	**Multiplexed Address Override**—Enables/Disables the MA0 pin to be a meaningful bit. The multiplexed address, original MA0, is always zero at read access in 16-bit port memory configuration. This introduces difficulty with the SyncFlash read-device configuration which requires that the address ranges from 0 to 3. Working with bit 0 in tandem, this overrides the original MA0 generated by the internal address multiplexer during the 16-bit SyncFlash read-device configuration command.	0 = Address from internal address multiplexed is routed out to MA0 1 = Force RMA0, bit 0, out to MA0 pin
Reserved Bits 30–1	Reserved—These bits are reserved and should read 0.	
RMA0 Bit 0	**MA0 Replacement**—Contains value of MA0 when OMA is set.	

Code Example 22-1 is a programming example for SyncFlash read-device configurations.

Code Example 22-1. Read-Device

```
HWSF_read_device_ID
            ldr    r2, 0x80000001            //force ma0 to 1
            ldr    r8, 0x00221014
            str    r2,(r8)

            ldr    r8, 0x0c000000            //read SyncFlash device ID
            ldrh   r2, (r8,0)
            ldr    r1, 0x00d3
            cmp    r1, r2
            bt     ERROR_OUT

            ldr    r2, 0x00000000            //release ma0
            ldr    r8, 0x00221014
            str    r2,(r8)
```

22.5 Operating Modes

Each of the SDRAM Controller operating modes is described in this section, including details on basic operation, relationship to SDRAM/SyncFlash operating modes, and any special precautions to observe. State and timing diagrams are included where appropriate.

22.5.1 SDRAM and SyncFlash Command Encoding

Table 22-10 summarizes the command encodings used by the SDRAM controller. This command list is a subset of the commands defined by the JEDEC standard. Note that the SDRAM Auto, Self-Refresh, and SyncFlash Load Command register commands share the same encoding. Also note, that encodings are based from the view of the SDRAM memory to the controller, and therefore use SDRAM signal names. See Figure 22-46 on page 22-51 for an example.

Table 22-10. SDRAM and SyncFlash Command Encoding

Function	Symbol	CKEn-1	CKEn	$\overline{CS}$	$\overline{RAS}$	$\overline{CAS}$	$\overline{WE}$	A11	A10	BA[1:0]	A[9:0]
Deselect	DSEL	H	X	H	X	X	X	X	X	X	X
No operation	NOP	H	X	L	H	H	H	X	X	X	X
Read	READ	H	X	L	H	L	H	V	L	V	V
Write	WRIT	H	X	L	H	L	L	V	L	V	V
Bank activate	ACT	H	X	L	L	H	H	V	V	V	V
Burst terminate	TBST	H	X	L	H	H	L	X	X	V	X
Precharge select bank	PRE	H	X	L	L	H	L	X	L	V	X
Precharge all banks	PALL	H	X	L	L	H	L	X	H	X	X
Auto-refresh	CBR	H	X	L	L	L	H	X	X	X	X
SyncFlash load command register	LCR	H	X	L	L	L	H	X	X	V	V
Self refresh entry	SLFRSH	H	L	L	L	L	H	X	X	X	X
Self refresh exit	SLFRSHX	L	H	H	X	X	X	X	X	X	X
Power-down entry	PWRDN	H	L	X	X	X	X	X	X	X	X
Power-down exit	PWRDNX	L	H	H	X	X	X	X	X	X	X
Mode register set	MRS	H	X	L	L	L	L	L	L	V	V

Assertion of the sd_rst signal initializes the controller into the idle state. As long as the SDRAM Controller is not the boot source, that assertion also disables the module. While disabled, the controller remains in the idle state with the internal clocks stopped.

If the SDRAM Controller has been selected as the boot device, then the module is enabled following reset. The reset state of the control register allows for basic read/write operations sufficient to fetch the reset vector and execute the initialization code. A complete initialization of the controller must be performed as part of the start-up code sequence.

Read/write cycles, refresh and low-power mode requests, and clock suspend time-outs will all trigger transitions out of the idle state. State transitions due to a read or write request depend on the operating mode. Other transitions require the corresponding function to be enabled in the SDCTL register. Some state transitions have been removed to minimize complexity and allow an easier understanding of the basic controller operation.

The following subsections document the operation of each of the operating modes.

22.5.2 Normal Read/Write Mode (SMODE = 000)

The Normal Read/Write mode (SMODE = 000) is used for general read and write access to the SDRAM controller, and for reads of the SyncFlash. SyncFlash writes use the SyncFlash Program mode (See Section 22.5.7, “SyncFlash Program Mode.”) Both single and burst accesses are supported, although burst requests are limited to a length of 8 words (one word = 32 bits).

Read or write requests to the SDRAM Controller initiate a check to see whether the page is already open. This check consists of comparing the request address against the last row accessed within the corresponding bank. If the rows are different, it indicates that a precharge has occurred after the last access, or there has never been an access to the bank. In that case, the access must follow the “off-page” sequence. If the requested row and last row match, the shorter “on-page” access is used.

An off-page sequence must first activate the requested row, an operation which is analogous to a conventional DRAM RAS cycle. An activate cycle is the first operation depicted in Figure 22-7. During the activate cycle, the appropriate chip-select is driven low, the row addresses are placed on the multiplexed address pins, the non-multiplexed addresses are driven to their respective values, write enable is driven high, $\overline{CAS}$ is driven high, and $\overline{RAS}$ is driven low. These latter three pins form the SDRAM command word. The data bus is unused during the activate command.

When the selected row has been activated, the read operation begins after the row-to-column delay (t_{RCD}) has been met. This delay is either 2 or 3 clocks, as determined by the SRCD field of the appropriate control register. During the read cycle, the chip-select is once again asserted, the column addresses are driven onto the multiplexed address bus, the non-multiplexed addresses remain driven to the value presented during the activate cycle, the write enable is driven high, $\overline{RAS}$ is driven high, and $\overline{CAS}$ is driven low. After the CAS latency has expired, data is transferred across the data bus. CAS latency is programmable via the SCL field of the control register. As data is being returned across the AHB, transfer acknowledge is asserted back to the CPU indicating that the CPU must latch data. While data is still on the bus, the SDRAM Controller must begin monitoring transfer requests because the CPU is free to issue the next bus request on the same edge that data is being latched.

Data transfers can be either single operand or a burst of up to 8 operands. Burst requests are designated as such by the $\overline{P_BURST}$ attribute. It is activated by the load multiple command from the ARM920T core when the data or instruction cache is enabled. This AHB signal is asserted low for all except the last operand of a burst transfer. Non-burst transfers do not assert the signal.

SDRAM memories assume that all transfers are burst transfers unless terminated early. Burst transfers can be terminated by a variety of mechanisms: another read or write cycle, a precharge operation, or a burst terminate command. Burst terminate commands are the general mechanism used by the SDRAM controller for early burst termination. The burst terminate command is subject to the CAS latency and must be pipelined similar to the Read command, as shown in Figure 25-7 and Figure 25-8. When a load-multiple command is executed, the SDRAM controller will not issue a burst terminate command.

SDRAM write cycles are different than read cycles in one important aspect. Whereas read data was delayed by the CAS latency, write data has no delay and is supplied at the same time as the Write command. Figure 25-13 illustrates an off-page write cycle followed by an on-page write cycle. Note that the write data is driven during the same clock that the Write command is issued. The SDRAM controller only supports single burst writes and does not issue a burst terminate after each write. Therefore, the user must make sure to program the SDRAM memory’s

mode control register for single burst writes for proper operation. The SDRAM controller does, however, have the capability to issue “single-clock-cycle” writes. This is different than the traditional burst writes, where the WRITE command and column address are presented on the bus during the first write data, and the SDRAM memory internally increments its address. In the case of the MC9328MXL, each data to be written to the SDRAM is accompanied with both the WRITE command and the associated column address that is being written to. This still achieves the same bandwidth that the burst write would. However, to take advantage of this feature, the data cache of the ARM920T core must be enabled and the MMU set up with a page table such that the desired region of the SDRAM memory map is cacheable. Also, the SDRAM controller does not issue a burst terminate command after the end of a series of burst write, it simply discontinues the WRITE command.

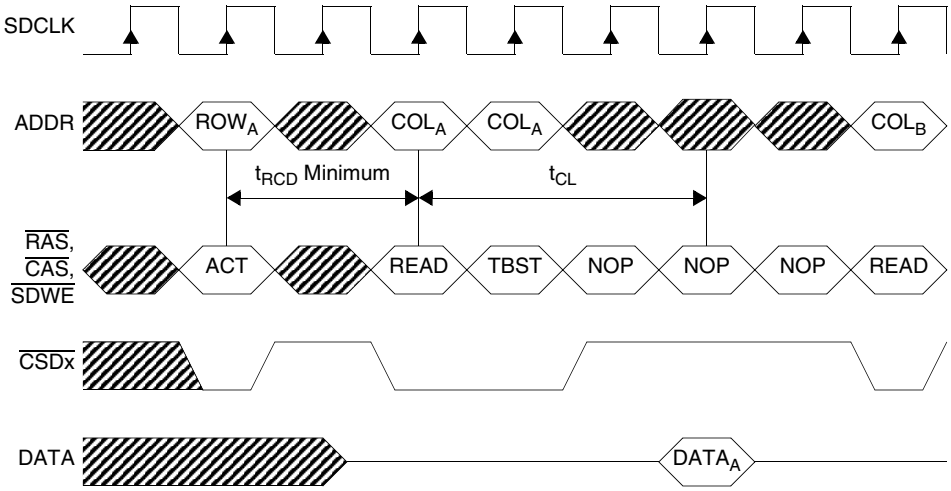


Figure 22-7. Off-Page Single Read Timing Diagram (32-Bit Memory)

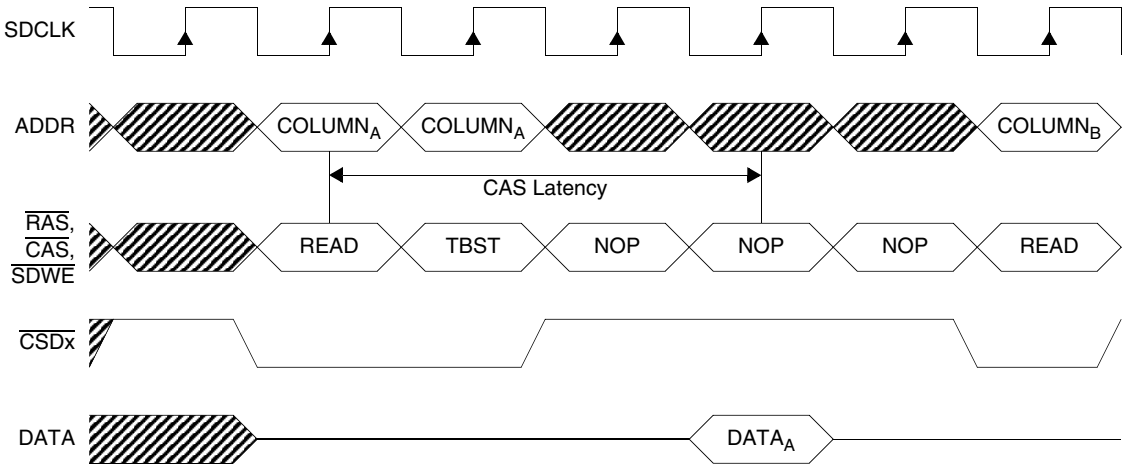


Figure 22-8. On-Page Single Read Timing Diagram (32-Bit Memory)

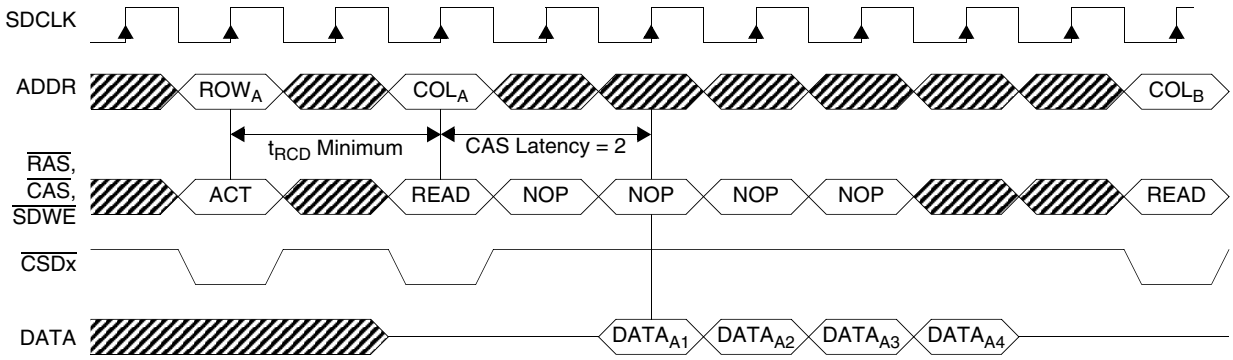


Figure 22-9. Off-Page Burst Read Timing Diagram (32-Bit Memory)

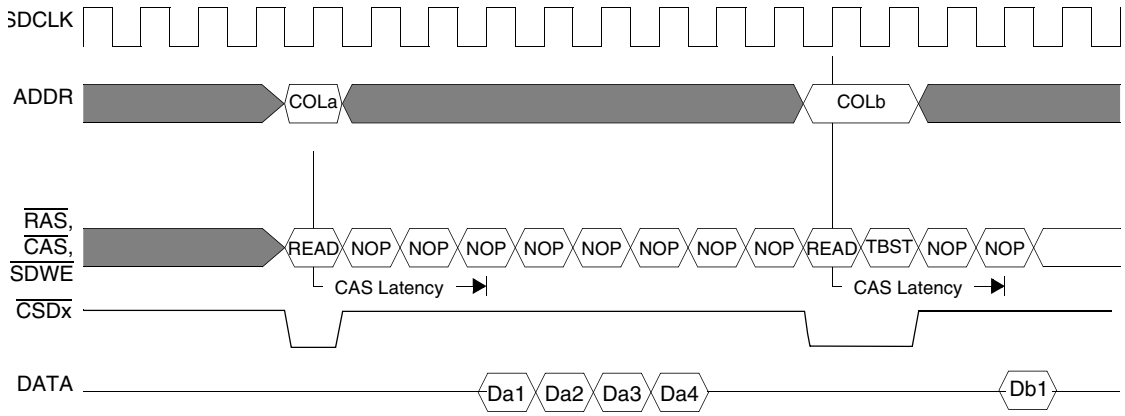


Figure 22-10. On-Page Burst Read Timing Diagram (32-Bit Memory)

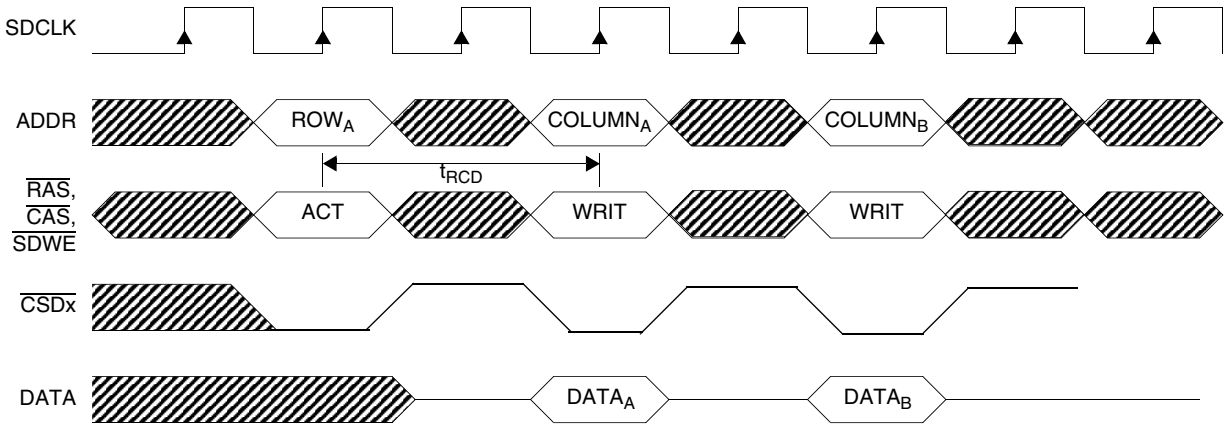


Figure 22-11. Off-Page Write Followed by On-Page Write Timing Diagram

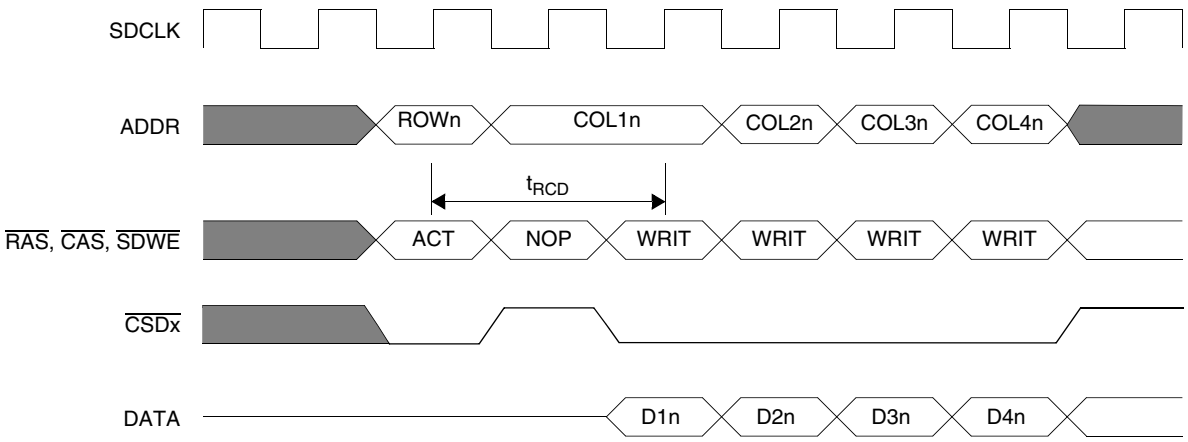


Figure 22-12. Off-Page Burst Write Timing Diagram

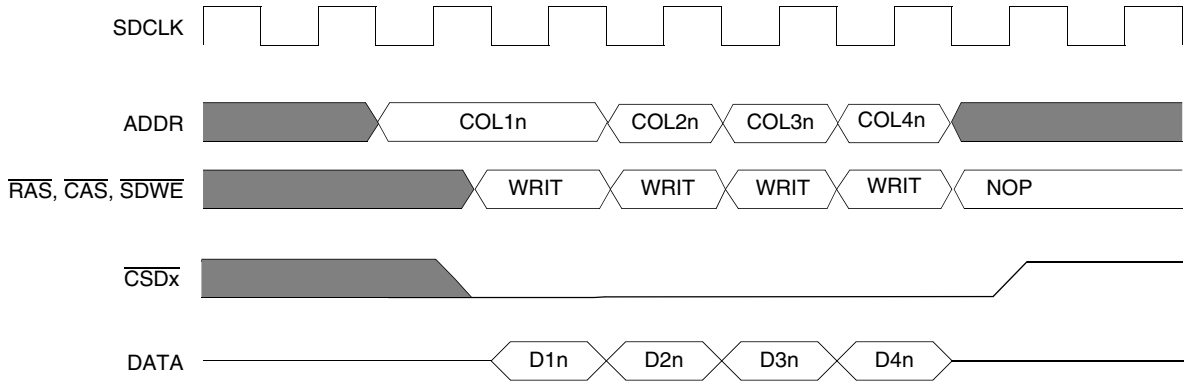


Figure 22-13. On-Page Burst Write Timing Diagram

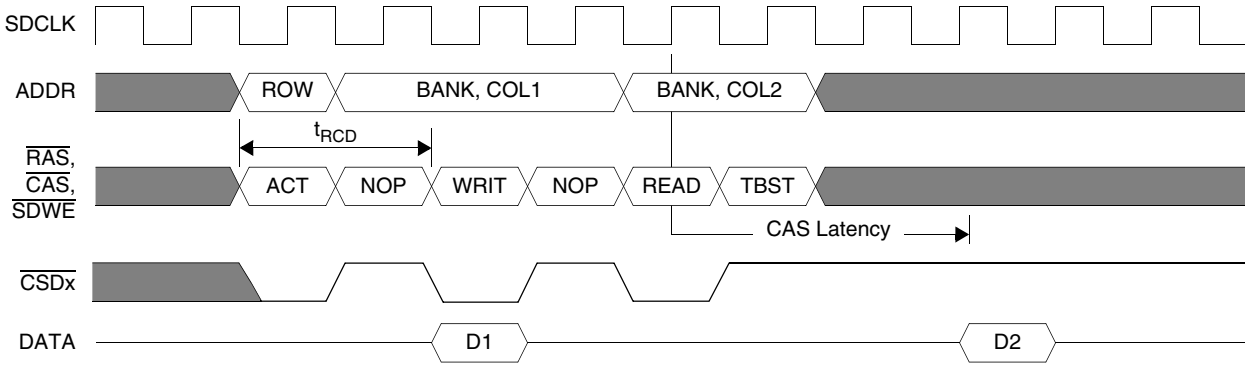


Figure 22-14. Single Write Followed by On-Page Read Timing Diagram

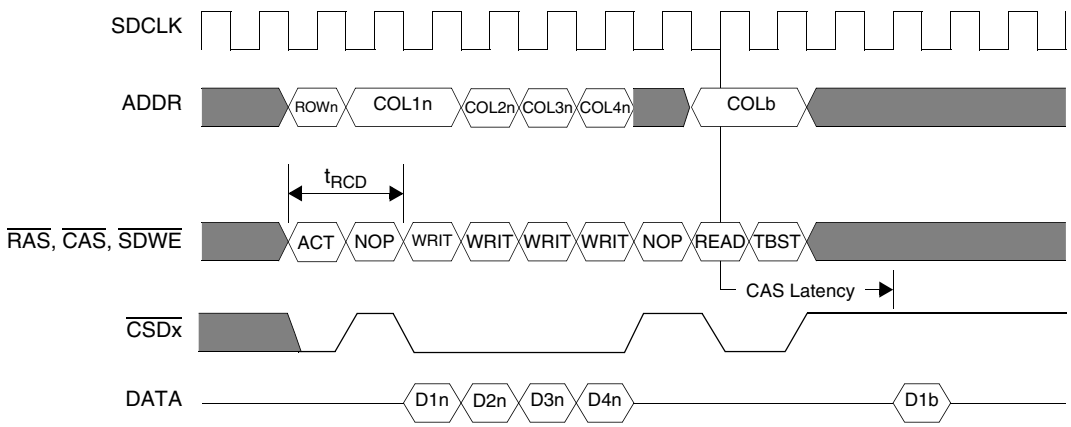


Figure 22-15. Burst Write Followed by On-Page Read Timing Diagram

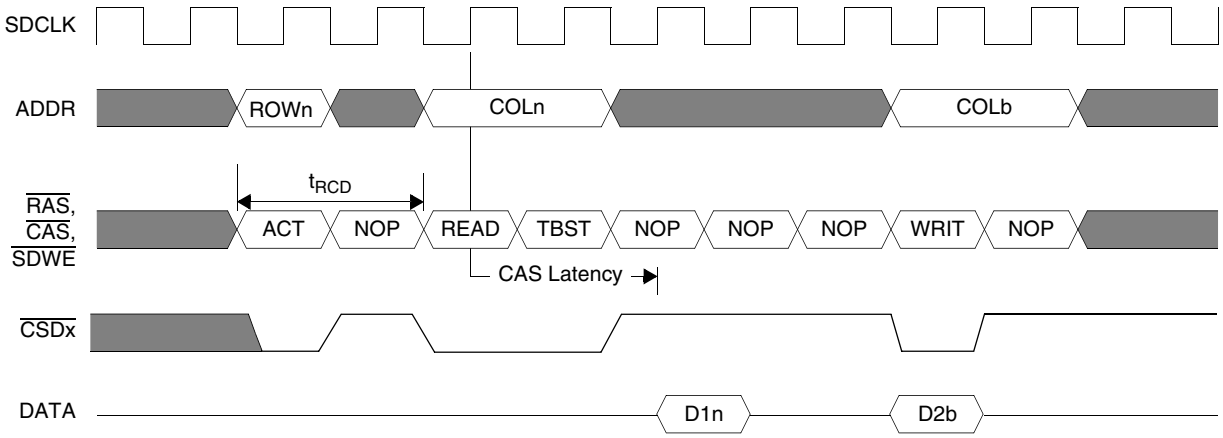


Figure 22-16. Single Read Followed by On-Page Write Timing Diagram

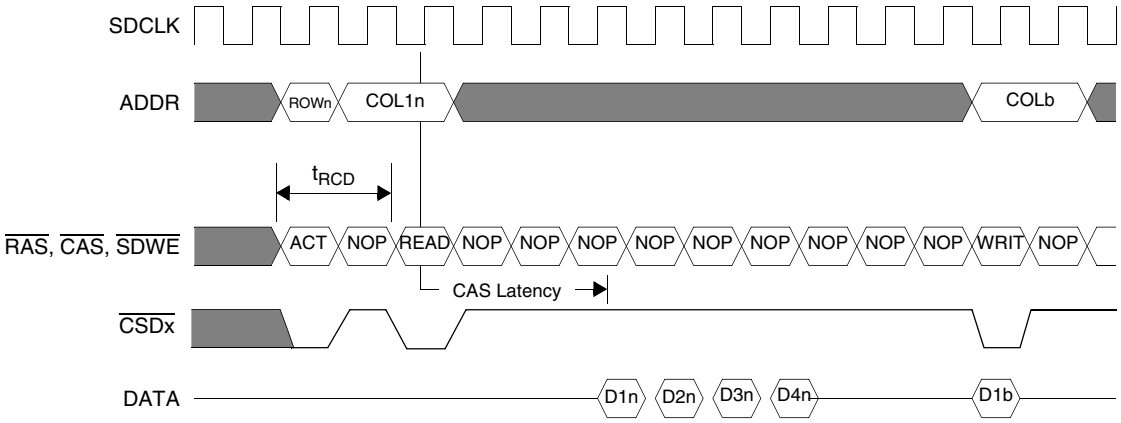


Figure 22-17. Burst Read Followed by On-Page Write Timing Diagram

22.5.3 Precharge Command Mode (SMODE = 001)

The Precharge Command Mode (SMODE = 001) is used during SDRAM device initialization, and to manually deactivate any and all active banks. While in this mode, an access (either read or write) to the SDRAM address space will generate a precharge command cycle. SDRAM address bit A10 determines whether a single bank, or all banks, are precharged by the command. (See Figure 22-18). Accessing an address with the SDRAM address A10 low will precharge only the bank selected by the bank address. Conversely, accesses with A10 high will precharge all banks regardless of the bank address. Note that A10 is the SDRAM pin, not the ARM920T processor's address. Translation of the SDRAM A10 address bit to the corresponding ARM920T processor's address is dependent on the memory configuration. The precharge command access is two clocks in length on the AHB, and one cycle to the SDRAM controller.

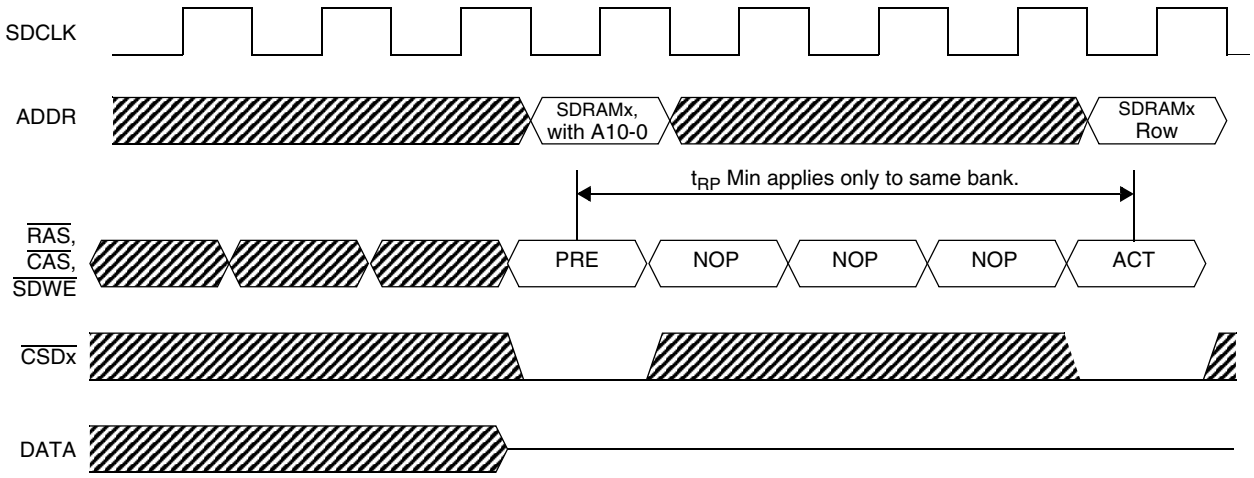


Figure 22-18. Precharge Bank Timing Diagram

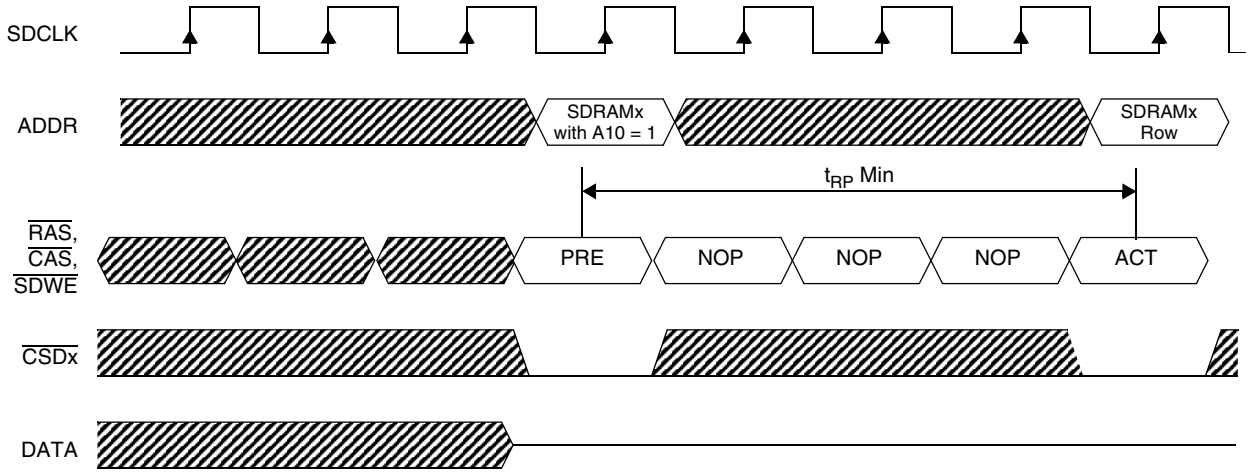


Figure 22-19. Precharge All Timing Diagram

22.5.4 Auto-Refresh Mode (SMODE = 010)

The Auto-Refresh Mode (SMODE = 010) is used to manually request SDRAM refresh cycles. It is normally used only during device initialization because the SDRAM Controller will automatically generate refresh cycles when properly configured. The auto-refresh command refreshes all banks in the device, so the address supplied during the refresh command only needs to specify the correct SDRAM device. The lower address lines are ignored. Either a read or write cycle may be used. If a write is used, the data will be ignored and the external data bus will not be driven. The cycle will be 2 clocks on the AHB and a single clock to the SDRAM device.

The SDRAM Controller guarantees that the SDRAM is in the idle state before the auto-refresh command is given. If one or more rows are active, a precharge-all command will be issued prior to the auto-refresh command. The precharge-all command adds one additional clock to the access time.

Figure 22-20 on page 22-25 provides the software initiated Auto-Refresh timing diagram.

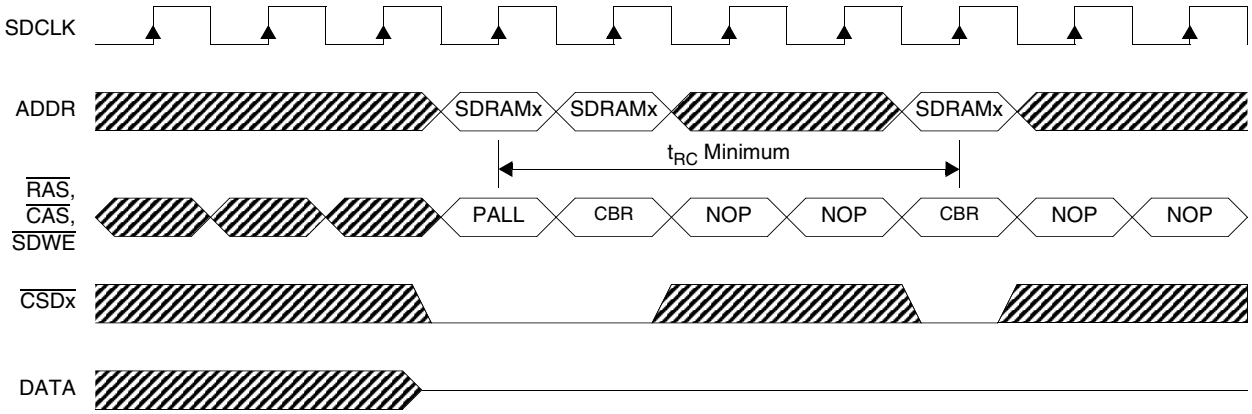


Figure 22-20. Software Initiated Auto-Refresh Timing Diagram

NOTE:

SDRAM devices require a minimum delay of t_{RC} between refresh cycles. The SDRAM Controller incorporates a timer to guarantee this timing is met. The timer is user configurable through the SRC field in the SDCTLx register.

22.5.5 Set Mode Register Mode (SMODE = 011)

The Set Mode Register mode (SMODE = 011) is used to program the SDRAM/SyncFlash mode register. This mode differs from normal SDRAM write cycles because the data to be written is transferred across the address bus. Reads of the mode register are not allowed.

Either a read or write cycle may be used for this transfer. In the case of a write, the AHB data will be ignored and the external data bus will not be driven. The row and bank address signals are used to transfer the data. The cycle will be 2 clocks on the AHB and a single clock to the SDRAM device.

Figure 22-21 and Figure 22-22 illustrate the bus sequence for a mode register set operation. Mode register set commands must be issued while the SDRAM is idle.

The SDRAM Controller does not guarantee that the SDRAMs have returned to the idle state before issuing the mode register set command. Therefore, software must generate a precharge all sequence before issuing the mode register set command if there is any possibility that one or more banks could be active. Also keep in mind that the row cycle time (t_{RC}) must be met before the mode register set command is issued.

Section 22.7.4, “Mode Register Programming,” provides a detailed example of the mode register data value calculation and mapping to the ARM920T processor’s address.

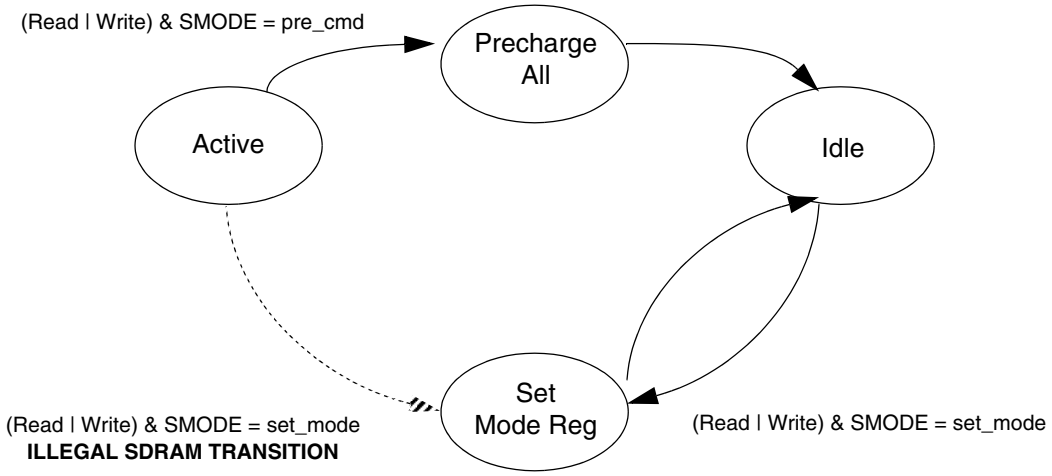


Figure 22-21. Set Mode Register State Diagram

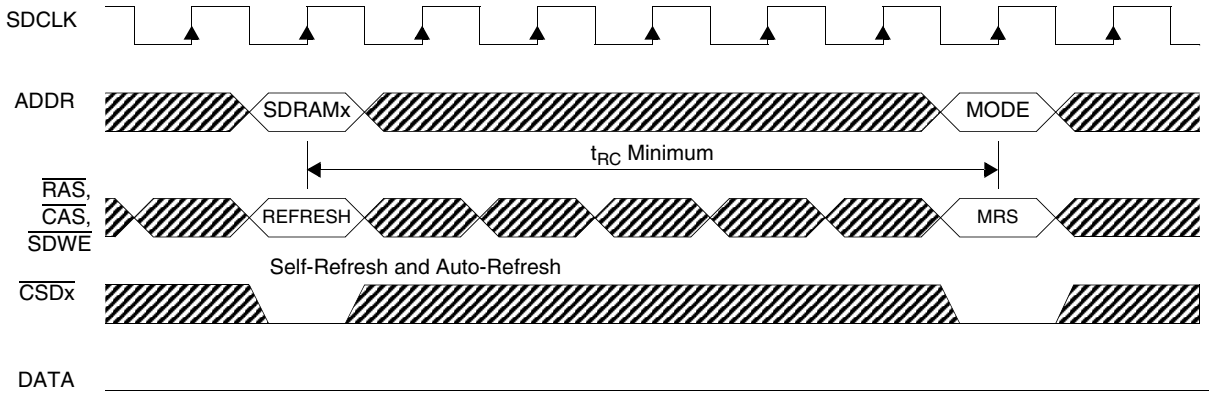


Figure 22-22. Set Mode Register Timing Diagram

22.5.6 SyncFlash Load Command Mode

SyncFlash memories extend the standard SDRAM command set to include device-specific program, erase, and configure commands. These proprietary operations use a 3 command sequence: Load Command register, Activate, and Read or Write. The Load Command Register serves as an “escape” code to provide special meaning to the read or write commands which follow. The 3 command sequences must always occur in the correct order, however any number of NOPs can be inserted between them. The Load Command Register command is mapped to the same encoding as the SDRAM Auto-Refresh and Self-Refresh commands.

The SyncFlash Load Command mode provides the mechanism for software to generate the first command in the sequence. By toggling between this mode and Normal Read/Write any of the command triplets can be generated.

An example of a configuration register read is shown in Figure 22-23. The first bus cycle places the SDRAM Controller in the Load Command Register Mode. The following bus cycle begins the first command of the triplet by loading the command register with the desired operation code. Because the operation code is transferred across the address bus, either a read or write bus cycle could have been used for this cycle. The third bus cycle returns the

SDRAM Controller to the normal read/write mode. Finally, the fourth cycle performs the actual read of the configuration register. Although not shown in the diagram, the device configuration data would be returned across the data bus after the CAS latency had been met. The activate and read comprise the final two commands in the triplet and must refer to the same bank as the original command register load. Changes to the bank address within the triplet will return indeterminate results. Details on command register encodings and operation are covered in Section 22.8.4, “SyncFlash Configuration.”

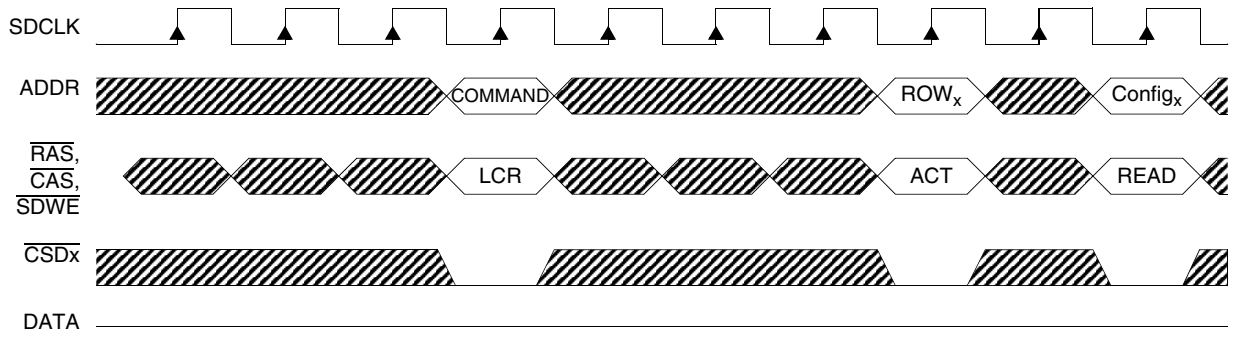


Figure 22-23. Load Command Register Timing Diagram

22.5.7 SyncFlash Program Mode

SyncFlash programming and status checking operations require the same 3 command sequence described in Section 22.5.6. Because these operations are repeated many thousands of times to program the entire array, a hardware programming mode is provided as an alternative to the software-intensive method previously described.

SyncFlash programming mode implements two command sequence triplets: a program sequence (write) and a status check sequence (read). The state diagram for these two sequences is illustrated in Figure 22-24.

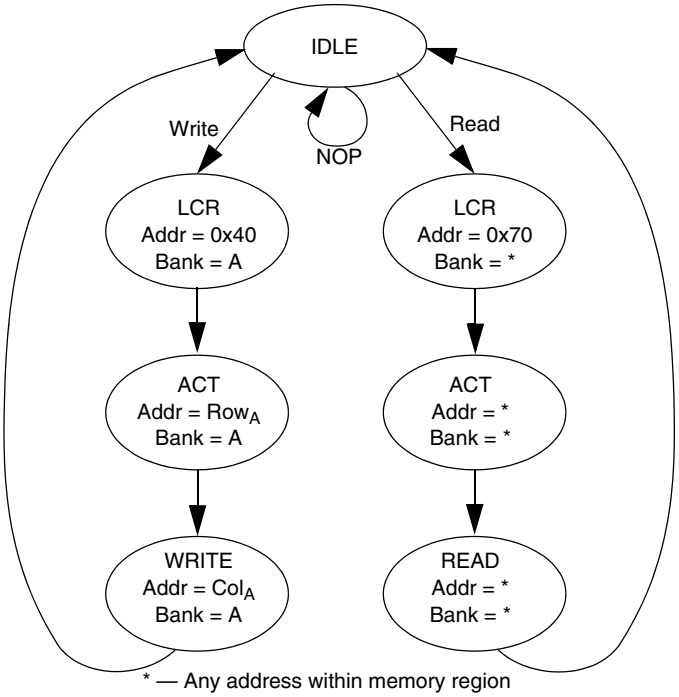


Figure 22-24. SyncFlash Program Mode State Diagram

A write to an address within the memory region initiates the program sequence. The first command issued to the SyncFlash is Load Command register. A [7:0] determines which operation the command performs. For this write setup operation, an address of 0x40 is hardware generated. The bank and other address lines are driven with the address to be programmed. The second command, Active, registers the row address and confirms the bank address. The third command supplies the column address, re-confirms the bank address, and supplies the data to be written. SyncFlash does not support burst writes, therefore a Burst Terminate command is not required.

A read to the memory region initiates the status read sequence. The first command issued to the SyncFlash is the Load Command Register with A [7:0] set to 0x70 which corresponds to the Read Status Register operation. The bank and other address lines are driven to the selected address. The second command, Active, sets up the status register read. The bank and row addresses are driven during this command. The third command of the triplet is Read. Bank and column addresses are driven on the address bus during this command. Data is returned from memory on the low-order 8 data bits following the CAS latency.

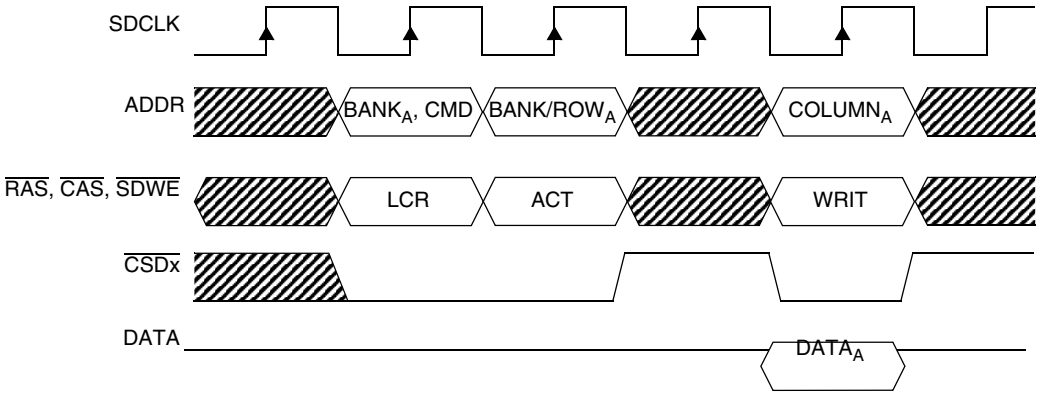


Figure 22-25. SyncFlash Program Timing Diagram

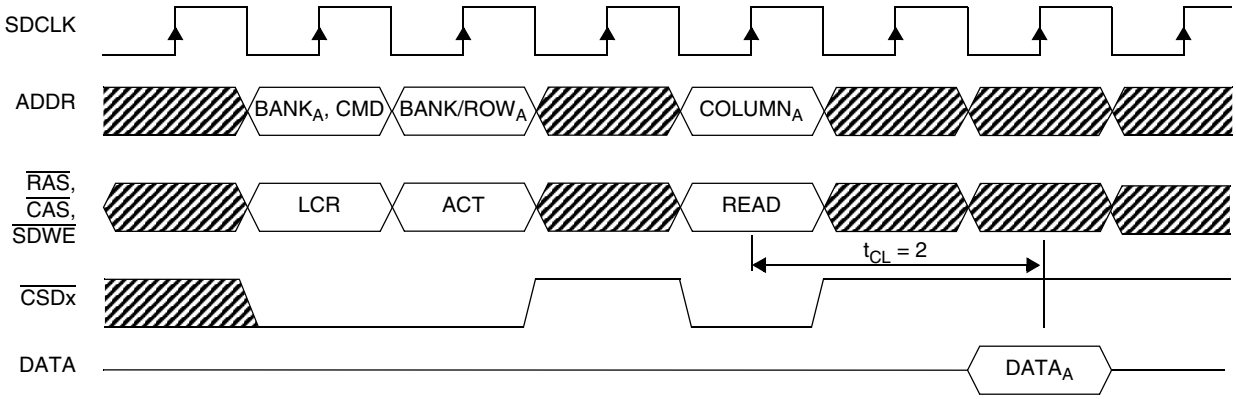


Figure 22-26. SyncFlash Read Status Register Timing Diagram

22.6 General Operation

The general operation of the SDRAM controller is discussed in this section and includes address multiplexing, refresh and self-refresh modes. The SDRAM Controller is designed to support a broad range of JEDEC standard SDRAM configurations including devices of 64-, 128-, and 256-Mbit densities. Given the physical size constraints of the target applications, the design was optimized for a memory device data width of 32 bits. Table 22-11

summarizes the devices targeted by the design. The controller is capable of interfacing with devices of other widths and densities, however, only devices with 4 banks are supported. A 100 MHz system bus operation is possible with PC100 compliant single data rate memory devices.

Table 22-11. JEDEC Standard Single Data Rate SDRAMs

	SDRAM Configuration					
Item	**64 Mbit**		**128 Mbit**		**256 Mbit**	
Bus Width	16	32	16	32	16	32
Depth	4 Mword	2 Mword	8 Mword	4 Mword	16 Mword	8 Mword
Refresh Rows	4096 (15.62 μs)	4096 (15.62 μs)	4096 (15.62 μs)	4096 (15.62 μs)	8192 (7.81 μs)	8192 (7.81 μs)
# Banks	4	4	4	4	4	4
Bank Address	2	2	2	2	2	2
Row Address	12	11	12	12	13	13
Column Address	8	8	9	8	9	8
Data Qualifiers	2	4	2	4	2	4

22.6.1 Address Multiplexing

The JEDEC standard SDRAMs for which the controller was optimized, use an asymmetrical array architecture with more row than column address lines. The SDRAM Controller multiplexes only those pins which change between the row and column addresses. The remaining (most significant) row addresses and the bank addresses are not multiplexed.

22.6.1.1 Multiplexed Address Bus

The SDRAM Controller multiplexed address bus is aligned to the column addresses so that address line A1 always appears on pin MA0. With this alignment, the “folding point” in the multiplexor is driven solely by the number of column address bits, although interleave mode causes a two bit shift to account for the bank addresses. Column bus widths of 8 to 11 bits are supported in non-interleave mode, although only 8 and 9 bit widths are allowed in interleave mode. Table 22-12 summarizes the multiplex options supported by the controller. Column addresses through A10 are driven regardless of the multiplexor configuration, although some of the lines will be unused for the smaller page sizes.

Memory width does not affect the multiplexer; however, it does affect how the memories are connected to the SDRAM Controller pins. The width of the multiplexed bus is one bit larger than in previous generations of the SDRAMC so that 32-bit memory systems can be supported with a minimal impact on the multiplex hardware because 32-bit memory systems are shifted left by one bit and use MA [n+1:1]. This is demonstrated in the last two rows of Table 22-12 by the grayed-out boxes. Note that the AP signal is duplicated in two bit positions to permit this signal to always appear on memory pin A10. Table 22-13 lists the SDRAM interface connections for different configurations of JEDEC SDRAM.

Table 22-12. Address Multiplexing by Column Width

Column Bits		Memory Width	SDRAM Controller Pin											
IAM=0	IAM=1		MA11	MA10	MA9	MA8	MA7	MA6	MA5	MA4	MA3	MA2	MA1	MA0
			ROW											
8	-	32	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9
9	-	32	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10
10	8	32	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11
11	9	32	A23	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12
			COLUMN											
ALL		32	AP	AP	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1

Indicates address lines not required for this memory width.

Table 22-13. MC9328MXL to SDRAM Interface Connections

Memory Configuration	**4M x 16 Bits x 2 Chips (16 Mbyte)**		**8M x 16 Bits x 2 Chips (32 Mbyte)**		**16M x 16 Bits x 2 Chips (64 Mbyte)**		**2M x 32 Bits x 1 Chip (8 Mbyte)**		**4M x 32 Bits x 1 Chip (16 Mbyte)**		**8M x 32 Bits x 1 Chip (32 Mbyte)**	
i.MX Pins	**SDRAM Memory Address Pins**											
Rows	12		12		13		11		12		13	
Columns	8		9		9		8		8		8	
Data size	32		32		32		32		32		32	
Refresh rows	4096		4096		8192		2048		4096		8192	
IAM	0	1	0	1	0	1	0	1	0	1	0	1
i.MX Pins	**SDRAM Memory Address Pins[1]**											
A19				BA1		BA1						
A18		BA1		BA0		BA0		BA1		BA1		BA1
A17		BA0						BA0		BA0		BA0
A16												
A15					BA1	A12						
A14			BA1	A11	BA0	A11					BA1	A12
A13	BA1	A11	BA0		A12				BA1	A11	BA0	A11
A12	BA0		A11		A11		BA1		BA0		A12	
A11	A11						BA0		A11		A11	

Table 22-13. MC9328MXL to SDRAM Interface Connections (continued)

Memory Configuration	4M x 16 Bits x 2 Chips (16 Mbyte)		8M x 16 Bits x 2 Chips (32 Mbyte)		16M x 16 Bits x 2 Chips (64 Mbyte)		2M x 32 Bits x 1 Chip (8 Mbyte)		4M x 32 Bits x 1 Chip (16 Mbyte)		8M x 32 Bits x 1 Chip (32 Mbyte)	
i.MX Pins	**SDRAM Memory Address Pins**											
MA11	A10	A10	A10	A10	A10	A10	A10	A10	A10	A10	A10	A10
MA10	A9	A9	A9	A9	A9	A9	A9	A9	A9	A9	A9	A9
A10	A8	A8	A8	A8	A8	A8	A8	A8	A8	A8	A8	A8
A9	A7	A7	A7	A7	A7	A7	A7	A7	A7	A7	A7	A7
A8	A6	A6	A6	A6	A6	A6	A6	A6	A6	A6	A6	A6
A7	A5	A5	A5	A5	A5	A5	A5	A5	A5	A5	A5	A5
A6	A4	A4	A4	A4	A4	A4	A4	A4	A4	A4	A4	A4
A5	A3	A3	A3	A3	A3	A3	A3	A3	A3	A3	A3	A3
A4	A2	A2	A2	A2	A2	A2	A2	A2	A2	A2	A2	A2
A3	A1	A1	A1	A1	A1	A1	A1	A1	A1	A1	A1	A1
A2	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0

1. Grayed-out boxes are unused.

22.6.1.2 Non-Multiplexed Address Bus

The most significant row address bits are not sampled when the column addresses are being driven, and therefore do not need to be multiplexed. The SDRAM Controller implementation takes advantage of this fact and uses the existing non-multiplexed address bus to provide these signals. Addresses A21 through A25 are needed across the supported configurations. The specific connections which will need to be made are dependent on the memory device type, density and bank interleave mode. These configuration-dependent connections are handled through the design of the external hardware. Examples are provided in Section 22.7, “SDRAM Operation,” and Section 22.8, “SyncFlash Operation.”

22.6.1.3 Bank Addresses

Which bank addresses of the SDRAM controller are multiplexed with which MC9328MXL address pins, depends on whether or not the memory system is in interleaved mode. For non-interleaved mode, the SDRAM controller bank addresses SDBA[4:0] are multiplexed with the address pins A[15:11]. For interleaved mode, the SDRAM controller “interleaved” bank addresses SDIBA[3:0] are multiplexed with the MC9328MXL address pins A[19:16]. Instead of detailing the complexity of how these bank addresses are multiplexed with the corresponding MC9328MXL address pins with different memory configurations, the user is directed to Table 22-13. This table explicitly shows which SDRAM memory bank address pin must be connected to which corresponding MC9328MXL address pins given different the JEDEC standard memory configurations. Also, if the user wants to derive how the density of the memory is calculated or how to derive the page size, they use the following equations:

$$\text{Page Size (Bytes)} = 2^{\#\text{Column Address Bits}} \times (\text{Memory Width in Bits} / 8) \qquad \textit{Eqn. 22-1}$$

$$\text{Density (Bytes)} = 2^{(\#\ \text{Column Address Bits} + \#\ \text{Row Address Bits})} \times (\text{Memory Width in Bits} / 2) \qquad \textit{Eqn. 22-2}$$

22.6.2 Refresh

SDRAM Controller hardware satisfies all SDRAM refresh requirements after an initial configuration by the user software. 0, 1, 2, or 4 refresh cycles are scheduled at 31.25 µS (nominal 32 kHz clock) intervals, providing 0, 2048, 4096, or 8192 refresh cycles every 64 ms. The refresh rate is programmed through the SREFR field in the SDCTLx registers. Each array can have a different rate, allowing a mix of different density SDRAMs. Refresh is disabled by hardware reset.

A refresh request is made pending at each rising edge on the 32 kHz clock. In response to this request, the hardware gains control of the SDRAM as soon as any in-process bus cycle completes. Once it has gained control of the memory, commands are issued to precharge all banks. Following a row precharge delay (t_{RP}), an auto-refresh command is issued. At t_{RC} intervals, additional auto-refresh cycles are issued until the specified number of cycles have been run. Figure 22-27 illustrates a 2 refresh sequence.

Burst transfers in progress when the refresh request arrives are allowed to complete prior to the refresh operation. SDRAM bus accesses queued after the refresh request are held off until the refresh completes. In Figure 22-28, an access is queued just as the refresh begins. This cycle is delayed until the precharge and single refresh (SREFR = 01) cycles are run. Bus cycles targeted to other memory or peripheral devices are allowed to progress normally while the refresh is in progress. None of the pins shared between the SDRAM and other devices are required for the refresh operation.

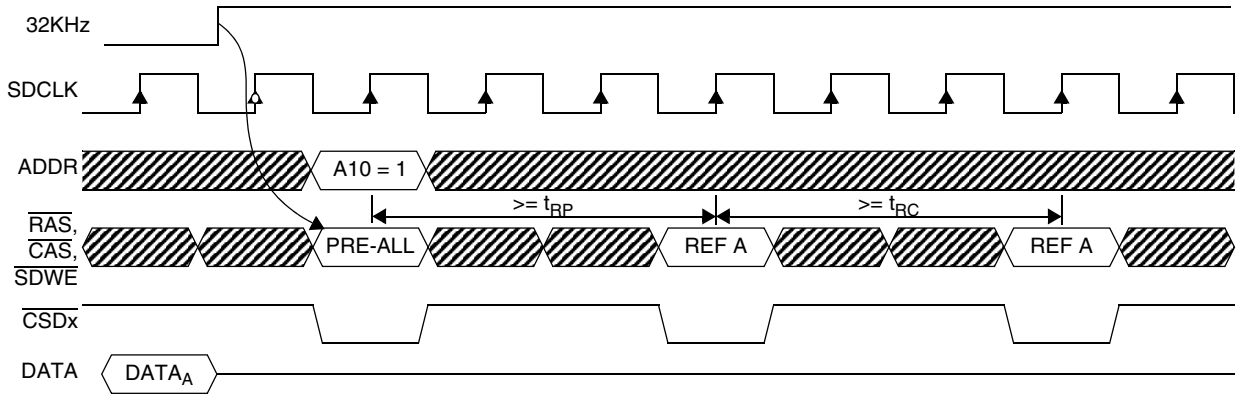


Figure 22-27. Hardware Refresh Timing Diagram

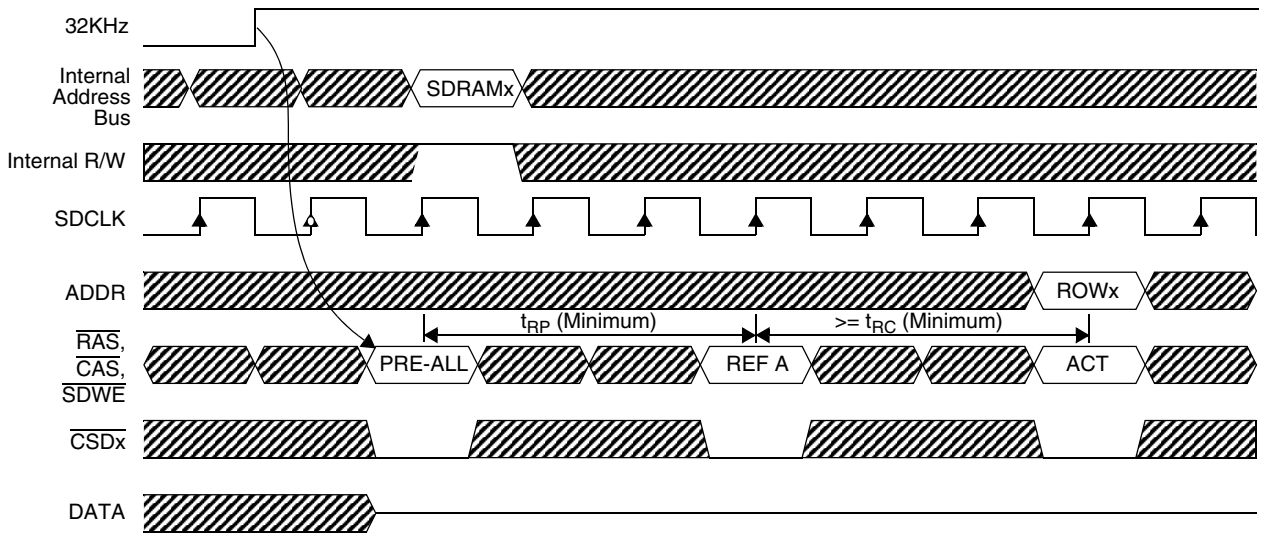


Figure 22-28. Hardware Refresh With Pending Bus Cycle Timing Diagram

22.6.3 Self-Refresh

SDRAM data must be retained during assertion of system reset ($\overline{\text{RESET_IN}}$) and power conservation modes if refresh has been enabled. The SDRAM Controller detects these conditions and places the memory in self-refresh. If refresh has not been enabled, the SDRAM Controller places the memories in a lower power consumption mode known as Powerdown. This operation is described in Section 22.6.3.3, “Powerdown Operation During Reset and Low-Power Modes.”

22.6.3.1 Self-Refresh During $\overline{\text{RESET_IN}}$

The assertion of system reset ($\overline{\text{RESET_IN}}$) triggers the SDRAM Controller to place the memory in self-refresh provided refresh had been previously enabled. Refresh during system reset is disabled by an SDRAM Controller reset. It remains disabled until the refresh rate is programmed to a non-zero value. Once enabled, self-refresh is invoked anytime system reset is asserted without a corresponding SDRAM reset.

22.6.3.2 Self-Refresh During Low-Power Mode

If refresh is enabled, low-power mode also forces the SDRAM into self-refresh mode. When the SDRAM Controller detects that the bus masters are entering a low-power condition it begins a self-refresh sequence once any in-progress bus access has completed. A Precharge All command is issued to close any open memory pages, the Self-Refresh command is issued, and the clock enable is brought low. Once the memories are safely in their low-power state, the SDRAMC tells the system clock controller to enter sleep mode.

22.6.3.3 Powerdown Operation During Reset and Low-Power Modes

The powerdown mode is used instead of self-refresh whenever system reset or any of the low-power modes occur and refresh has not been enabled. This memory operating mode does not remove power, as the name might imply. It simply lowers power consumption by disabling the clock input buffer and halting all internal activity. Because powerdown can only be entered if all banks are idle, a Precharge All command must be issued to the memories prior to stopping the clock. Figure 22-31 illustrates the powerdown sequence following assertion of system reset.

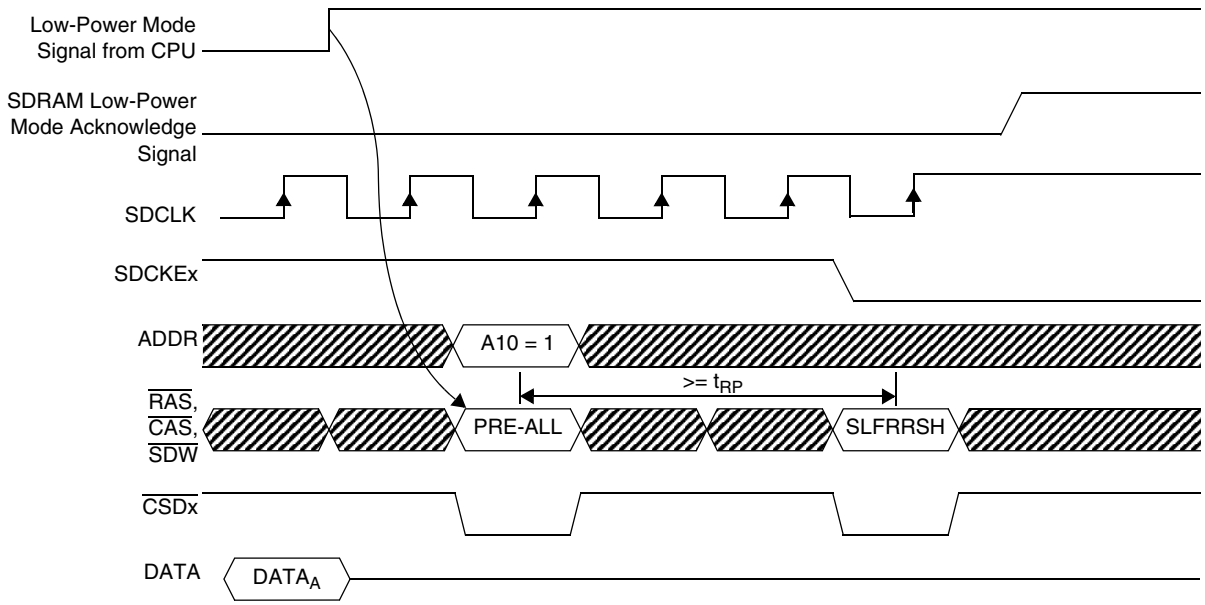


Figure 22-29. Self-Refresh Entry Due to Low-Power Mode Timing Diagram

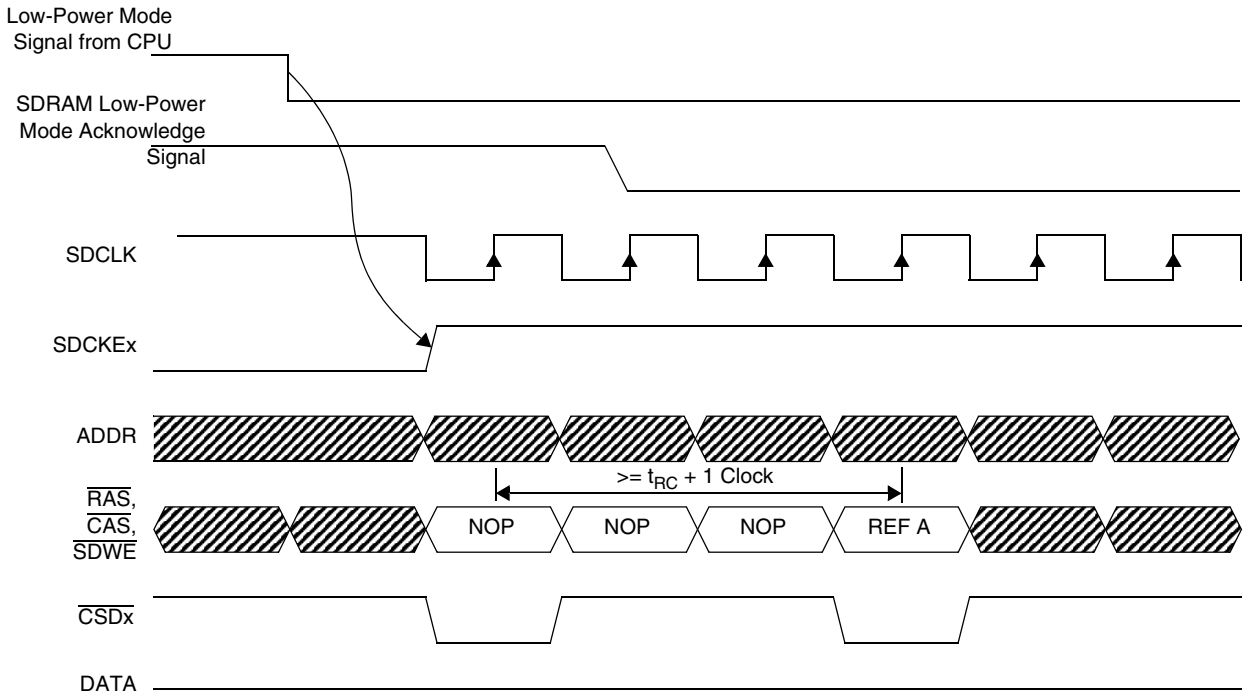


Figure 22-30. Low-Power Mode Self-Refresh Exit Timing Diagram

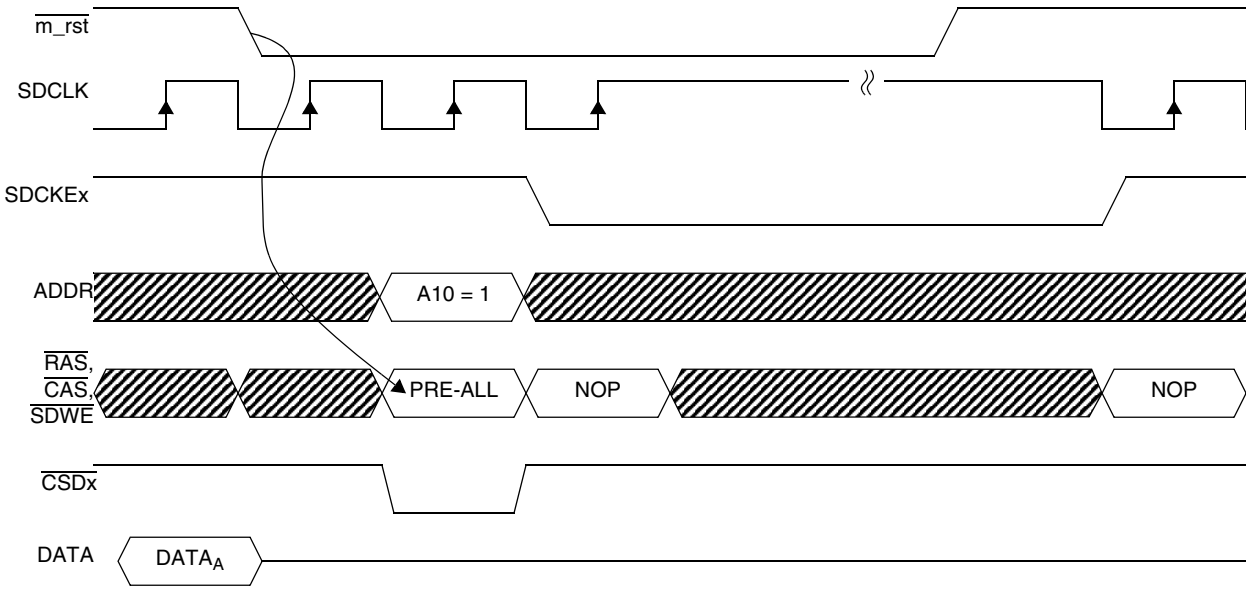


Figure 22-31. Powerdown Mode Resulting From Reset with Refresh Disabled

22.6.4 Clock Suspend Low-Power Mode

SDRAM and SyncFlash memories incorporate a command sequence to disable the clock input buffer and suspend internal activity, lowering power consumption as much as an order of magnitude. The SDRAM Controller implements a clock suspend time-out mechanism to take advantage of this feature. Several software selectable time-outs are provided to accommodate for varying system conditions, with the time-out condition being specified in the Clock Suspend Timeout (CLKST) field in the SDCTLx register. The feature is disabled out of reset.

SDRAM/SyncFlash clock suspend actually consists of two sub-modes: Clock Suspend and Powerdown. The distinguishing factor between the two is whether banks remain active while the clock is stopped. Clock suspend allows banks to remain activated, while Powerdown does not.

22.6.4.1 Powerdown (Precharge Powerdown)

Programming CLKST [1:0] = 01 causes the SDRAM Controller to place the memories in powerdown mode anytime the controller detects that no banks are active. The Precharge Powerdown mode is useful in applications where a memory array is accessed infrequently and the chances of another access to the same page are minimal.

Reading or writing to memory activates a page within the addressed bank. Reset, software generated precharge, and hardware initiated refresh are three ways to close an active bank. The periodically occurring refresh will be the normal means that invokes the powerdown mode. At each refresh interval, all banks will be closed by a precharge-all command, followed by the refresh operation. The controller will then issue the powerdown command to the memories. A few cycle delays are incurred with the first read or write cycle to restart the clocks, however only on the first cycle. After that, the clocks will continue to run until the next refresh operation or until any active banks are manually precharged.

Page misses on read and write cycles cause the addressed bank to be closed (precharged) and a new page opened within the bank. This operation does not cause the clocks to stop, nor does manually precharging only a single bank within the memory. All banks within the memory must be inactive before the powerdown mode is invoked.

22.6.4.2 Clock Suspend (Active Powerdown)

The second clock suspend mode is selected whenever CLKST [1:0] = 1x. In Active Powerdown mode the clocks are stopped after a selectable delay from the last access to the array. Active banks are not closed prior to disabling the SDRAM clock. Either 64 (CLKST [1:0] = 10) or 128 (CLKST [1:0] = 11) cycle delays are possible. SDRAM clocks are counted from the end of the last read or write access. Subsequent read or write accesses, and self-refresh modes reset the counter. Auto-refresh cycles do not affect the counter; however, if the counter expires during a refresh operation the clock will be disabled immediately following the refresh.

22.6.4.3 Refresh During Powerdown or Clock Suspend

Refresh requests queued while the clock is suspended will restart the clock, run the appropriate number of refresh cycles, and then disable the clock again.

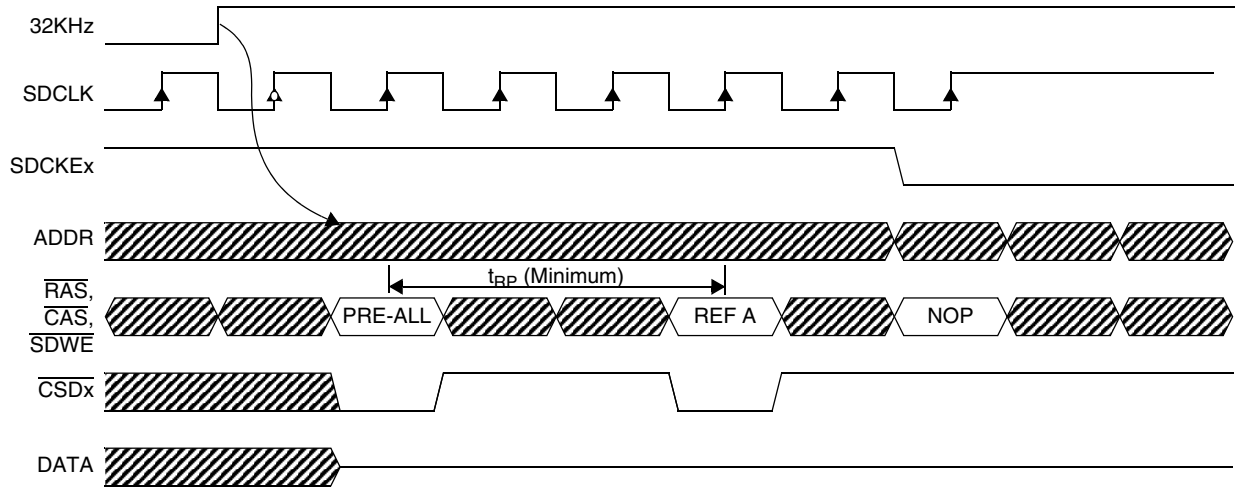


Figure 22-32. Precharge Powerdown Mode Entry Timing Diagram

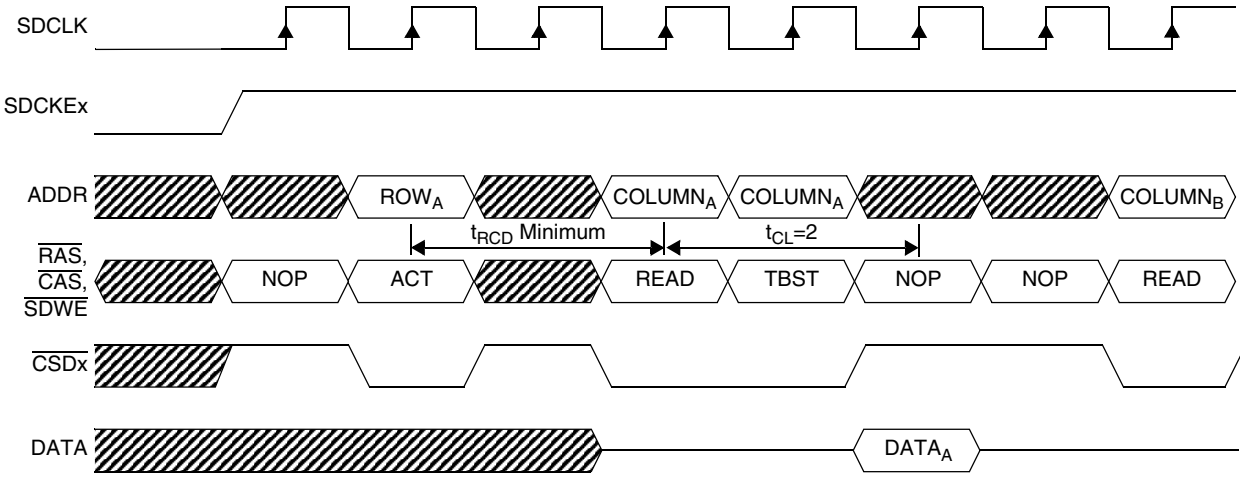


Figure 22-33. Precharge Powerdown Exit Timing Diagram

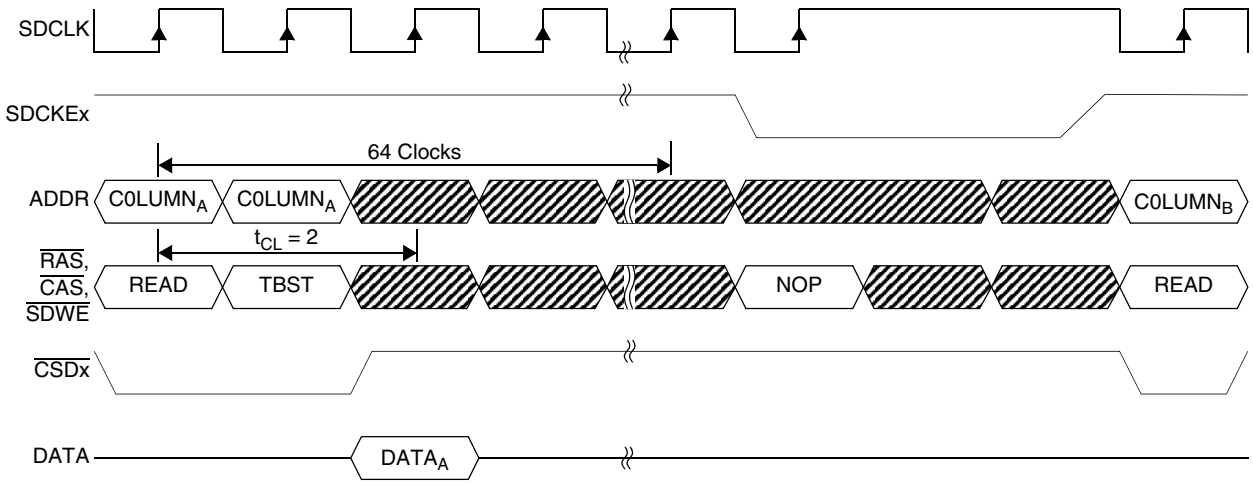


Figure 22-34. Clock Suspend (Active Powerdown) Timing Diagram

NOTE:

For each of the previous Precharge and Active Powerdown figures, it is assumed that only one chip-select has been enabled. If both chip-selects are enabled and one of the two chip-selects is not currently in a powerdown mode, then the SDCLK will continue to run even while the other chip-select is in powerdown mode since both chip selects share the SDCLK.

22.7 SDRAM Operation

The following subsections provide details on selecting compatible SDRAM memories and configuring the controller to work with the memory system.

22.7.1 SDRAM Selection

Table 22-11 on page 22-29 lists the memories around which the controller is optimized, however many other memory types are also suitable for use. Important characteristics to consider when choosing a memory module are:

- Page comparators expect 4 bank memories—2 bank devices are not supported. Their use results in the memory and controller losing synchronization when crossing page boundaries.
- Page (column) addressing must match one of the supported sizes.
- Memory density can be larger or smaller than those directly supported, although some memory may be inaccessible or redundantly mapped. In linear mode (IAM bit of the Control Register is 0), the bank addresses are the most significant addresses and connecting a memory smaller than the selected density will result in one or more banks being inaccessible.
- Controller design is for memories meeting PC100 timing specifications at 100 MHz system operation. Use of non-compliant memories or other system clock rates require a thorough analysis of all timing parameters.

22.7.2 Configuring Controller for SDRAM Memory Array

Configuration register programming options and controller-memory physical connections provide flexibility to accommodate different memory types and system configurations. Options are broadly grouped into 3 categories:

- Physical Characteristics: row and column address bus widths, data bus width, and interleave mode
- Timing Parametrics: CAS latency, row precharge and cycle delays, and refresh rate
- Functional Features: clock suspend timer and supervisor/user protection

Table 22-20 on page 22-46 and Table 22-21 on page 22-47 are provided to assist the designer with the selection of the correct physical parameters for a number of preferred memory configurations. Timing parametrics are addressed in the following subsections.

22.7.2.1 CAS Latency

CAS latency is determined by the operating frequency and access time of the memories. For a 100 MHz system clock frequency and PC100 compliant memories, the CAS latency will generally be programmed to 3 clocks, although the memory specifications must always be consulted to confirm this value. CAS latency must be programmed in two places: the chip-select Control Register and the SDRAM Mode Register. See Table 22-6 on page 22-9 for a description of the control register encoding and Section 22.7.4, “Mode Register Programming,” for the details on programming the SDRAM mode register.

22.7.2.2 Row Precharge Delay

Row precharge delay is defined as the delay between a precharge command and the next row activate command to the same bank. The SDRAM memory specification provides the minimum precharge delay as t_{RP}, usually in terms of ns. Therefore the user will have to calculate the number clocks needed for the row precharge delay and program this value into the SRP bit in SDRAM Control register described in Table 22-6 on page 22-9. For example, a t_{RP} specification of 15 ns with a 100 MHz clock (10 ns period) results in 1.5 clocks, rounded to 2.

22.7.2.3 Row-to-Column Delay

The row-to-column delay is defined as the delay between a row activate command and a subsequent read or write command. The SDRAM memory specification provides the minimum row-to-column (or ACTIVE to READ or WRITE) delay as t_{RCD}, usually in terms of ns. Given the system bus speed, the user must calculate the number of clocks needed after an activate command is given before the subsequent read or write command and program this value into the SRCD bit field in the SDRAM Control register, as described in Table 22-6 on page 22-9. For example, a t_{RCD} specification of 15 ns with a 100 MHz clock (10 ns period) results in 1.5 clocks, rounded to 2.

22.7.2.4 Row Cycle Delay

Row cycle delay is defined as the delay between a refresh and any subsequent refresh or read/write access. Because the t_{RC} timing for row activate to row activate with the same bank is implicitly guaranteed by the sum of $t_{RCD} + t_{CL} + t_{RP}$, the value programmed into the SRC field in the SDRAM Control register must depend on the AUTO REFRESH period, t_{RCF}, specified in the SDRAM memory.

22.7.2.5 Refresh Rate

The refresh rate is the rate by which the SDRAM controller is required to refresh each row in the SDRAM memory. The SDRAM memory specification provides the total number of rows per bank, and the corresponding refresh rate must be programmed into the SREFR field in the SDRAM control register (See Table 22-6 on page 22-9). For example, if the SDRAM memory contains 8192 rows, or requires 8192 AUTO REFRESH cycles every 64 ms, then the refresh rate can be calculated as 64 ms / 8192 rows = 7.81 µs per row. Therefore the user programs the SREFR field with 11. Refer to Section 22.6.2, “Refresh,” on page 22-32 for more information.

22.7.2.6 Memory Configuration Examples

Twelve different examples of SDRAM configurations are provided in this section and shown in Figure 22-35 on page 22-40 through Figure 22-56 on page 22-70. These examples are 64 Mbit, 128 Mbit, and 256 Mbit SDRAM memories in single x16, dual x16, and single x32 configurations. All examples use bank and non-bank interleaving for address configuration. Each figure is accompanied with the control register values in Table 22-14 on page 22-40 through Table 22-49 on page 22-70.

NOTE:

The following examples assume that the number of rows and columns for each given SDRAM density follow the JEDEC standard. The SDRAM Controller can interface to SDRAMS which do not follow the JEDEC standards for row and column sizes, however the user must take care to ensure that the SDRAM Control Register bits ROW and COL are programmed to the appropriate number of rows and columns given in the SDRAM data sheet. These examples do not cover non-JEDEC standard SDRAMS.

Table 22-14. Single 4M x 16 Control Register Values

Control Field	Value
Density	8 Mbyte
Page Size	512
ROW	12
COL	8
DSIZ	16 (D[15:0])
IAM	Bank Interleaved

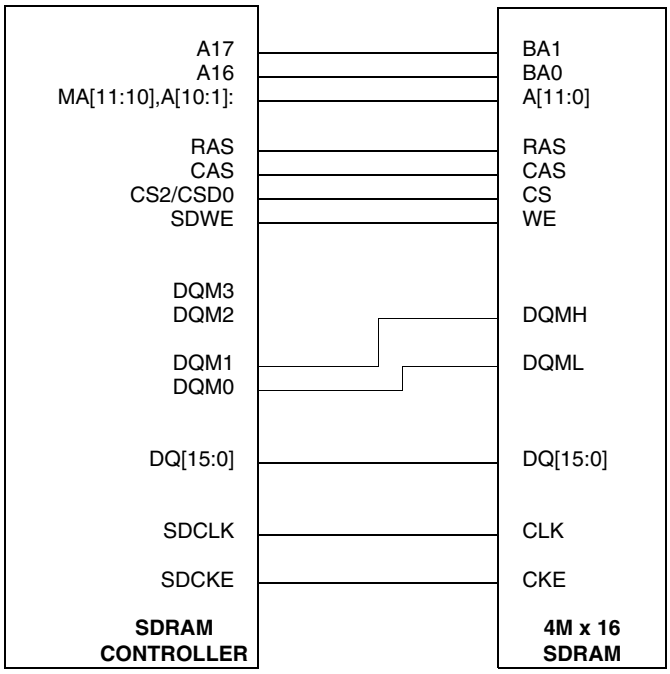


Figure 22-35. Single 64 Mbit (4M x 16) Connection Diagram (IAM = 1)

Table 22-15. Single 4M x 16 Control Register Values

Control Field	Value
Density	8 Mbyte
Page Size	512
ROW	12
COL	8
DSIZ	16 (D[15:0])
IAM	Non-bank Interleaved

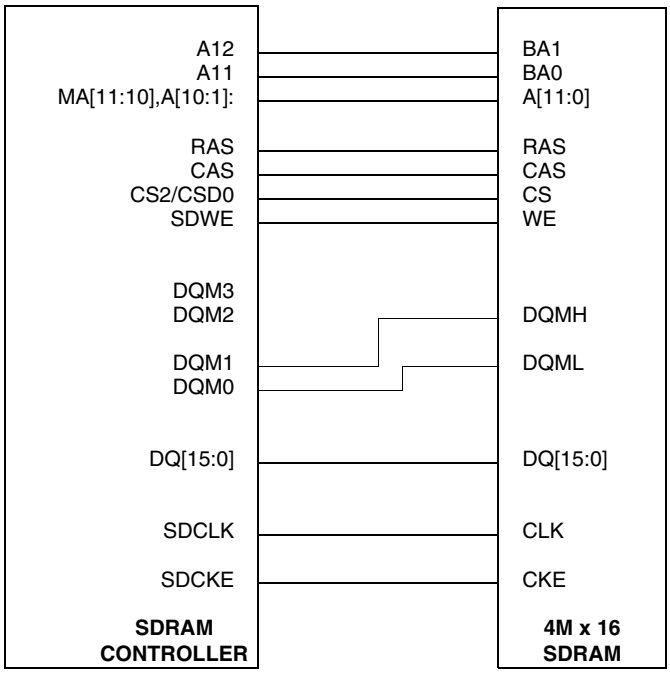


Figure 22-36. Single 64 Mbit (4M x 16 x 1) Connection Diagram (IAM = 0)

Table 22-16. Single 8M x 16 Control Register Values

Control Field	Value
Density	16 Mbyte
Page Size	1024 bytes
ROW	12
COL	9
DSIZ	16 (D[15:0])
IAM	Bank Interleaved

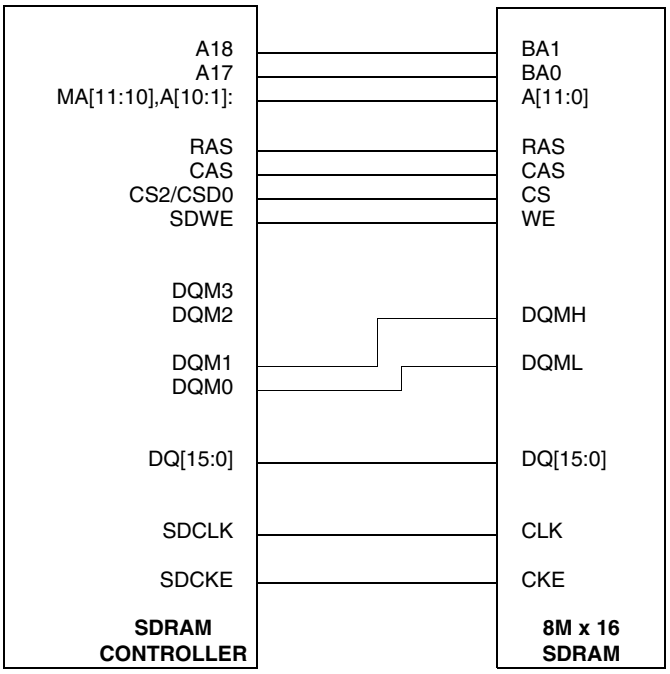


Figure 22-37. Single 128 Mbit (8M x 16) Connection Diagram (IAM = 1)

Table 22-17. Single 8M x 16 Control Register Values

Control Field	Value
Density	16 Mbyte
Page Size	1024 bytes
ROW	12
COL	9
DSIZ	16 (D[15:0])
IAM	Non-bank Interleaved

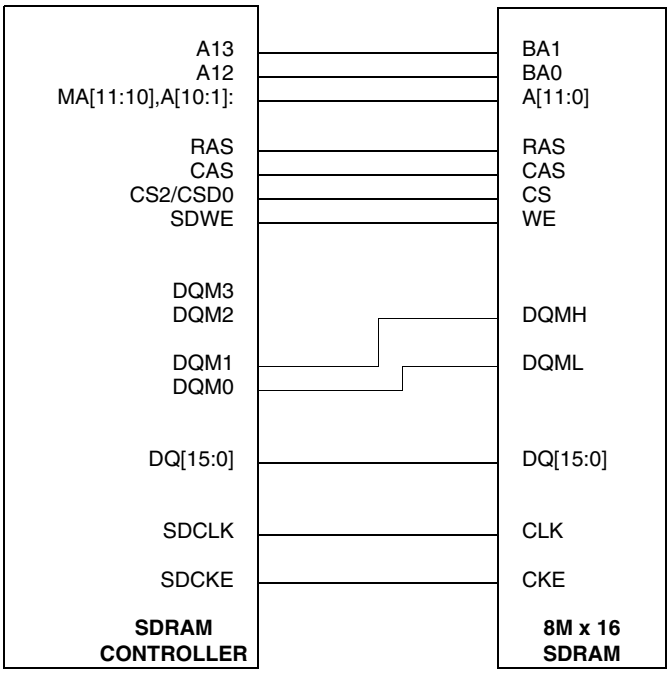


Figure 22-38. Single 128 Mbit (8M x 16) Connection Diagram (IAM = 0)

Table 22-18. Single 16M x16 Control Register Values

Control Field	Value
Density	32 Mbyte
Page Size	1024
ROW	13
COL	9
DSIZ	16 (D[15:0])
IAM	Bank-Interleaved

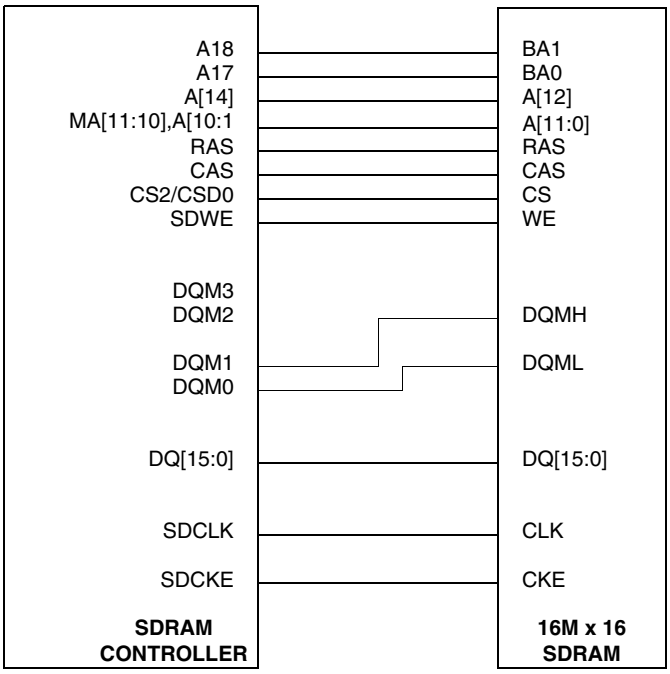


Figure 22-39. Single 256 Mbit (16M x 16) Connection Diagram (IAM = 1)

Table 22-19. Single 16M x16 Control Register Values

Control Field	Value
Density	32 Mbyte
Page Size	1024
ROW	13
COL	9
DSIZ	16 (D[15:0])
IAM	Non-Bank-Interleaved

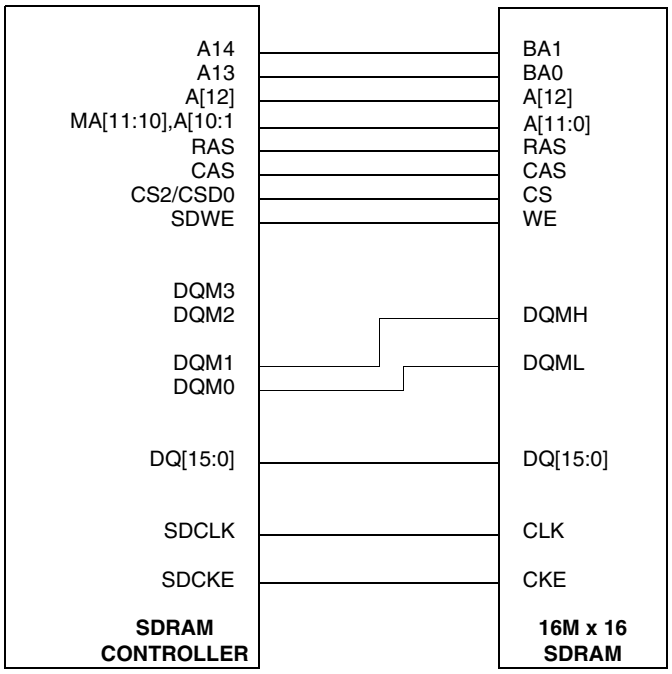


Figure 22-40. Single 256 Mbit (16M x 16) Connection Diagram (IAM = 1)

Table 22-20. Dual 64 Mbit (4M x 16 x 2) Control Register Values (IAM = 1)

Control Field	Value
Density	16 Mbyte
Page size	1024 bytes
ROW	12
COL	8
DSIZ	32 (D [31:0])
IAM	Bank–interleaved

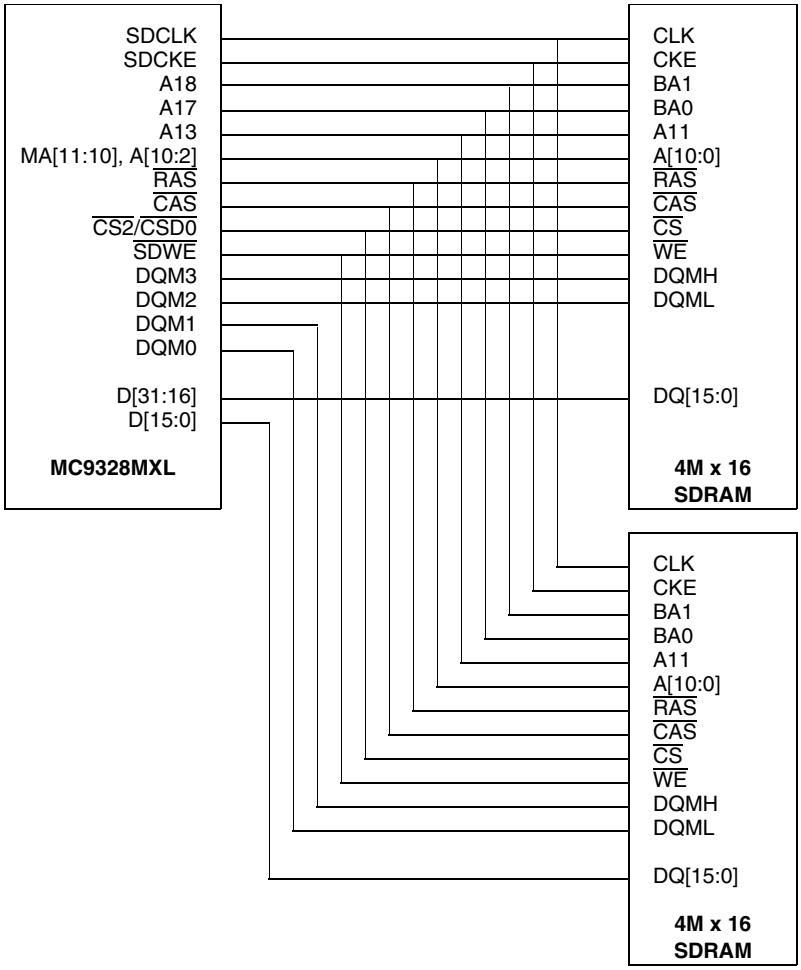


Figure 22-41. Dual 64 Mbit (4M x 16 x 2) Connection Diagram (IAM = 1)

Table 22-21. Dual 64 Mbit (4M x 16 x 2) Control Register Values (IAM = 0)

Control Field	Value
Density	16 Mbyte
Page size	1024 bytes
ROW	12
COL	8
DSIZ	32 (D [31:0])
IAM	Non-bank–interleaved

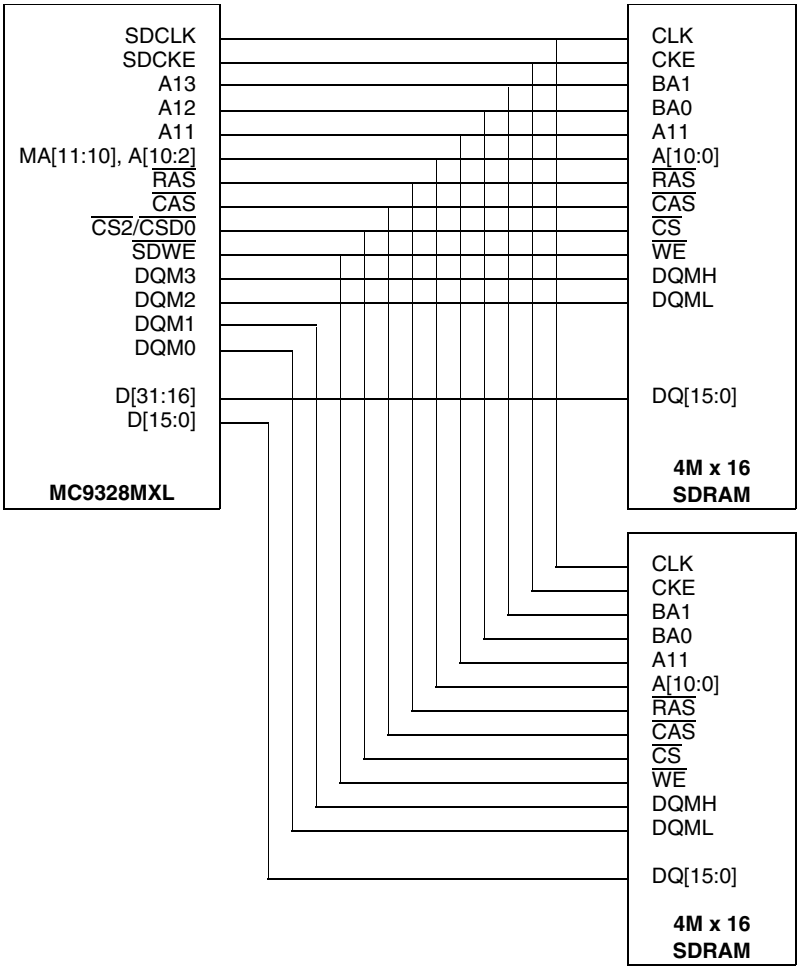


Figure 22-42. Dual 64 Mbit (4M x 16 x 2) Connection Diagram (IAM = 0)

Table 22-22. Dual 128 Mbit (8M x 16 x 2) Control Register Values (IAM = 1)

Control Field	Value
Density	32 Mbyte
Page size	2048 bytes
ROW	12
COL	9
DSIZ	32 (D [31:0])
IAM	Bank–interleaved

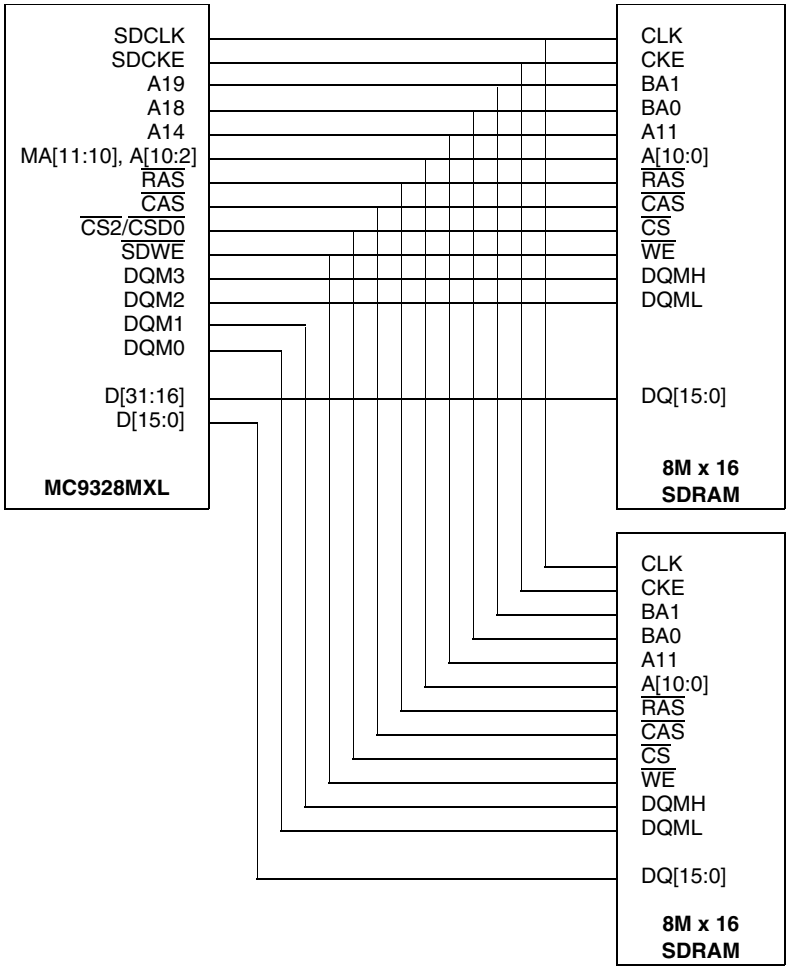


Figure 22-43. Dual 128 Mbit (8M x 16 x 2) Connection Diagram (IAM = 1)

Table 22-23. Dual 128 Mbit (8M x 16 x 2) Control Register Values (IAM = 0)

Control Field	Value
Density	32 Mbyte
Page size	2048 bytes
ROW	12
COL	9
DSIZ	32 (D [31:0])
IAM	Non-bank–interleaved

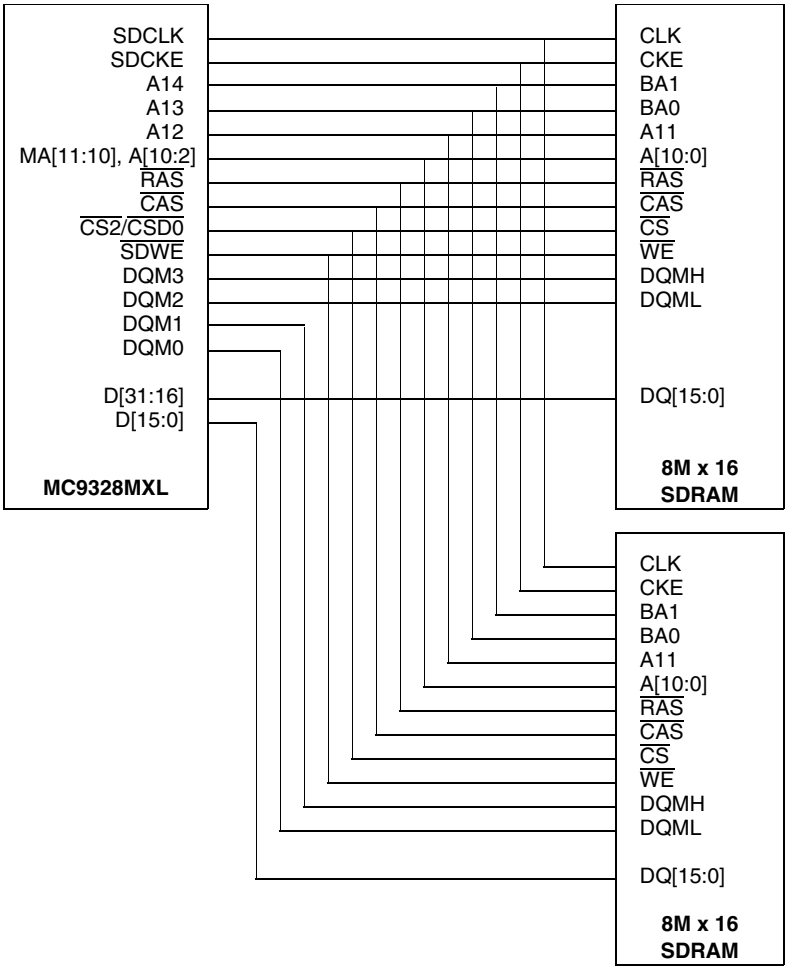


Figure 22-44. Dual 128 Mbit (8M x 16 x 2) Connection Diagram (IAM = 0)

Table 22-24. Dual 256 Mbit (16M x 16 x 2) Control Register Values (IAM = 1)

Control Field	Value
Density	64 Mbyte
Page size	2048 bytes
ROW	13
COL	9
DSIZ	32 (D [31:0])
IAM	Bank–interleaved

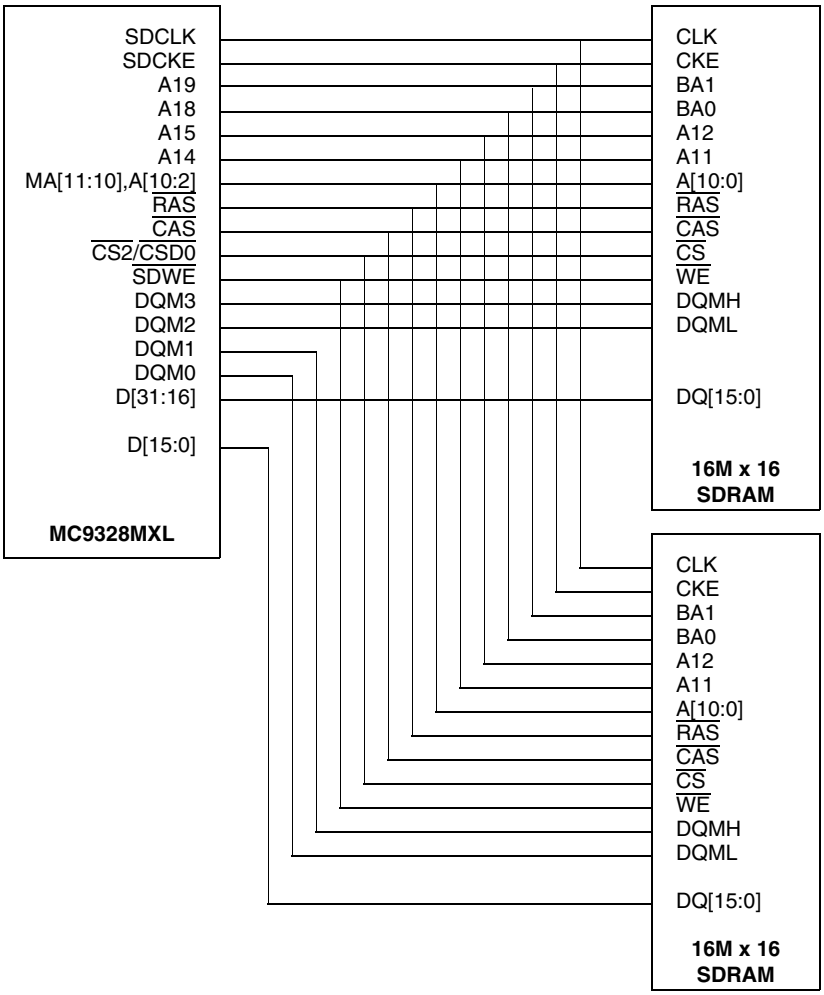


Figure 22-45. Dual 256 Mbit (16M x 16 x 2) Connection Diagram (IAM = 1)

Table 22-25. Dual 256 Mbit (16M x 16 x 2) Control Register Values (IAM = 0)

Control Field	Value
Density	64 Mbyte
Page size	2048 bytes
ROW	13
COL	9
DSIZ	32 (D [31:0])
IAM	Non-bank–interleaved

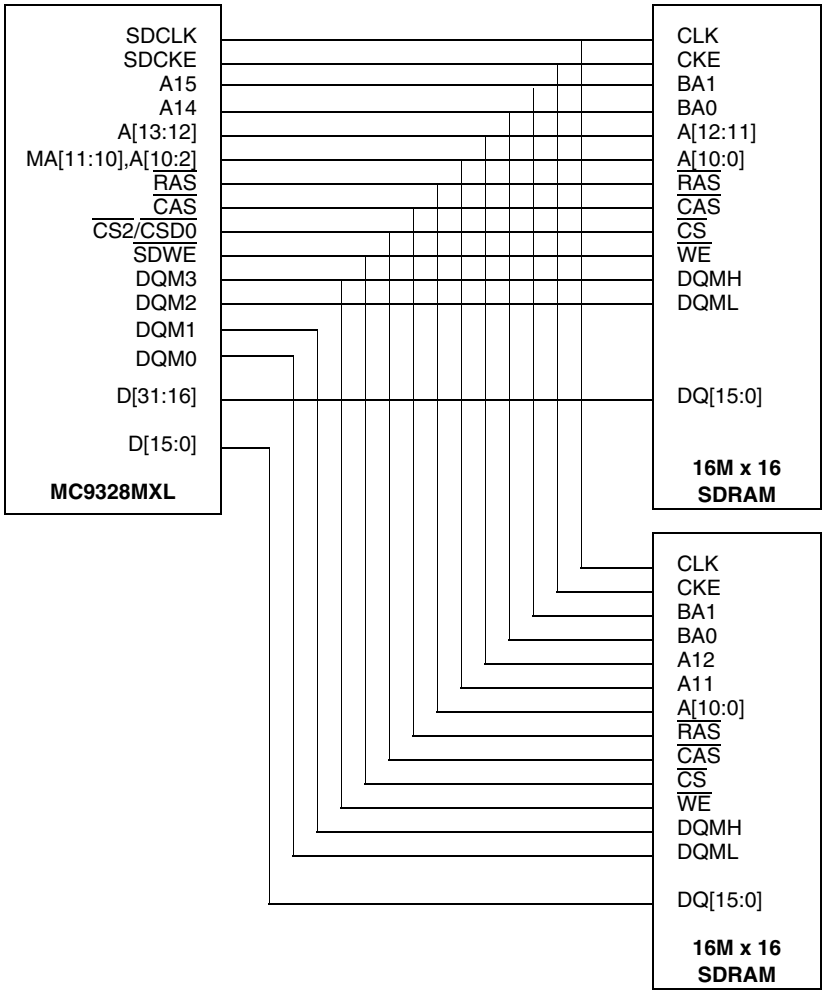


Figure 22-46. Dual 256 Mbit (16M x 16 x 2) Connection Diagram (IAM = 0)

Table 22-26. Single 64 Mbit (2M x 32) Control Register Values (IAM = 1)

Control Field	Value
Density	8 Mbyte
Page size	1024 bytes
ROW	11
COL	8
DSIZ	32 (D [31:0])
IAM	Bank–interleaved

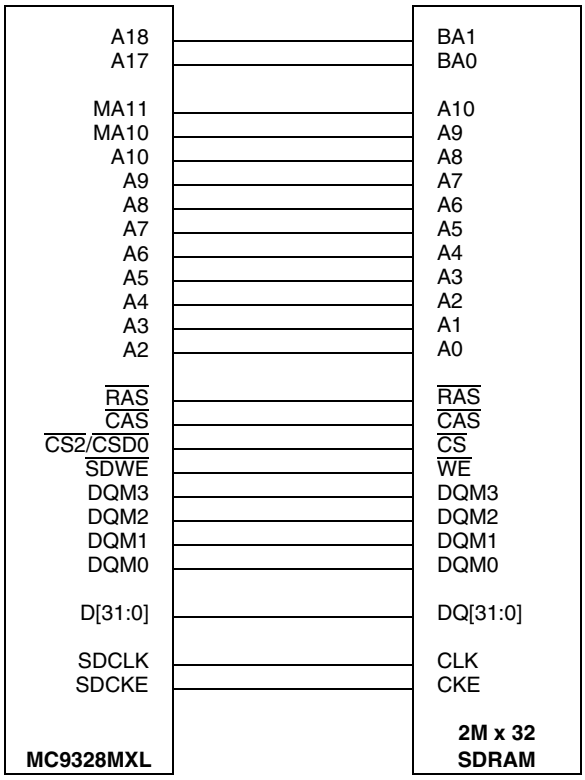


Figure 22-47. Single 64 Mbit (2M x 32) Connection Diagram (IAM = 1)

Table 22-27. Single 64 Mbit (2M x 32) Control Register Values (IAM = 0)

Control Field	Value
Density	8 Mbyte
Page size	1024 bytes
ROW	11
COL	8
DSIZ	32 (D [31:0])
IAM	Non-bank–interleaved

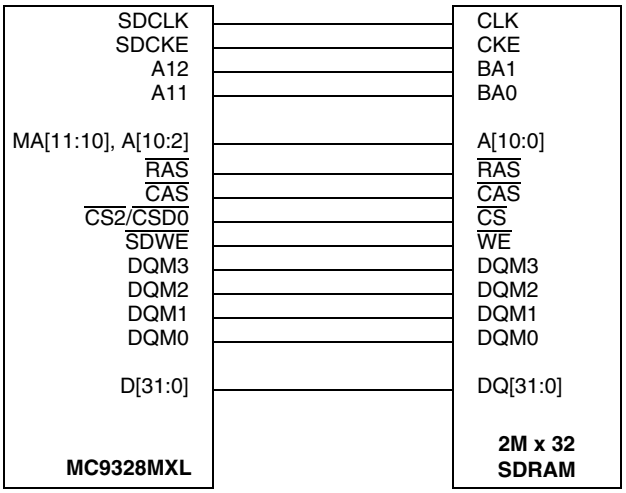


Figure 22-48. Single 64 Mbit (2M x 32) Connection Diagram (IAM = 0)

Table 22-28. Single 128 Mbit (4M x 32) Control Register Values (IAM = 1)

Control Field	Value
Density	16 Mbyte
Page size	1024 bytes
ROW	12
COL	8
DSIZ	32 (D [31:0])
IAM	Bank–interleaved

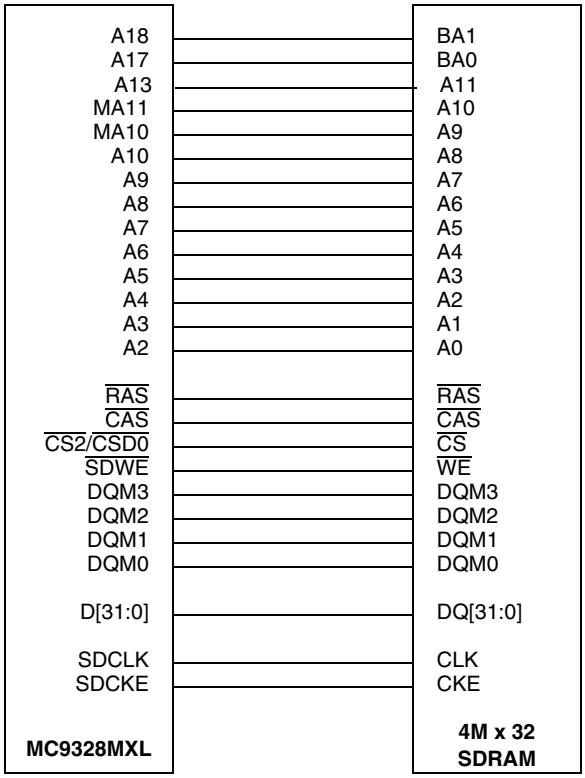


Figure 22-49. Single 128 Mbit (4M x 32) Connection Diagram (IAM = 1)

Table 22-29. Single 128 Mbit (4M x 32) Control Register Values (IAM = 0)

Control Field	Value
Density	16 Mbyte
Page size	1024 bytes
ROW	12
COL	8
DSIZ	32 (D [31:0])
IAM	Non-bank–interleaved

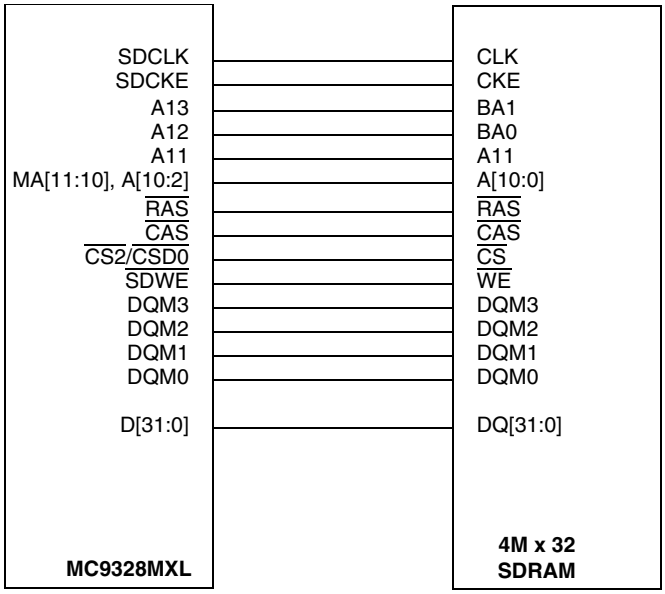


Figure 22-50. Single 128 Mbit (4M x 32) Connection Diagram (IAM = 0)

NOTE:

JEDEC has not issued a standard pinout and array configuration for the 128 Mbit density memories in a x32 package option. This connection diagram is based on the PC100 Standard.

Table 22-30. Single 256 Mbit (8M x 32) Control Register Values (IAM = 1)

Control Field	Value
Density	32 Mbyte
Page size	2048 bytes
ROW	12
COL	9
DSIZ	32 (D [31:0])
IAM	Bank-interleaved

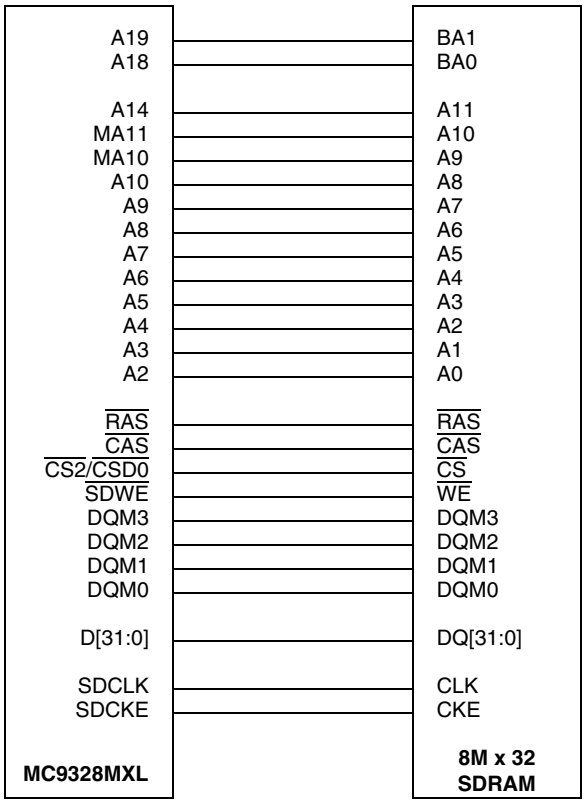


Figure 22-51. Single 256 Mbit (8M x 32) Connection Diagram (IAM = 1)

Table 22-31. Single 256 Mbit (8M x 32) Control Register Values (IAM = 0)

Control Field	Value
Density	32 Mbyte
Page size	2048 bytes
ROW	12
COL	9
DSIZ	32 (D [31:0])
IAM	Non-bank–interleaved

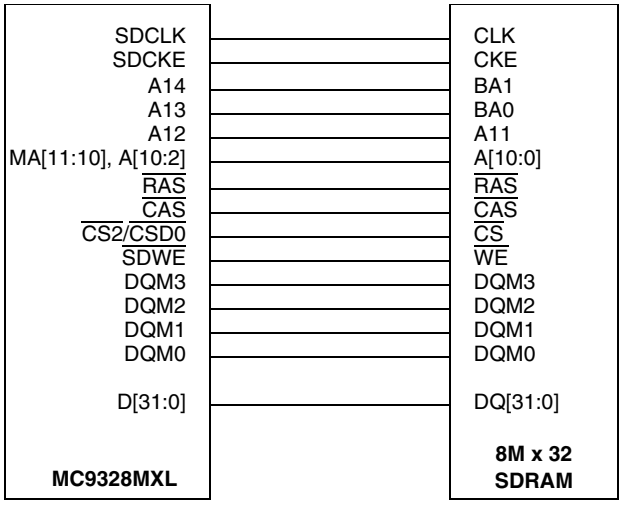


Figure 22-52. Single 256 Mbit (8M x 32) Connection Diagram (IAM = 0)

NOTE:

JEDEC has not issued a standard pinout and array configuration for the 256 Mbit density memories in a x32 package option. This connection diagram is based on the PC100 Standard.

22.7.3 SDRAM Reset Initialization

SDRAM initialization must follow a defined sequence following the power-on condition. The steps are as follows:

1. Apply power and start clock. Attempt to maintain CKE high, DQM high and NOP conditions at the command inputs.
2. Maintain stable power, clock, and NOP conditions for a minimum of 200 µs.
3. Issue precharge commands for all banks either with precharge all or precharge individual bank commands.
4. After all banks are in the idle state for a minimum time of t_{RP}, issue 8 or more auto-refresh commands.

5. Issue a mode register set command to initialize the mode register.
6. SDRAM is now ready for normal operation.

The SDRAM Controller accomplishes steps 1 and 2 in hardware, however it relies on software assistance to complete the remaining actions. The 200 µs stabilization period is guaranteed by the use of 2 reset signals whose negations are separated by this amount. An SDRAM reset signal ($\overline{SD_RST}$) is asserted to coincide with the system reset used by the rest of the chip, however it negates 200 ms prior to the negation of system reset. The SDRAM Controller leaves the SDRAM arrays in a NOP condition following the negation of the DRAM reset. Figure 22-53 shows the SDRAM Power-on initialization sequence.

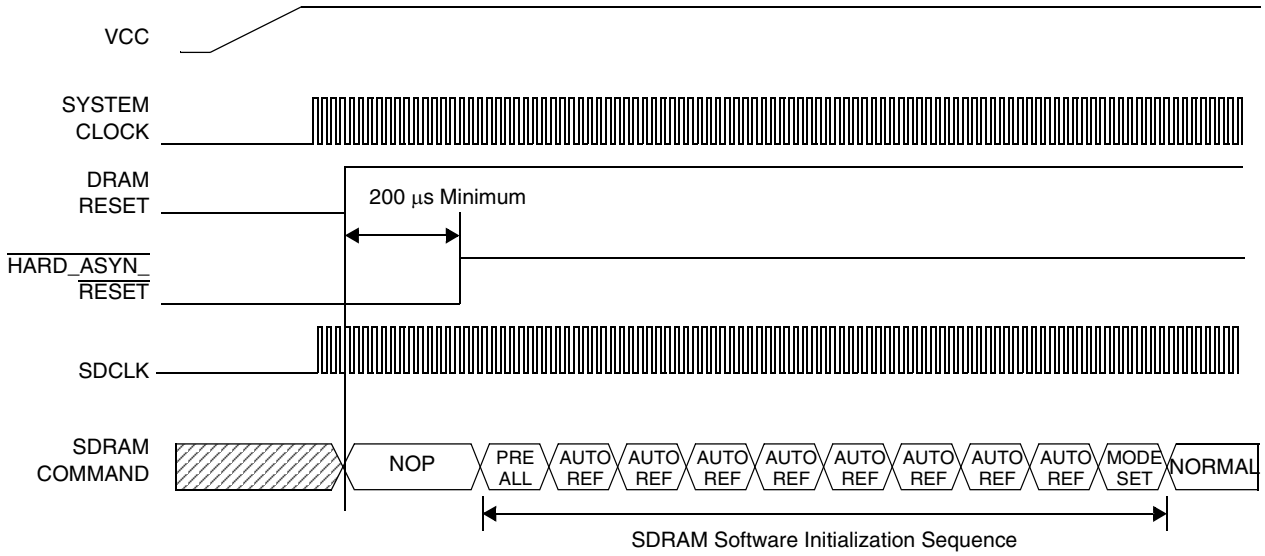


Figure 22-53. SDRAM Power-On Initialization Sequence

Following negation of system reset, initialization software must complete steps 3 through 5 using the special operating modes enabled by the SMODE field in the SDRAM Control Register. To precharge the SDRAM array, the SDRAM Controller operating mode is set to “precharge command” and an access is made to the SDRAM address range with address bit A10 = 1. Instead of running a normal read or write cycle, the controller issues a precharge all command to the addressed array. The operating mode is then switched to “auto-refresh” and 8 accesses are made to the SDRAM address space. Each of the accesses results in a refresh command to the addressed array. A “mode register set” command is required to complete the initialization sequence. The value written is system dependent. Consult Section 22.7.4, “Mode Register Programming,” for details. Finally, the controller is placed back in the normal mode of operation so that subsequent accesses to the address space result in normal read and write cycles to the SDRAM array.

Although the initialization sequence described in the previous paragraphs is only required at power-on, it may be repeated at any time the programmer deems necessary.

Code Example 22-2 on page 22-59 provides the code necessary for the initialization sequence.

Code Example 22-2. init_sdram

```
init_sdram:
        ldr     r2,CSD_REGS             // base address of registers
        ldr     r3,PRE_ALL_CMD
        st      r3,(0,r2)               // put array 0 in precharge command mode
        st      r3,(4,r2)               // put array 1 in precharge command mode
        ldr     r4,SDRAM_ARRAY_0        // get address of first array
        ldr     r5,SDRAM_ARRAY_1        // get address of second array
        ld      r3,(0,r4)               // precharge array 0
        ld      r3,(0,r5)               // precharge array 1
        ldr     r3,AUTO_REF_CMD
        st      r3,(0,r2)               // put array 0 in auto-refresh mode
        st      r3,(4,r2)               // put array 1 in auto-refresh mode
        movi    r6,7                    // load loop counter
L1      ld      r3,(0,r4)               // run auto-refresh cycle to array 0
        ld      r3,(0,r5)               // run auto-refresh cycle to array 1
        decgt   r6
        bt      L1                      // 8 refresh cycles complete?
        ldr     r3,SET_MODE_REG_CMD
        st      r3,(0,r2)               // setup CSD0 for mode register write
        st      r3,(4,r2)               // setup CSD1 for mode register write
        ldr     r3,MODE_REG_VAL0        // array 0 mode register value
        ld      r3,(0,r3)               // New mode register value on address bus
        ldr     r3,MODE_REG_VAL1        // array 1 mode register value
        ld      r3,(0,r3)               // Write CSD1 mode register
        ldr     r3,NORMAL_MODE
        st      r3,(0,r2)               // setup CSD0 for normal operation
        st      r3,(4,r2)               // setup CSD1 for normal operation

CSD_REGS        .long   0x00221000
SDRAM_ARRAY_0:  .long   0x08000000
SDRAM_ARRAY_1:  .long   0x0c000000
PRE_ALL_CMD     .long   0xXXXXXXXX
AUTO_REF_CMD    .long   0xXXXXXXXX
SET_MODE_REG_CMD.long   0xXXXXXXXX
MODE_REG_VAL0   .long   0xXXXXXXXX
MODE_REG_VAL1   .long   0xXXXXXXXX
NORMAL_MODE     .long   0xXXXXXXXX
```

22.7.4 Mode Register Programming

This section describes how to program the SDRAM mode register using the MC9328MXL external address bus. The mode register is used to set the SDRAM operating characteristics including CAS latency, burst length, burst mode, and write data length. The settings depend on system characteristics including the operating frequency, memory device type, burst buffer/cache line length, and bus width. Operating characteristics vary by device type, so the data sheet must be consulted to determine the actual value to be written.

Table 22-32 through Table 22-37 provide examples of the SDRAM memory mode register configuration for different SDRAM memory densities. These tables depict which mode register bit is connected to the SDRAM memory address pin. Because the mode register is written via the MC9328MXL external address bus, and the SDRAM data sheet specifies the SDRAM addresses on which to place the data, the next step is to then determine the address translation from the MC9328MXL internal address bus to the MC9328MXL external address pins. Table 22-38 on page 22-61 provides an example of which mode bits are mapped to which internal address bit for each of the given memory densities.

Table 22-32. 4M x 16 Memory Configuration

SDRAM Address	BA1	BA0	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Mode Register Bit	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
Content	Reserved				WB	Reserved		CAS latency			BT	Burst length		

Table 22-33. 8M x 16 Memory Configuration

SDRAM Address	BA1	BA0	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Mode Register Bit	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
Content	Reserved				WB	Reserved		CAS latency			BT	Burst length		

Table 22-34. 16M x 16 Memory Configuration

SDRAM Address	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Mode Register Bit	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
Content	Reserved			WB	Reserved		CAS latency			BT	Burst length		

Table 22-35. 2M x 32 Memory Configuration

SDRAM Address	BA1	BA0	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Mode Register Bit	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
Content	Reserved			WB	Reserved		CAS latency			BT	Burst length		

Table 22-36. 4M x 32 Memory Configuration

SDRAM Address	BA1	BA0	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Mode Register Bit	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
Content	Reserved				WB	Reserved		CAS latency			BT	Burst length		

Table 22-37. 8M x 32 Memory Configuration

SDRAM Address	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Mode Register Bit	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
Content	Reserved			WB	Reserved		CAS latency			BT	Burst length		

Table 22-38. MC9328MXL SDRAM Memory Configuration

		MC9328MXL Internal Address Bus (internal address bits are differentiated using the nomenclature A'x)																															
Memory Config.	IAM	A'31	A'30	A'29	A'28	A'27	A'26	A'25	A'24	A'23	A'22	A'21	A'20	A'19	A'18	A'17	A'16	A'15	A'14	A'13	A'12	A'11	A'10	A'9	A'8	A'7	A'6	A'5	A'4	A'3	A'2	A'1	A'0
4Mx16Bit x 2 Chips (16Mbyte)	0	0	0	0	0	X	X	0	0	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	X	X	0	0	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0	M13	M12	0	0	0	0	0	0	0	0	0	0
8Mx16Bit x 2 Chips (32 Mbyte)	0	0	0	0	0	X	X	0	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	X	X	0	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0	M13	M12	0	0	0	0	0	0	0	0	0	0	0
16Mx16Bit x 2 Chips (64 Mbyte)	0	0	0	0	0	X	X	0	0	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	X	X	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0	0	0	0	0	0	0	0	0	0	0	0	0	0
2Mx32Bit x 1 Chip (8 Mbtye)	0	0	0	0	0	X	X	0	0	0	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	X	X	0	0	0	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0	M12	M11	0	0	0	0	0	0	0	0	0	0
4Mx32Bit x 1 Chip (16 Mbyte)	0	0	0	0	0	X	X	0	0	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	X	X	0	0	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0	M13	M12	0	0	0	0	0	0	0	0	0	0
8Mx32Bit x 1 Chip (32 Mbyte)	0	0	0	0	0	X	X	0	0	0	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	X	X	0	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0	0	0	0	0	0	0	0	0	0	0	0	0
	CSDx Base							SDRAM Memory Mode Register Bits																									

22.7.5 Mode Register Programming Examples

To illustrate how these tables are used and as a demonstration of the programming procedure for the mode register, two examples are provided. Refer to the SDRAM data sheet for up-to-date characteristics, as the values used in the example may change from the date of this document's publication.

22.7.5.1 Example 1—256 Mbit SDRAM Mode Register

For Example 1, the following system characteristics will be used:

- 2 Micron MT48LC16M16A2-7E SDRAMs configured as a x32 memory (16M × 16 bits × 2 chips)
- 100 MHz system clock frequency
- Non-bank interleaved mode (IAM = 0)
- Burst length of 8 (not optional as the MC9328MXL performs 8 word burst reads)
- Single word writes (not optional as the MC9328MXL performs single writes only and does not provide a burst terminate after each write)

Table 22-39 illustrates the Mode register bit assignments for the Micron 256 Mbit SDRAM.

Table 22-39. 256 Mbit SDRAM Mode Register

SDRAM Address	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Mode Register Bit	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
Content	Reserved			WB	Reserved		CAS latency			BT	Burst length		

Table 22-40. 256 Mbit SDRAM Mode Register Description

Name	Description	Settings
WB Bit M9	**Write Burst Mode**—Needs to be programmed to 1 for single location accesses.	0 = Burst writes (not supported by MC9328MXL) 1 = Single word writes
CAS Latency Bits M6–M4	**CAS Latency**—Sets latency between column address and data. Note, the MC9328MXL does not support CAS Latencies of 1.	000 = Reserved 001 = Reserved 010 = 2 clocks 011 = 3 clocks 1xx = Reserved
BT Bit M3	**Burst Type**—Selects sequential or interleaved bursts.	0 = Sequential 1 = Interleaved

Table 22-40. 256 Mbit SDRAM Mode Register Description (continued)

Name	Description	Settings
Burst Length Bits M2–M0	**Burst Length**—A burst length of 8 matches the ARM920T cache line length.	000 = 1 001 = 2 010 = 4 011 = 8 111 = Full page (if BT = 0) 10x = Reserved 1x0 = Reserved

The values programmed into the SDRAM Mode register for example 1 are as follows:

- Sequential burst (BT = 0)
- Burst length of 8 (BL = 011), not optional
- Single word writes (WB = 1), not optional
- 2 Clock latency (LTMODE = 010)

When the Mode register value has been determined, it must be converted to an address. The Mode register is written via the address bus and the memory data sheet specifies the SDRAM address bits on which to place the data. One final transformation is necessary to align the address to the multiplexed outputs of the SDRAM controller. Memory density and bus width determine the alignment of the SDRAM to the controller pins and must be considered during the calculation.

The first step is to determine the value of the 256 Mbit SDRAM Mode register and convert this into an SDRAM address. Table 22-41 is the same as Table 22-39, however it includes the values for this 256 Mbit SDRAM example. This table illustrates what value needs to be placed on the address bus to the SDRAM memory.

Table 22-41. 256 Mbit SDRAM Mode Register with Values

SDRAM Address	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Mode Register Bit	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
Content	Reserved			WB	Reserved		CAS latency			BT	Burst length		
Value	0	0	0	1	0	0	0	1	0	0	0	1	1

The next step is to determine that the proper value is written to the MC9328MXL internal address bus to ensure that the SDRAM controller's ROW/COLUMN ADDRESS MUX writes these values to the correct address pins of the MC9328MXL's external address bus and then to the SDRAM memory. Table 22-38 simplifies this procedure by allowing the user to simply plug in the Mode register bits into this table therefore generating the correct address to write from the MC9328MXL. Referring to Table 22-38 and locating the proper SDRAM memory density (in this case the 16M × 16 SDRAM), proceed by plugging in the Mode register bits into this table. Table 22-42 illustrates this procedure.

Table 22-42. MC9328MXL Address Calculation for Given Mode Register Values

Internal Address	A'31	A'30	A'29	A'28	A'27	A'26	A'25	A'24	A'23	A'22	A'21	A'20	A'19	A'18	A'17	A'16	A'15	A'14	A'13	A'12	A'11	A'10	A'9	A'8	A'7	A'6	A'5	A'4	A'3	A'2	A'1	A'0
Mode Register Bit	0	0	0	0	X	X	0	0	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0	0	0	0	0	0	0	0	0	0	0
Mode Register Bit Value	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0

Table 22-42 assumes CSD0 is being used as the chip-select for the SDRAM memory, therefore the chip-select base address bits, A'31 through A'24, are set to 00001000 for the memory map region 0x08000000. As a result, for this example, the final value (in hexadecimal format) written to the MC9328MXL internal address for proper translation to the SDRAM memory mode register is 0x08111800. The procedure would then be to issue a Set Mode Register Command to the SDRAM memory, followed by an access (either READ or WRITE) to the SDRAM memory at address 0x08111800. This value is also written into the MODE_REG_VAL0 variable of Code Example 22-2 on page 22-59. Refer to Section 22.5.5, "Set Mode Register Mode (SMODE = 011)," on page 22-25 for more information on the Set Mode Register Command.

22.7.5.2 Example 2—64 Mbit SDRAM Mode Register

For Example 2, the following system characteristics are used:

- 2 Mitsubishi M5M4V64S40ATP-8 64Mbit (4M × 16) SDRAMs configured as a x32 memory (4M × 16 bits × 2 chips)
- 100 MHz system clock frequency
- Bank-interleaved mode (IAM = 1)
- Burst length of 8 (not optional as the MC9328MXL performs 8 word burst reads)
- Single word writes (not optional as the MC9328MXL performs single writes only and does not provide a burst terminate after each write)

Table 22-43 illustrates the Mode Register bit assignments for the Mitsubishi 64 Mbit SDRAM and Table 22-44. provides the register descriptions.

Table 22-43. 64 Mbit SDRAM Mode Register

SDRAM Address	BA1	BA0	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Mode Register Bit	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
Content	Reserved				WB	Reserved		CAS latency			BT	Burst length		

Table 22-44. 64 Mbit SDRAM Mode Register Description

Name	Description	Settings
WB Bit M9	**Write Burst Mode**—Needs to be programmed to 1 for single location accesses.	0 = Burst writes (not supported by MC9328MXL) 1 = Single word writes

Table 22-44. 64 Mbit SDRAM Mode Register Description (continued)

CAS Latency Bits M6–M4	**CAS Latency**—Sets latency between column address and data. Note, the MC9328MXL does not support CAS Latencies of 1.	000 = Reserved 001 = Reserved 010 = 2 clocks 011 = 3 clocks 1xx = Reserved
BT Bit M3	**Burst Type**—Selects sequential or interleaved bursts.	0 = Sequential 1 = Interleaved
Burst Length Bits M2–M0	**Burst Length**—A burst length of 8 matches the ARM920T cache line length.	000 = 1 001 = 2 010 = 4 011 = 8 111 = Full page (if BT = 0) 10x = Reserved 1x0 = Reserved

The values programmed into the SDRAM mode register for Example 2 are as follows:

- Sequential burst (BT = 0)
- Burst length of 8 (BL = 011), not optional
- Single word writes (WB = 1), not optional
- 3 clock latency (LTMODE = 011)

In example 1, we saw that once the mode register value has been determined, it must be converted to an address. The mode register is written via the address bus and the data sheet for the memory being used specifies the SDRAM address bits on which to place the data. One final transformation is necessary to align the address to the multiplexed outputs of the SDRAM controller. Memory density and bus width determine the alignment of the SDRAM to the controller pins and must be defined during the calculation.

To defined memory density and bus width, the value of the 64 Mbit SDRAM mode register must be determined and converted into a SDRAM address. Table 22-45 is the same as Table 22-43, however it includes the values given in the 64 Mbit SDRAM mode register—example 2. Table 22-45 illustrates what value needs to be placed on the address bus to the SDRAM memory.

Table 22-45. 64 Mbit SDRAM Mode Register with Values

SDRAM Address	BA1	BA0	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Mode Register Bit	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
Content	Reserved				WB	Reserved		CAS latency			BT	Burst length		
Value		0	0	0	1	0	0	0	1	1	0	0	1	1

The next step is to determine the proper value that is written to the MC9328MXL internal address bus so the SDRAM controller ROW/COLUMN ADDRESS MUX may properly write these values to the correct address pins of the MC9328MXL external address bus and then to the SDRAM memory. Table 22-38 on page 22-61 simplifies this procedure by allowing the user to simply plug in the correct mode register bits into this table therefore generating the correct address to write from the MC9328MXL. Referring to Table 22-38 and locating the proper SDRAM memory density (in this case the 4M×16 SDRAM), we can proceed by entering the mode register bits into this table. Table 22-46 illustrates this procedure.

Table 22-46. MC9328MXL Address Calculation for Given Mode Register Value

Internal Address	A'31	A'30	A'29	A'28	A'27	A'26	A'25	A'24	A'23	A'22	A'21	A'20	A'19	A'18	A'17	A'16	A'15	A'14	A'13	A'12	A'11	A'10	A'9	A'8	A'7	A'6	A'5	A'4	A'3	A'2	A'1	A'0
Mode Register Bit	0	0	0	0	X	X	0	0	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0	M13	M12	0	0	0	0	0	0	0	0	0	0
Mode Register Bit Value	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-42 assumes CSD0 is being used as the chip-select for the SDRAM memory, therefore the chip-select base address bits, A’31 through A’24, are set to 00001000 for the memory map region 0x08000000. Therefore, for this example, the final value (in hexadecimal format) that is written to the MC9328MXL internal address for proper translation to the SDRAM memory mode register is 0x08233000. The procedure then must issue a Set Mode Register Command to the SDRAM memory, followed by an access (either READ or WRITE) to the SDRAM memory at address 0x08233000. This value will also be written into the “MODE_REG_VAL0” variable of Code Example 22-2 on page 22-59. Refer to Section 22.5.5, “Set Mode Register Mode (SMODE = 011),” for more information on the Set Mode Register Command.

22.7.6 SDRAM Memory Refresh

SDRAM memory specifications generally specify an interval during which all rows in the device must be refreshed. The memory refresh requirements are outlined in Table 22-47. The SDRAM Controller refresh rate (SREFR field of the Control Register) is programmable to meet these varying requirements. Refresh must be enabled prior to storing data in the memory.

Table 22-47. SDRAM Memory Refresh

Device Size	Array Size	Refresh Interval	Refresh Rate	Requirement	SREFR Value
64 Mbit	4096 rows	64 ms	1 row every 15.6 µs	2 rows refreshed during each 32 kHz clock	10
128 Mbit	4096 rows	64 ms	1 row ever 15.6 µs	2 rows refreshed during each 32 kHz clock	10
256 Mbit	8192 rows	64 ms	1 row every 7.8 µs	4 rows refreshed during each 32 kHz clock	11

NOTE:

The memory data sheet must always be consulted to determine the correct refresh interval and array architecture (number of rows). Refresh clock rates other the nominal value require recalculation of the value to be programmed into REFR.

22.8 SyncFlash Operation

Micron SyncFlash memories are 3 V electrically sector-erasable, non-volatile memories architected to comply with the SDRAM interface specifications. With few exceptions, they are capable of matching all SDRAM operating modes.

22.8.1 SyncFlash Reset Initialization

SyncFlash initialization is considerably more straightforward than SDRAM. The SyncFlash sequence is:

1. Apply power to Vcc, VccQ, and VccP simultaneously.
2. Apply clock.
3. After clock is stable, transition $\overline{\text{RESET_SF}}$ from low to high.
4. Maintain power, stable clock, and $\overline{\text{RESET_SF}}$ high for minimum of 100 µs.
5. Initialization is now complete.

The SDRAM Controller asserts $\overline{\text{RESET_SF}}$ low anytime $\overline{\text{sd_rst}}$ is asserted. The 200 µs delay between the negation of $\overline{\text{sd_rst}}$ and negation of $\overline{\text{m_rst}}$ easily meets the 100 µs stabilization requirement of the SyncFlash.

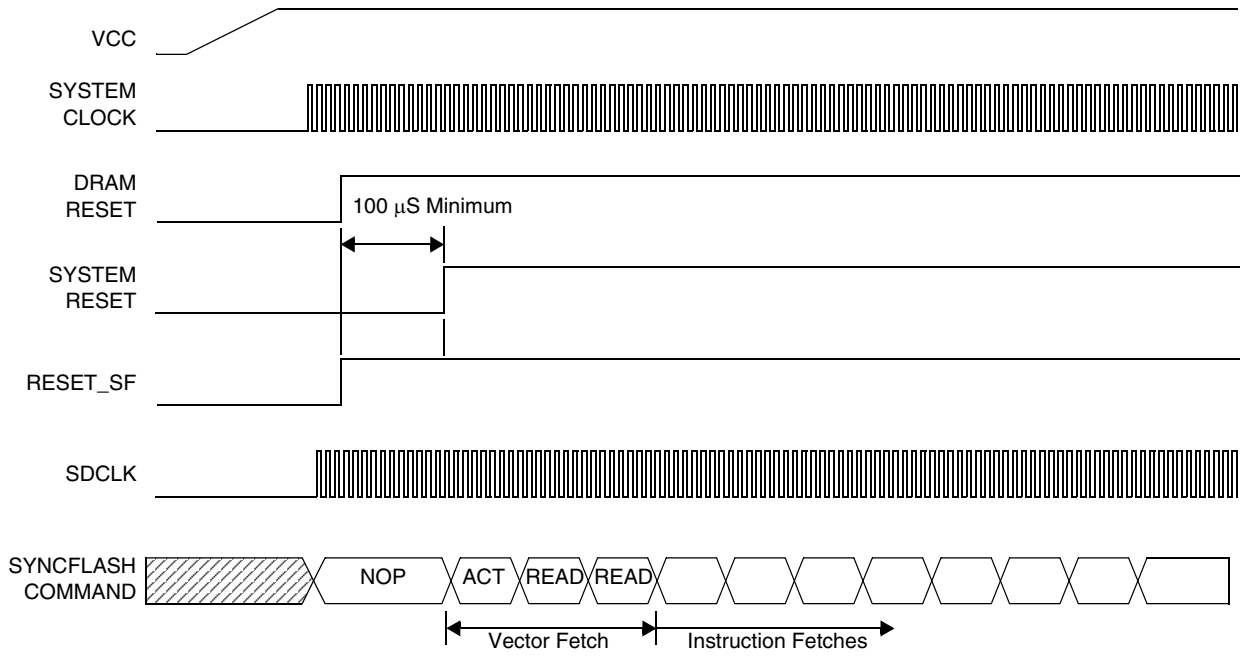


Figure 22-54. Sync Flash Reset Timing

22.8.2 SyncFlash Mode Register Programming

The mode register can be programmed following the initialization sequence, although generally it is not required. A non-volatile mode register is initially programmed at the same time as the rest of the array. This non-volatile register is copied into the mode register automatically during device initialization, and does not require reloading prior to the first operational command. Software may overwrite the default value at any time the memory is idle using the same sequence as an SDRAM mode register. Consult Section 22.7.4, “Mode Register Programming,” for details.

Programming the SyncFlash non-volatile mode register requires a specialized sequence of load command register triplets with VccP applied. Consult the SyncFlash data sheet for details.

22.8.3 Booting From SyncFlash

The SDRAM Controller is designed to permit booting from the SyncFlash device immediately out of reset. Default values in the configuration register allow booting at frequencies up to 100 MHz. Complete initialization of the controller is still required, however, and must be completed as quickly as possible.

22.8.4 SyncFlash Configuration

Hardware connections are similar to those for SDRAM of like density. One difference is that a SyncFlash boot device is limited to using $\overline{\text{CSD1}}$. SyncFlash can be connected to $\overline{\text{CSD0}}$, however, it cannot be the boot device. The second difference is the added connection to the reset/powerdown pin ($\overline{\text{RESET_SF}}$). An example of a 32-bit configuration is provided in Figure 22-56.

The only significant difference in the software configuration is that refresh *must* be disabled. SyncFlash maps the control register access commands to the same basic commands as SDRAM refresh. Enabling hardware refresh would most likely result in unexpected behavior.

Table 22-48. Single 4M x 16 SyncFlash Control Register Values

Control Field	Value
Density	8 Mbyte
Page Size	512
ROW	12
COL	8
DSIZ	16 (D[15:0])
IAM	Non-Interleaved

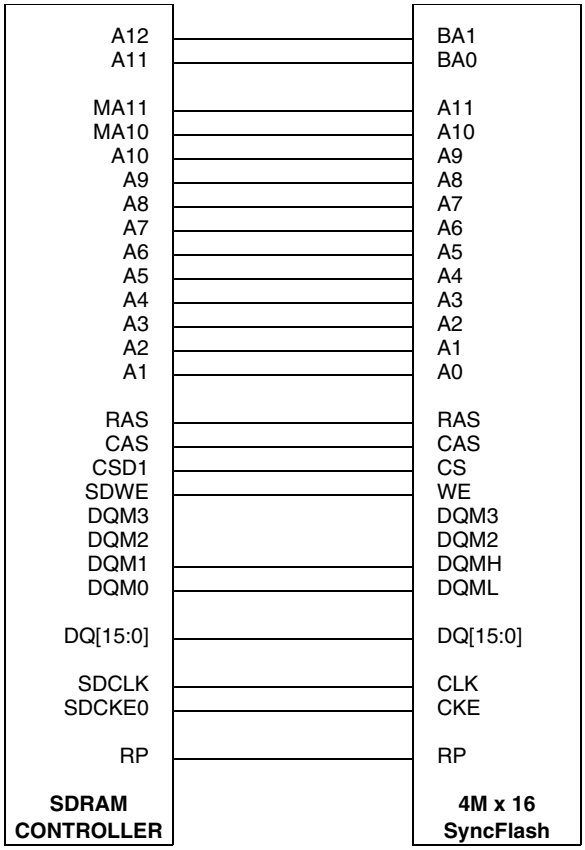


Figure 22-55. Single 64 Mbit SyncFlash Connection Diagram (IAM = 0)

Table 22-49. Dual 4M x 16 SyncFlash Control Register Values (IAM = 0)

Control Field	Value
Density	16 Mbyte
Page size	1024
ROW	12
COL	8
DSIZ	32 (D [31:0])
IAM	Non-Interleaved

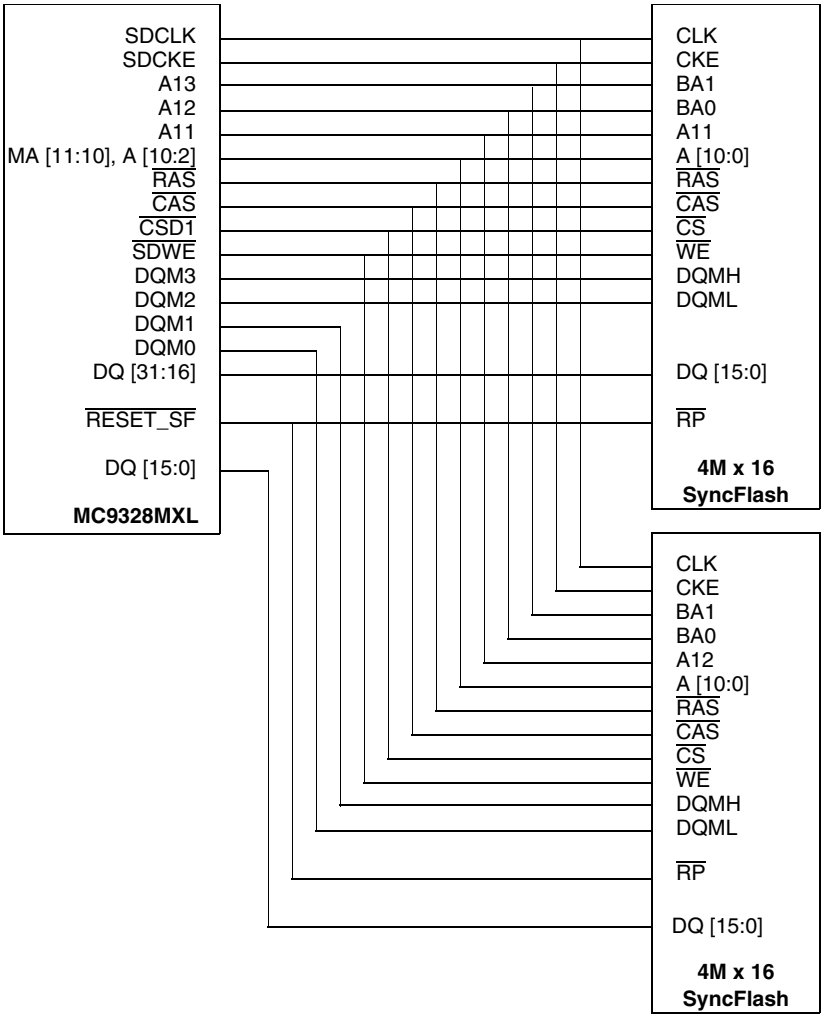


Figure 22-56. Dual 64 Mbit SyncFlash Connection Diagram (IAM = 0)

22.8.5 SyncFlash Programming

Table 22-50 provides a quick reference of the available commands for SyncFlash operation. The detailed command encoding for SDRAM compatibility, is located in Table 22-10. All SyncFlash operations with LCR (LOAD COMMAND REGISTER), LCR/ACTIVE/READ, or LCR/ACTIVE/WRITE commands and command sequences are defined in Table 22-50 and Table 22-10.

Table 22-50. SyncFlash Command Sequences

Operation	1st Cycle				2nd Cycle				3rd Cycle			
	CMD	ADDR	ADDR	DQ	CMD	ADDR	ADDR	DQ	CMD	ADDR	ADDR	DQ
Read Device Configuration	LCR	90H	Bank	X	ACT	Row	Bank	X	READ	CA	Bank	X
Read Status Register	LCR	70H	X	X	ACT	X	X	X	READ	X	X	X
Clear Status Register	LCR	50H	X	X	—	—	—	—	—	—	—	—
Erase Setup/ Confirm	LCR	20H	Bank	X	ACT	Row	Bank	X	WRITE	X	Bank	D0H
Write Setup/ Confirm	LCR	40H	Bank	X	ACT	Row	Bank	X	WRITE	Col	Bank	D1H
Protect Block/ Confirm	LCR	60H	Bank	X	ACT	Row	Bank	X	WRITE	X	Bank	01H
Protect Device/ Confirm	LCR	60H	Bank	X	ACT	X	Bank	X	WRITE	X	Bank	F1H
Unprotect Blocks/ Confirm	LCR	60H	Bank	X	ACT	X	Bank	X	WRITE	X	Bank	D0H
Erase NVMODE Register	LCR	30H	Bank	X	ACT	X	Bank	X	WRITE	X	Bank	C0H
Write NVMODE Register	LCR	A0H	Bank	X	ACT	X	Bank	X	WRITE	X	Bank	X

22.8.6 Clock Suspend Timer Use with SyncFlash

The SyncFlash enters clock suspend mode when the CLKST[1:0] bits are set to 10 or 11 in the SDRAM control register (SDCTL0/1). The clock to the SyncFlash memory will stop after a programmable delay following the last access to the SyncFlash (64 clock cycles for CLKST=10 or 128 clock cycles for CLKST = 11). When CLKST is set to 10 or 11, the active bank is not closed (no Precharge command is issued) before the entry into the clock suspend mode. Figure 22-57 illustrates this timing relationship.

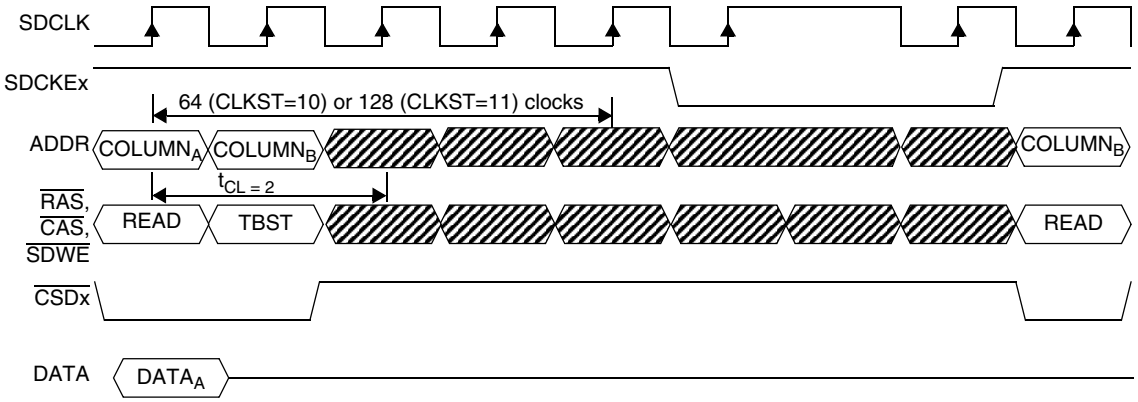


Figure 22-57. SyncFlash Clock Suspend Timer Operation Timing Diagram

22.8.7 Powerdown Operation with SyncFlash

The SyncFlash enters powerdown mode when the (CLKST [1:0] bits are set to 01 in the SDRAM control register (SDCTL0/1) and only when no bank is active. In powerdown mode, the clock to the SyncFlash memory will stop. Again, the powerdown mode can only entered when all banks within the memory area are inactive. Figure 22-58 illustrates the operation of SyncFlash powerdown mode.

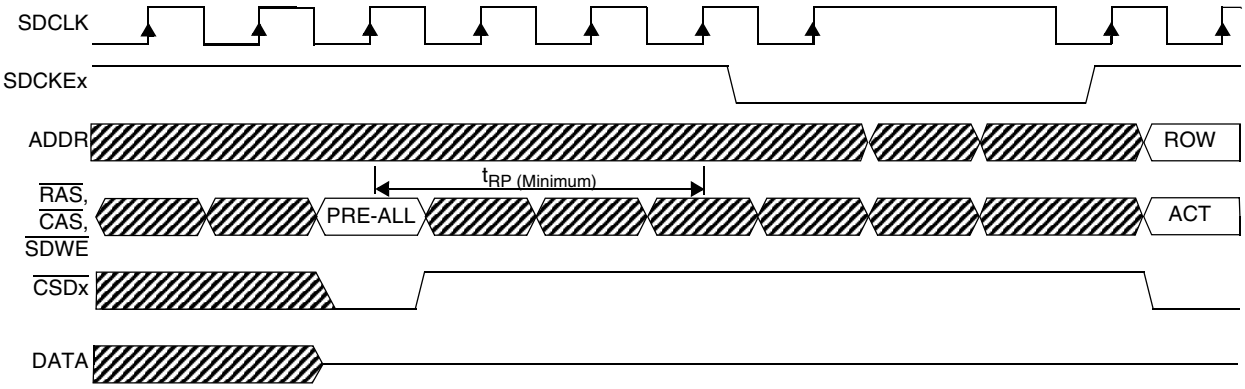


Figure 22-58. SyncFlash Powerdown Operation Timing Diagram

22.9 Deep Powerdown Operation with SyncFlash

The SyncFlash enters deep powerdown mode when the MC9328MXL enters stop mode (both the MCU PLL and System PLL are shut down). Upon entry of deep powerdown mode, all active memory banks are closed, the clock input to the SyncFlash stops, and the $\overline{\text{RESET_SF}}$ signal is asserted. The SyncFlash exits deep powerdown mode

after the MC9328MXL exits stop mode (when the MCU PLL and System PLL have waken up) and the clock to the SyncFlash is stable. The $\overline{\text{RESET_SF}}$ signal is then deasserted. Figure 22-59 illustrates the operation of the SyncFlash when entering deep powerdown mode.

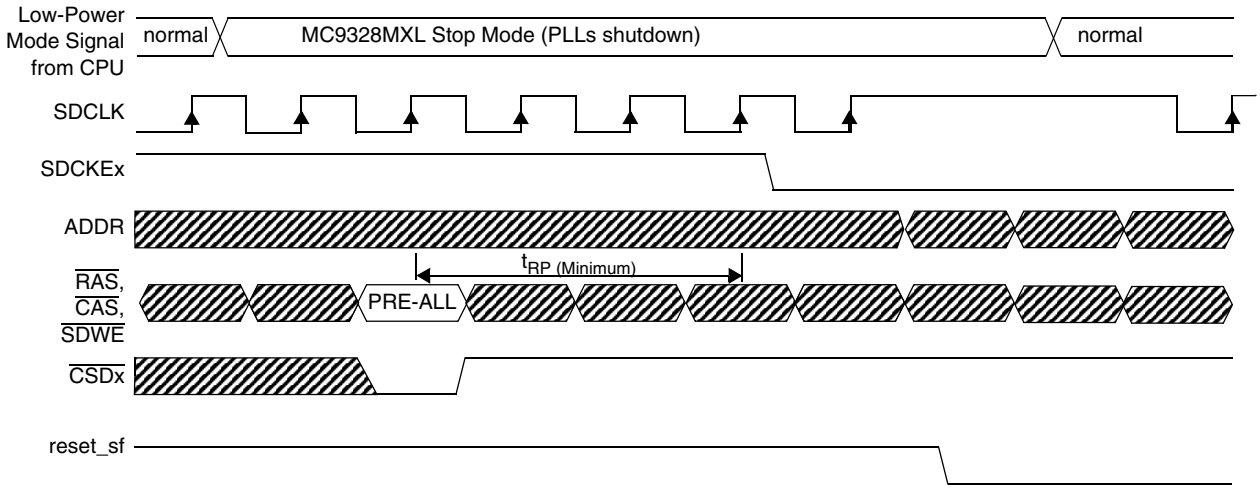


Figure 22-59. SyncFlash Deep Powerdown Operation Timing Diagram

Chapter 23
General-Purpose Timers

The MC9328MXL contains two identical general-purpose 32-bit timers with programmable prescalers and compare and capture registers. Each timer counter value can be captured using an external event and can be configured to trigger a capture event on either the leading or trailing edges of an input pulse. The timer can also generate an interrupt when the timer reaches a programmed value. Each timer has an 8-bit prescaler providing a programmable clock frequency derived from PERCLK1. Figure 23-1 illustrates the general-purpose timers block diagram.

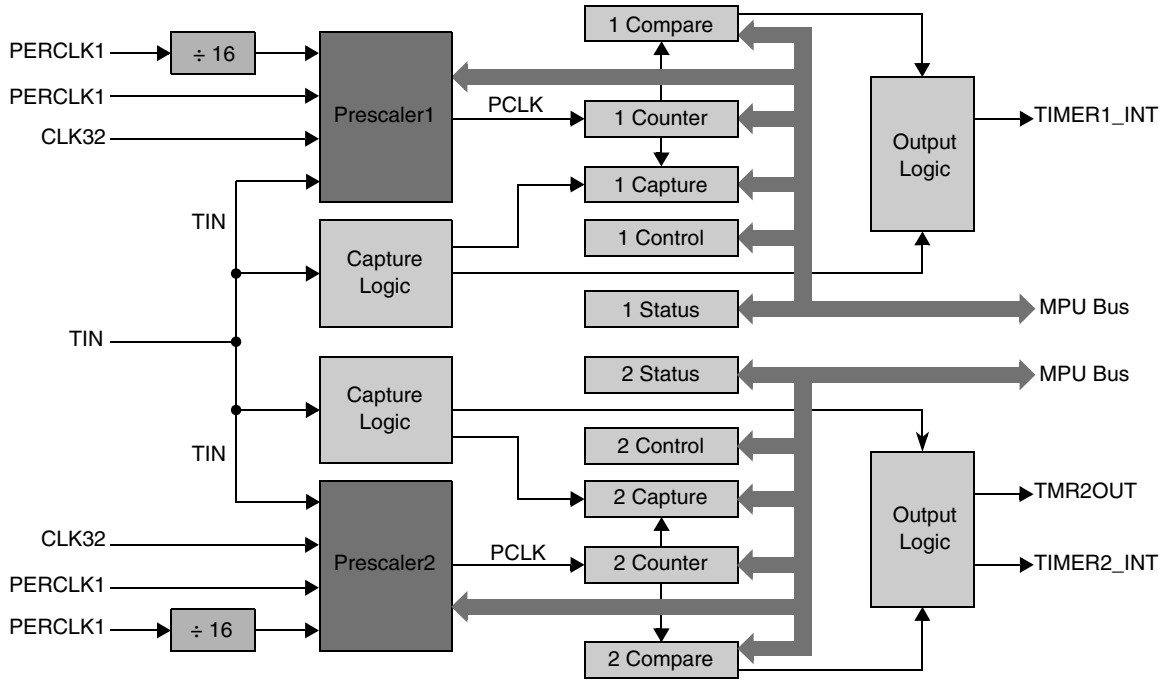


Figure 23-1. General-Purpose Timers Block Diagram

The timers have the following features:

- Maximum period of 512 × 65536 seconds at 32.768 kHz
- 10 ns resolution at 100 MHz
- Programmable sources for the clock input, including external clock
- Input capture capability with programmable trigger edge
- Output compare with programmable mode
- Free-run and restart modes
- Software reset function

23.1 Operation

The clock that feeds the prescaler can be selected from the main clock (divided by 1 or by 16), from the timer input pin (TIN), or from the 32 kHz clock (CLK32). The clock input source is determined by the clock source (CLKSOURCE) field in the timer control registers (TCTL1 and TCTL2). The timer prescaler registers (TPRER1 and TPRER2) are used to select the divide ratio of the input clock that drives the main counter. The prescaler can divide the input clock by a value between 1 and 256. When CLK32 is selected as the clock source, the timer operates even while the PLL is in sleep mode (at that time, the PERCLK1 from the PLL is off).

Each timer can be configured for free-run or restart mode by programming the free-run/restart (FRR) bit in the corresponding TCTLx register. In restart mode, after the compare value is reached, the counter resets to 0x00000000, the compare event (COMP) bit of the timer status register is set, an interrupt is issued if the interrupt request enable (IRQEN) bit of the corresponding TCTLx register is set, and the counter resumes counting. This mode is useful when you need to generate periodic events or audio tones when the free-running timer is used with the timer output signals. In free-run mode, the compare function operates as it does in restart mode, however the counter continues counting without resetting to 0x00000000. When 0xFFFFFFFF is reached, the counter rolls over to 0x00000000 and keeps counting.

Each timer has a 32-bit capture register that takes a “snapshot” of the counter when a defined transition of the timer input (TIN) is detected by the capture edge detector. The type of transition that triggers this capture is selected by the capture edge (CAP) field of the corresponding TCTLx register. Pulses that produce the capture edge can be as short as 20 ns. The minimum time between pulses is two PCLK periods.

When a capture or compare event occurs, the corresponding (CAPT or COMP) status bit is set in the timer status register and an interrupt is posted if the capture function is enabled or if the IRQEN bit of the corresponding TCTLx register is set. The free-running timer is disabled at reset.

When 1 is written to the software reset (SWR) bit in the TCTLx register, the module resets immediately. The reset signal is asserted for three times the period of the 96 MHz SystemCLK cycle. For example, if the system clock period is 10 ns, the reset signal is asserted for 30 ns and then is automatically released.

23.1.1 Pin Configuration for General-Purpose Timers

Figure 23-1 shows the pins used for the General-Purpose Timer module. These pins are multiplexed with other functions on the device, and must be configured for General-Purpose Timer operation. Table 23-1 lists the pin configuration for the General-Purpose Timers.

NOTE:

The user must ensure that the data direction bits in the GPIO are set to the correct direction for proper operation. See Section 29.5.1, “Data Direction Registers,” on page 29-8 for details.

Table 23-1. Pin Configuration

Pin	Setting	Configuration Procedure
TIN	Primary function of GPIO Port A [1]	1. Clear bit 1 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 1 of Port A General Purpose Register (GPR_A)
TMR2OUT	Primary function of GPIO Port D [31]	1. Clear bit 31 of Port D GPIO In Use Register (GIUS_D) 2. Clear bit 31 of Port D General Purpose Register (GPR_D)

23.2 Programming Model

The General-Purpose Timers module includes 6 user-accessible 32-bit registers for each timer. Table 23-2 summarizes these registers and their addresses.

Table 23-2. GP Timers Module Register Memory Map

Description	Name	Address
Timer Control Register 1	TCTL1	00202000
Timer Prescaler Register 1	TPRER1	00202004
Timer Compare Register 1	TCMP1	00202008
Timer Capture Register 1	TCR1	0020200C
Timer Counter Register 1	TCN1	00202010
Timer Status Register 1	TSTAT1	00202014
Timer Control Register 2	TCTL2	00203000
Timer Prescaler Register 2	TPRER2	00203004
Timer Compare Register 2	TCMP2	00203008
Timer Capture Register 2	TCR2	0020300C
Timer Counter Register 2	TCN2	00203010
Timer Status Register 2	TSTAT2	00203014

23.2.1 Timer Control Registers 1 and 2

Each timer control register (TCTL1 or TCTL2) controls the overall operation of the corresponding general-purpose timer. Table 23-3 provides the register description. The TCTLx registers control the following:

- Selecting the free-running or restart mode after a compare event
- Selecting the capture trigger event
- Controlling the output compare mode
- Enabling the compare event interrupt
- Selecting the prescaler clock source
- Enabling and disabling the GP timer

		Addr
TCTL1	Timer Control Register 1	**00202000**
TCTL2	Timer Control Register 2	**00203000**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SWR							FRR	CAP		OM	IRQEN	CLKSOURCE			TEN
TYPE	rw	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 23-3. Timer 1 and 2 Control Registers Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
SWR Bit 15	**Software Reset**—Controls the software reset function. When 1 is written to this bit, a reset signal is generated to the reset timer module. The reset signal is active for 3 system clock cycles, and then is automatically released. The software reset does not reset the timer enable (TEN) bit.	0 = No software reset sent 1 = Software reset sent to timer module
Reserved Bits 14–9	Reserved—These bits are reserved and should read 0.	
FRR Bit 8	**Free-Run/Restart**—Controls how the timer operates after a compare event occurs. In free-run mode, the timer continues running. In restart mode, the counter resets to 0x00000000 and resumes counting.	0 = Restart mode 1 = Free-run mode
CAP Bits 7–6	**Capture Edge**—Controls the operation of the capture function. The direction (DIRx) bit in the corresponding port register must be set to 0 for the capture function to operate correctly.	00 = Disable the capture function 01 = Capture on the rising edge and generate an interrupt 10 = Capture on the falling edge and generate an interrupt 11 = Capture on the rising and falling edges and generate an interrupt
OM Bit 5	**Output Mode**—Controls the output mode of the timer after a reference-compare event occurs. This bit has no function unless the CAP field is set to 0. When the counter value (COUNT) period is less than 40 ns, timer out (TMR2OUT) is not valid.	0 = Active-low pulse for one IPG_CLK period 1 = Toggle output
IRQEN Bit 4	**Interrupt Request Enable**—Enables/Disables the generation of an interrupt on a compare event.	0 = Disable the compare interrupt 1 = Enable the compare interrupt

Table 23-3. Timer 1 and 2 Control Registers Description (continued)

Name	Description	Settings
CLKSOURCE Bits 3–1	**Clock Source**—Selects the source of the clock to the prescaler. The stop-count setting freezes the timer at its current value.	000 = Stop count (clock disabled) 001 = PERCLK1 to prescaler 010 = PERCLK1 ÷16 to prescaler 011 = TIN to prescaler 1xx = 32 kHz clock to prescaler
TEN Bit 0	**Timer Enable**—Enables/Disables the general-purpose timer. The TEN bit can be reset only by a hardware asynchronous reset, not by the SWR reset. **Note:** When configuring this control register, configure all other bits before configuring the TEN bit.	0 = Timer is disabled (counter reset to 0x00000000) 1 = Timer is enabled

23.2.2 Timer Prescaler Registers 1 and 2

Each timer prescaler register (TPRER1 and TPRER2) controls the divide ratio of the associated 8-bit prescaler. The settings for the registers are described in Table 23-4.

		Addr
TPRER1	Timer Prescaler Register 1	**00202004**
TPRER2	Timer Prescaler Register 2	**00203004**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									PRESCALER							
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 23-4. Timer 1 and 2 Prescaler Registers Description

Name	Description	Settings
Reserved Bits 31–8	Reserved—These bits are reserved and should read 0.	
PRESCALER Bits 7–0	**Prescaler**—Determines the division value (1–256) of the prescaler.	0x00 = Divide by 1 0x01 = Divide by 2 ... 0xFF = Divide by 256

23.2.3 Timer Compare Registers 1 and 2

Each timer compare register (TCMP1 and TCMP2) contains the value that is compared with the free-running counter. A compare event is generated when the counter matches the value in this register. This register is set to 0xFFFFFFFF at system reset. Table 23-5 provides the register description.

		Addr
TCMP1	Timer Compare Register 1	**00202008**
TCMP2	Timer Compare Register 2	**00203008**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	COMPARE VALUE															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	0xFFFF															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	COMPARE VALUE															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	0xFFFF															

Table 23-5. Timer 1 and 2 Compare Registers Description

Name	Description
COMPARE VALUE Bits 31–0	**Compare Value**—Holds the value at which a compare event will be triggered.

23.2.4 Timer Capture Registers 1 and 2

Each read-only timer capture register (TCR1 and TCR2) stores the counter value when a capture event occurs. This register is read-only, and resets to 0x00000000. Table 23-6 provides the register description.

		Addr
TCR1	Timer Capture Register 1	**0020200C**
TCR2	Timer Capture Register 2	**0020300C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CAPTURE VALUE															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CAPTURE VALUE															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 23-6. Timer 1 and 2 Capture Registers Description

Name	Description
CAPTURE VALUE Bits 31–0	**Capture Value**—Stores the counter value that existed at the time of the capture event.

23.2.5 Timer Counter Registers 1 and 2

Each read-only timer counter register (TCN1 and TCN2) contains the current count. The timer counter registers can be read at any time without affecting the current count. Table 23-7 describes the register descriptions.

		Addr
TCN1	Timer Counter Register 1	**00202010**
TCN2	Timer Counter Register 2	**00203010**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	COUNT															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	COUNT															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 23-7. Timer 1 and 2 Counter Registers Description

Name	Description
COUNT Bits 31–0	**Counter Value**—Contains the current count value.

23.2.6 Timer Status Registers 1 and 2

Each read-only timer status register (TSTAT1 and TSTAT2) indicates the corresponding timer's status. When a capture event occurs, the CAPT bit is set. When a compare event occurs, the COMP bit is set. These bits must be cleared to clear the interrupt, if it is enabled. These bits are cleared by writing 0x0, write 0 to clear—w0c, and will clear only if they have been read while set. This ensures that an interrupt is not missed if it occurs between the status read and the interrupt clear. The registers and settings are described Table 23-8.

		Addr
TSTAT1	Timer Status Register 1	**00202014**
TSTAT2	Timer Status Register 2	**00203014**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															CAPT	COMP
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r/w0c	r/w0c
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 23-8. Timer 1 and 2 Status Registers Description

Name	Description	Settings
Reserved Bits 31–2	Reserved—These bits are reserved and should read 0.	
CAPT Bit 1	**Capture Event**—Indicates when a capture event occurs.	0 = No capture event occurred 1 = A capture event occurred
COMP Bit 0	**Compare Event**—Indicates when a compare event occurs.	0 = No compare event occurred 1 = A compare event occurred

Chapter 24
Universal Asynchronous Receiver/Transmitters (UART) Modules

24.1 Introduction

This chapter describes the two universal asynchronous receiver/transmitter (UART) modules in the MC9328MXL. The UART modules are capable of standard RS-232 non-return-to-zero (NRZ) encoding format and IrDA-compatible infrared modes. Each UART provides serial communication capability with external devices through an RS-232 cable or through use of external circuitry that converts infrared signals to electrical signals (for reception) or transforms electrical signals to signals that drive an infrared LED (for transmission) to provide low speed IrDA compatibility to the MC9328MXL.

Both UARTs transmit and receive characters that are either 7 or 8 bits in length (program selectable). To transmit, data is written from the peripheral data bus to a 32-byte transmitter FIFO (TxFIFO). This data is passed to the shift register and shifted serially out on the transmitter pin (TXD). To receive, data is received serially from the receiver pin (RXD) and stored in a 32-half-words-deep receiver FIFO (RxFIFO). The received data is retrieved from the RxFIFO on the peripheral data bus. The RxFIFO and TxFIFO generate maskable interrupts as well as DMA Requests when the data level in each of the FIFO reaches a programmed threshold level.

The UARTs generate baud rates based on a configurable divisor and input clock. The UARTs also contain configurable auto baud detection circuitry to receive 1 or 2 stop bits as well as odd, even, or no parity. The receiver detects framing errors, idle conditions, BREAK characters, parity errors, and overrun errors.

The TXD pin is configurable for open-drain operation at the chip boundary. The UART modules use a software interface for control of modem operations and have a serial infrared (IR) module that decodes and encodes IrDA-compatible serial IR data.

24.2 Features

- 7 or 8 data bits
- 1 or 2 stop bits
- Programmable parity (even, odd, and no parity)
- Full 8-wire serial DCE interface[1] for modem flow control on UART 2
- Hardware flow control support for request to send (RTS) and clear to send (CTS) signals
- Software flow control support for data set ready (DSR), data carrier detect (DCD), and ring indicator (RI) signals on UART 2
- Edge selectable RTS and data terminal ready (DTR) edge detect interrupts
- Status flags for various flow control and FIFO states
- Serial IR interface (low speed, IrDA-compatible) enable via UART Control Register 1

1. UART GPIO pins must be used for DTR, DSR, DCD, and RI functions.

- Voting logic for improved noise immunity (16x oversampling)
- Transmitter FIFO empty interrupt suppression
- UART internal clocks enable/disable
- Auto baud rate detection
- Receiver and transmitter enable/disable
- RTS, IrDA asynchronous wake (AIRINT), and receive asynchronous wake (AWAKE) interrupts wake the ARM920T processor from STOP mode
- Twenty maskable interrupts
- Two DMA Requests (TxFIFO DMA Request and RxFIFO DMA Request)
- Escape character sequence detection
- Software reset ($\overline{\text{SRST}}$)

24.2.1 Module Interface

The serial and modem control signals used by the UART module are identified and described in Table 24-1.

Table 24-1. UART Module Interface Signals

Signal Name	I/O	Active	UART	Comments
Serial Signals				
UART1_TXD UART2_TXD	OUT	HIGH	1 and 2	Transmitter serial output (multiplexed IR or NRZ encoding format)
UART1_RXD UART2_RXD	IN	HIGH	1 and 2	Receiver serial input (multiplexed IR or NRZ encoding format)
Modem Control Signals				
$\overline{\text{UART1_RTS}}$ $\overline{\text{UART2_RTS}}$	IN	LOW	1 and 2	Controls the transmitter. By asserting RTS, the modem signals ready to receive to the UART. Normally, the transmitter waits until $\overline{\text{UARTx_RTS}}$ is active (low) before transmitting a character, however when the ignore $\overline{\text{UARTx_RTS}}$ pin (IRTS) bit is set, the transmitter sends each character as it is ready to transmit. When this pin serves as a general purpose input, its status is read in the RTSS bit. This pin can post an interrupt on any transition of this pin and wake the ARM9 core from STOP mode on its assertion. When $\overline{\text{RTS}}$ is negated during a transmission, the UART transmitter finishes transmitting the current character and shuts off. The contents of the TxFIFO (characters to be transmitted) remain undisturbed.
$\overline{\text{UART1_CTS}}$ $\overline{\text{UART2_CTS}}$	OUT	LOW	1 and 2	This output pin serves two purposes. Normally, the receiver indicates that it is ready to receive data by asserting this pin (low). When the receiver detects a pending overrun, it negates this pin. For other applications, this pin functions as a general purpose output controlled by the CTS bit in UART Control Register 2.
$\overline{\text{UART2_DSR}}$	OUT	LOW	2	DSR signal for modem control
$\overline{\text{UART2_RI}}$	OUT	LOW	2	RI signal for modem control
$\overline{\text{UART2_DCD}}$	OUT	LOW	2	DCD signal for modem control

Table 24-1. UART Module Interface Signals (continued)

Signal Name	I/O	Active	UART	Comments
$\overline{\text{UART2_DTR}}$	IN	LOW	2	DTR signal for modem control

24.2.2 UART Pin Configuration

Table 24-2 lists the pins used for the UART 1 and UART 2 modules. These pins are multiplexed with other functions on the device, and must be configured for UART operation.

NOTE:

The user must ensure that the data direction bits in the GPIO are set to the correct direction for proper operation. See Section 29.5.1, “Data Direction Registers,” on page 29-8 for details.

Table 24-2. Pin Configuration

Pin	Setting	Configuration Procedure
UART1_RXD	Primary function of GPIO Port C [12]	1.Clear bit 12 of Port C GPIO In Use Register (GIUS_C) 2.Clear bit 12 of Port C General Purpose Register (GPR_C)
UART1_TXD	Primary function of GPIO Port C [11]	1.Clear bit 11 of Port C GPIO In Use Register (GIUS_C) 2.Clear bit 11 of Port C General Purpose Register (GPR_C)
$\overline{\text{UART1_RTS}}$	Primary function of GPIO Port C [10]	1.Clear bit 10 of Port C GPIO In Use Register (GIUS_C) 2.Clear bit 10 of Port C General Purpose Register (GPR_C)
$\overline{\text{UART1_CTS}}$	Primary function of GPIO Port C [9]	1.Clear bit 9 of Port C GPIO In Use Register (GIUS_C) 2.Clear bit 9 of Port C General Purpose Register (GPR_C)
UART2_RXD	Primary function of GPIO Port B [31]	1.Clear bit 31 of Port B GPIO In Use Register (GIUS_B) 2.Clear bit 31 of Port B General Purpose Register (GPR_B)
UART2_TXD	Primary function of GPIO Port B [30]	1.Clear bit 30 of Port B GPIO In Use Register (GIUS_B) 2.Clear bit 30 of Port B General Purpose Register (GPR_B)
$\overline{\text{UART2_RTS}}$	Primary function of GPIO Port B [29]	1.Clear bit 29 of Port B GPIO In Use Register (GIUS_B) 2.Clear bit 29 of Port B General Purpose Register (GPR_B)
$\overline{\text{UART2_CTS}}$	Primary function of GPIO Port B [28]	1.Clear bit 28 of Port B GPIO In Use Register (GIUS_B) 2.Clear bit 28 of Port B General Purpose Register (GPR_B)
$\overline{\text{UART2_DSR}}$	Alternate function of GPIO Port D [10]	1.Clear bit 10 of Port D GPIO In Use Register (GIUS_D) 2.Set bit 10 of Port D General Purpose Register (GPR_D)
$\overline{\text{UART2_RI}}$	Alternate function of GPIO Port D [9]	1.Clear bit 9 of Port D GPIO In Use Register (GIUS_D) 2.Set bit 9 of Port D General Purpose Register (GPR_D)
$\overline{\text{UART2_DCD}}$	Alternate function of GPIO Port D [8]	1.Clear bit 8 of Port D GPIO In Use Register (GIUS_D) 2.Set bit 8 of Port D General Purpose Register (GPR_D)

Table 24-2. Pin Configuration (continued)

Pin	Setting	Configuration Procedure
$\overline{\text{UART2_DTR}}$	Alternate function of GPIO Port D [7]	1.Set bit 7 of Port D GPIO In Use Register (GIUS_D) 2.Set bit 7 of Port D General Purpose Register (GPR_D)

24.3 Interrupts and DMA Requests

Table 24-3 lists all of the interrupts that are output on each interrupt output. See the individual register descriptions for explanation of available enables and status flags.

Table 24-3. Interrupts and DMA

Interrupt Output	Interrupt Enable	Enable Register Location	Interrupt Flag	Flag Register Location
$\overline{\text{UART_MINT_DTR}}$	DTREN	UCR3_2 (bit 13)	DTRF	USR2_2 (bit 13)
$\overline{\text{UART_MINT_RTS}}$	RTSDEN RTSEN	UCR1_1/UCR1_2 (bit 5) UCR2_1/UCR2_2 (bit 4)	RTSD RTSF	USR1_1/USR1_2 (bit 12) USR2_1/USR2_2 (bit 4)
$\overline{\text{UART_MINT_RX}}$	RRDYEN IDEN DREN RXDSEN	UCR1_1/UCR1_2 (bit 9) UCR1_1/UCR1_2 (bit 12) UCR4_1/UCR4_2 (bit 0) UCR3_1/UCR3_2 (bit 6)	RRDY IDLE RDR RXDS	USR1_1/USR1_2 (bit 9) USR2_1/USR2_2 (bit 12) USR2_1/USR2_2 (bit 0) USR1_1/USR1_2 (bit 6)
$\overline{\text{UART_MINT_TX}}$	TXMPTYEN TRDYEN TCEN	UCR1_1/UCR1_2 (bit 6) UCR1_1/UCR1_2 (bit 13) UCR4_1/UCR4_2 (bit 3)	TXFE TRDY TXDC	USR2_1/USR2_2 (bit 14) USR1_1/USR1_2 (bit 13) USR2_1/USR2_2 (bit 3)
$\overline{\text{UART_MINT_UARTC}}$	OREN BKEN WKEN ADEN ESCI ENIRI AIRINTEN AWAKEN	UCR4_1/UCR4_2 (bit 1) UCR4_1/UCR4_2 (bit 2) UCR4_1/UCR4_2 (bit 7) UCR1_1/UCR1_2 (bit 15) UCR2_1/UCR2_2 (bit 15) UCR4_1/UCR4_2 (bit 8) UCR3_1/UCR3_2 (bit 5) UCR3_1/UCR3_2 (bit 4)	ORE BRCD WAKE ADET ESCF IRINT AIRINT AWAKE	USR2_1/USR2_2 (bit 1) USR2_1/USR2_2 (bit 2) USR2_1/USR2_2 (bit 7) USR2_1/USR2_2 (bit 15) USR1_1/USR1_2 (bit 11) USR2_1/USR2_2 (bit 8) USR1_1/USR1_2 (bit 5) USR1_1/USR1_2 (bit 4)
$\overline{\text{UART_MINT_PFRERR}}$	FRAERREN PARERREN	UCR3_1/UCR3_2 (bit 11) UCR3_1/UCR3_2 (bit 12)	FRAERR PARITYERR	USR1_1/USR1_2 (bit 10) USR1_1/USR1_2 (bit 15)
$\overline{\text{UART_RX_DMAREQ}}$	RDMAEN	UCR1_1/UCR1_2 (bit 8)	RRDY	USR1_1/USR1_2 (bit 9)
$\overline{\text{UART_TX_DMAREQ}}$	TDMAEN	UCR1_1/UCR1_2 (bit 3)	TRDY	USR1_1/USR1_2 (bit 13)

24.4 General UART Definitions

Definitions of terms that occur the following discussions are given in this section.

- **Bit time**—The period of time required to serially transmit or receive 1 bit of data (1 cycle of the baud rate frequency).

- **Start bit**—The bit time of a logic 0 that indicates the beginning of a data frame. A start bit begins with a 1-to-0 transition, and is preceded by at least 1 bit time of logic 1.
- **Stop bit**—1 bit time of logic 1 that indicates the end of a data frame.
- **BREAK**—A frame in which all of the data bits, including the stop bit, are logic 0. This type of frame is usually sent to signal the end of a message or the beginning of a new message.
- **Frame**—A start bit followed by a specified number of data or information bits and terminated by a stop bit. The number of data or information bits depends on the format specified and must be the same for the transmitting device and the receiving device. The most common frame format is 1 start bit followed by 8 data bits (least significant bit first) and terminated by 1 stop bit. An additional stop bit and a parity bit also can be included.
- **Framing error**—An error condition that occurs when the stop bit of a received frame is missing, usually when the frame boundaries in the received bit stream are not synchronized with the receiver bit counter. Framing errors can go undetected if a data bit in the expected stop bit time happens to be a logic 1. A framing error is always present on the receiver side when the transmitter is sending BREAKs. However, when the UART is programmed to expect 2 stop bits and only the first stop bit is received, this is not a framing error by definition.
- **Parity error**—An error condition that occurs when the calculated parity of the received data bits in a frame does not match the parity bit received on the RXD input. Parity error is calculated only after an entire frame is received.
- **Idle**—One in NRZ encoding format and selectable polarity in IrDA mode.
- **Overrun error**—An error condition that occurs when the latest character received is ignored to prevent overwriting a character already present in the UART receive buffer (RxFIFO). An overrun error indicates that the software reading the buffer (RxFIFO) is not keeping up with the actual reception of characters on the RXD input.

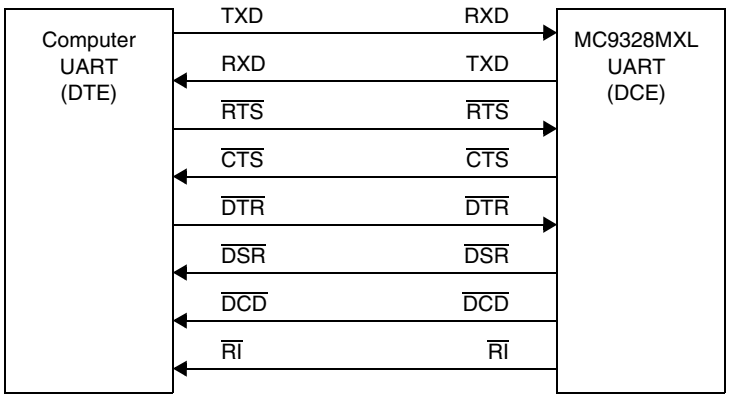


Figure 24-1. General Connections for a UART with a Modem

24.4.1 $\overline{\text{RTS}}$—UART Request To Send

The UART request to send input controls the transmitter. The modem or other terminal equipment signals the UART when it is ready to receive by asserting $\overline{\text{RTS}}$ on the $\overline{\text{UARTx_RTS}}$ pin. Normally, the transmitter waits until this signal is active (low) before transmitting a character, however when the ignore $\overline{\text{UARTx_RTS}}$ pin (IRTS) bit is set, the transmitter sends a character as soon as it is ready to transmit. This pin can post an interrupt on any

transition of this pin and wakes the ARM920T processor from STOP mode on its assertion. When $\overline{RTS}$ is negated during a transmission, the UART transmitter finishes transmitting the current character and shuts off. The contents of the TxFIFO (characters to be transmitted) remain undisturbed.

24.4.2 $\overline{RTS}$ Edge Triggered Interrupt

The input to the $\overline{UARTx_RTS}$ pin can be programmed to generate an interrupt on a selectable edge. The operation of the $\overline{RTS}$ edge triggered interrupt (RTSF) is summarized in Table 24-4.

To enable the $\overline{UARTx_RTS}$ pin to generate an interrupt, set the request to send interrupt enable (RTSEN) bit in the UART Control Register 2 (UCR2_*x*) to 1. Writing 1 to the RTS edge triggered interrupt flag (RTSF) bit in UCR2_*x* clears the interrupt flag.The interrupt can occur on the rising edge, falling edge, or either edge of the $\overline{RTS}$ input. The request to send edge control (RTEC) field in UCR2_*x* programs the edge that generates an interrupt. When RTEC is set to 0x00 and RTSEN = 1, the interrupt occurs on the rising edge (default). When RTEC is set to 0x01 and RTSEN = 1, the interrupt occurs on the falling edge. When RTEC is set to 0x1X and RTSEN = 1, the interrupt occurs on either edge.This is a synchronous interrupt. The RTSF bit is cleared by writing 1 to it. Writing 0 to RTSF has no effect.

Table 24-4. $\overline{RTS}$ Edge Triggered Interrupt Truth Table

RTS	RTSEN	RTEC [1]	RTEC [0]	RTSF	Interrupt Occurs On…	$\overline{UART_MINT_RTS}$
X	0	X	X	0	Interrupt disabled	1
1–>0	1	0	0	0	Rising edge	1
0–>1	1	0	0	1	Rising edge	0
1–>0	1	0	1	1	Falling edge	0
0–>1	1	0	1	0	Falling edge	1
1–>0	1	1	X	1	Either edge	0
0–>1	1	1	X	1	Either edge	0

There is another RTS interrupt that is not programmable, however it asserts the RTS Delta (RTSD) bit when the RTS pin changes state. The status bit RTSD asserts the $\overline{UART_MINT_RTS}$ interrupt when the RTS delta interrupt enable = 1. This is an asynchronous interrupt. The RTSD bit is cleared by writing 1 to it. Writing 0 to the RTSD bit has no effect.

24.4.3 $\overline{DTR}$—Data Terminal Ready

The data terminal ready signal to the $\overline{UART2_DTR}$ indicates the general readiness of the data terminal equipment (DTE). When the connection between the UART and the DTE is established, the DTR signal must remain active throughout the whole connection time. The DTR signal is only available on UART2. In general, the DTR and DSR signals establish the connection and the RTS and CTS signals control the data transfer and the transfer direction (for half-duplex configurations). The DTR signal is like a main switch—when the DTR signal is inactive the RTS and CTS signals have no effect and the modem does not respond to control signals. This functionality is not implemented in the hardware and is the result of how the registers in the UART are programmed.

24.4.4 DTR Edge Triggered Interrupt

The DTR signal can be used to generate an interrupt on a selectable edge. To enable the DTR signal to generate an interrupt, set the data terminal ready interrupt enable (DTREN) bit in UART Control Register 3 (UCR3_*x*). Clear the DTRF bit by writing 1 to it. Writing 0 to the DTRF bit has no effect.

Write to the DTR interrupt edge control (DPEC) field in UCR3_*x* to select the edge that generates an interrupt. When the DPEC field is set to 00b and DTREN = 1, the interrupt occurs on the rising edge (default). When the bits are set to 01b and DTREN = 1, the interrupt occurs on the falling edge. When the bits are set to 1Xb and DTREN = 1, the interrupt occurs on either edge. This is a synchronous interrupt.

Table 24-5. DTR Edge Triggered Interrupt Truth Table

DTR	DTREN	DPEC [1]	DPEC [0]	DTRF	Interrupt Occurs On…	$\overline{\text{UART_MINT_DTR}}$
X	0	X	X	0	Interrupt disabled	1
1–>0	1	0	0	0	Rising edge	1
0–>1	1	0	0	1	Rising edge	0
1–>0	1	0	1	1	Falling edge	0
0–>1	1	0	1	0	Falling edge	1
1–>0	1	1	X	1	Either edge	0
0–>1	1	1	X	1	Either edge	0

24.4.5 DSR—Data Set Ready

The UART uses the DSR signal to inform the DTE that it is powered on, all preparations are complete, and that it is ready to communicate with the DTE.

24.4.6 DCD—Data Carrier Detect

The UART uses this signal to inform the DTE that it detected the carrier signal and the connection is established. This signal remains active as long as the connection remains established.

24.4.7 RI—Ring Indicator

The UART uses this signal to inform the DTE when a ring occurs.

24.4.8 $\overline{\text{CTS}}$—Clear To Send

Normally, the receiver indicates that it is ready to receive data by asserting this pin (low). When the CTS trigger level is programmed to trigger at 32 characters received and the receiver detects the valid start bit of the 33rd character, it deasserts this pin.

24.4.9 Programmable CTS Deassertion

The CTS output can also be programmed to deassert when the RxFIFO reaches a certain level. Setting the CTS trigger level at any value less than 32 deasserts the CTS pin on detection of the valid start bit of the N + 1 character (where N is the trigger level setting). The receiver continues to receive characters until the RxFIFO is full.

24.4.10 TXD—UART Transmit

This is the transmitter serial output. When operating in normal mode, NRZ encoded data is output. When operating in infrared mode, a 3/16 bit-period pulse is output for each 0 bit transmitted, and no pulse is output for each 1 bit transmitted. For RS-232 applications, this pin must be connected to an RS-232 transmitter.

24.4.11 RXD—UART Receive

This is the receiver serial input. When operating in normal mode, NRZ encoded data is expected. When operating in infrared mode, a narrow pulse is expected for each 0 bit received and no pulse is expected for each 1 bit received. External circuitry must convert the IR signal to an electrical signal. RS-232 applications require an external RS-232 receiver to convert voltage levels.

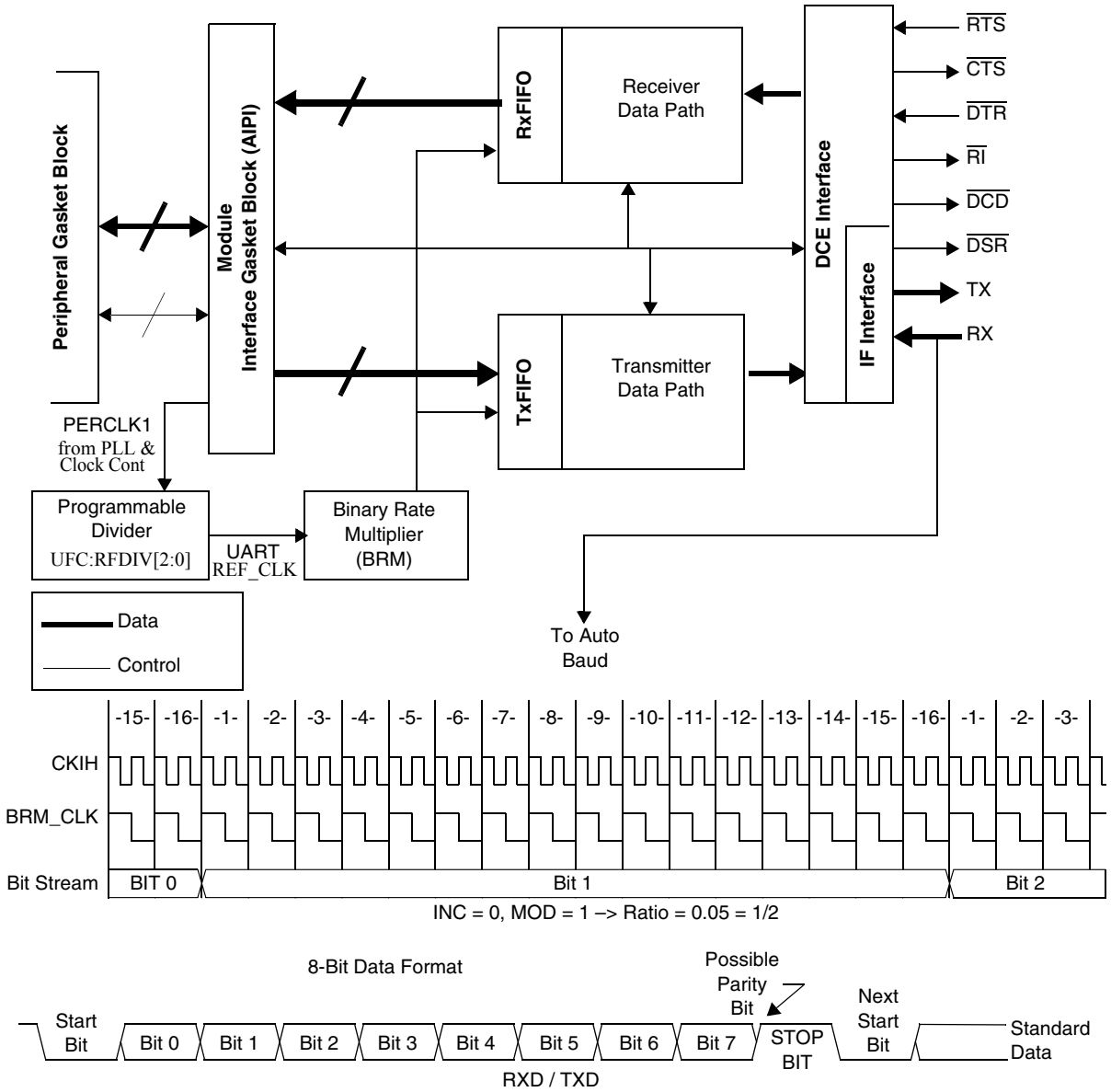


Figure 24-2. UART Block Diagram and Clock Generation Diagram

24.5 Sub-Block Description

The UART module is easy to use from both hardware and software perspectives. The working registers shown in Table 24-6 provide all status (USR1_*x*, USR2_*x*) and control (UCR1_*x* through UCR4_*x*) functions. A separate test register is provided for applications that require it. The binary rate multiplier (BRM) registers (UBIR_*x*, UBMR_*x*, BMPR1_*x*, and BMPR4_*x*) control the UART bit rate and there is a transmitter register and a receiver register all shown in Table 24-6.

Table 24-6. UART Register Summary

UART	Status Registers	Control Registers	BRM Registers	BRC Registers	Esc Character Registers	Transmitter Registers	Receiver Registers
1	USR1_1 USR2_1	UCR1_1 UCR2_1 UCR3_1 UCR4_1	UBIR_1 UBMR_1 BMPR1_1 BMPR4_1	UBRC_1	UESC_1 UTIM_1	UTXD_1	URXD_1
2	USR1_2 USR2_2	UCR1_2 UCR2_2 UCR3_2 UCR4_2	UBIR_2 UBMR_2 BMPR1_2 BMPR4_2	UBRC_2	UESC_2 UTIM_2	UTXD_2	URXD_2

The transmit and receive registers are optimized for a 32-bit bus. All status bits associated with the received data are accessible along with the data in a single read. Except for the transmit data (TX_DATA) field in the UART Transmitter Registers, all register bits are readable and most are read/write. The UART Baud Rate Count (BRC) Register performs automatic baud rate detection. There are also two registers for the escape sequence detection, the UART Escape Character Register (UESC_*x*) and the UART Escape Timer Register (UTIM_*x*). The following sections describe the basic functionality of the major blocks in UART module.

The MC9328MXL only supports a 16 MHz reference frequency. The reference frequency is defined as the input peripheral clock, PERCLK1, divided by the value of the RFDIV [2:0] bits in the UFCR. Also note that the frequency of the system/CPU clock (HCLK/BCLK) must be greater than the UART reference frequency. For example, if the UART reference frequency is 16 MHz, BCLK must be greater than 16 MHz. If the BCLK frequency is not greater, some of the UART features may not operate properly. For more details on BLCK please refer to Chapter 12, “Phase-Locked Loop and Clock Controller.”

It is recommended to use a reference frequency of 16 MHz. Since the default (a maximum) System PLL frequency is 96 MHz, 16 MHz is an integer multiple of 96 MHz. The accuracy of the reference frequency is needed to accurately determine auto baud frequencies and break detect. The System PLL default setting of 96 Mhz also assumes the use of a 32 kHz crystal. If a different crystal frequency is used (such as a 32.768 kHz crystal), the PLL would need to be re-programmed for 96 MHz in order for an accurate 16 MHz ref frequency to be produced. Although reference frequencies of 25 MHz and 30 MHz can also be used to determine features such as auto baud detection and break detect, these are not integer multiples of the default (and maximum) System PLL of 96 MHz.

24.5.1 Transmitter

The transmitter accepts a parallel character from the ARM920T processor and transmits it serially. The start, stop, and parity (when enabled) bits are added to the character. When the ignore RTS bit (IRTS) is set, the transmitter sends a character as soon as it is ready to transmit. RTS can be used to provide flow-control of the serial data. When RTS is negated (high), the transmitter finishes sending the character in progress (if any), stops, and waits for $\overline{\text{RTS}}$ to be asserted (low) again. Generation of BREAK characters and parity errors (for debugging purposes) is supported. The transmitter operates from the 1x clock provided by the BRM. Normal NRZ encoded data is transmitted when the IR interface is disabled.

The transmitter FIFO (TxFIFO) contains 32 bytes. The data is written to TxFIFO by writing to the appropriate register (UTXnD_*x*) with the byte data to the [7:0] bits. The TxFIFO is addressed using UTXnD_*x* register (depending on if you are using UART 1 or 2) with any one of the 16 addresses (0x00212040–0x0021207C) and the

data is written consecutively if the TxFIFO is not full or is read consecutively if the TxFIFO is not empty. If the TxFIFO is full and data is again attempted to be written to the FIFO, the overrun bit will be set and data cannot be written unless a read is first performed.

24.5.2 Transmitter FIFO Empty Interrupt Suppression

The transmitter FIFO empty interrupt suppression logic suppresses the interrupt between writes to the TxFIFO. When a character is written to the TxFIFO, it is immediately transferred to the transmitter shift register (PISO_OUT) on the next transmit baud rate clock (when the transmitter is enabled). The suppression logic allows the software to write another character to the TxFIFO before the interrupt is asserted. When the transmitter shift register empties before another character is written to the TxFIFO, the interrupt is asserted. Writing data (even a single character) to the TxFIFO releases the interrupt. The interrupt is asserted on the following conditions:

- System reset
- UART module reset
- When a single character has been written to Transmitter FIFO and then the Transmitter FIFO and the Transmitter Shift Register become empty until another character is written to the Transmitter FIFO.
- The last character in the TxFIFO is transferred to the shift register, when TxFIFO contains two or more characters. See Figure 24-3 on page 24-12.

To provide system bus bandwidth efficiency when filling the transmit FIFO, there is a threshold called Transmitter Trigger Level control (TXTL) which allows a maskable interrupt to be generated whenever the data level in the TxFIFO falls below the selected threshold. The following conditions are required to avoid the TxFIFO being overwritten, thus affecting parity calculation and data pattern transmission.

1. Maximum 32 bytes can be written into TxFIFO when transmission is completed.

   USR2 bit[3] TXDC = 1, requires the use of polling or interrupt service routine via source UART_MINT_TX with UCR1/TXEMPTYEN = 0, UCR1/TRDYEN = 0 and UCR4/TCEN = 1

2. Maximum 31 bytes can be written into TxFIFO when the FIFO is empty.

   USR2 bit[14] TXFE = 1, requires the use of polling or interrupt service routine via source UART_MINT_TX with UCR1/TXEMPTYEN = 1, UCR1/TRDYEN = 0 and UCR4/TCEN = 0

3. Write 32-TXTL bytes into TxFIFO when data level in TxFIFO falls below the selected threshold TXTL.

   USR1 bit[13] TRDY = 1, scenario where DMA is used with UCR1/TDMAEN = 1

Table 24-7. Number of Characters Allowed to be Written into TxFIFO

USR2/TXDC		USR2/TXFE		USR1/TRDY	
1	1	1	32	–	–
0	1	1	31	–	–
0	0	1	32–n	–	–

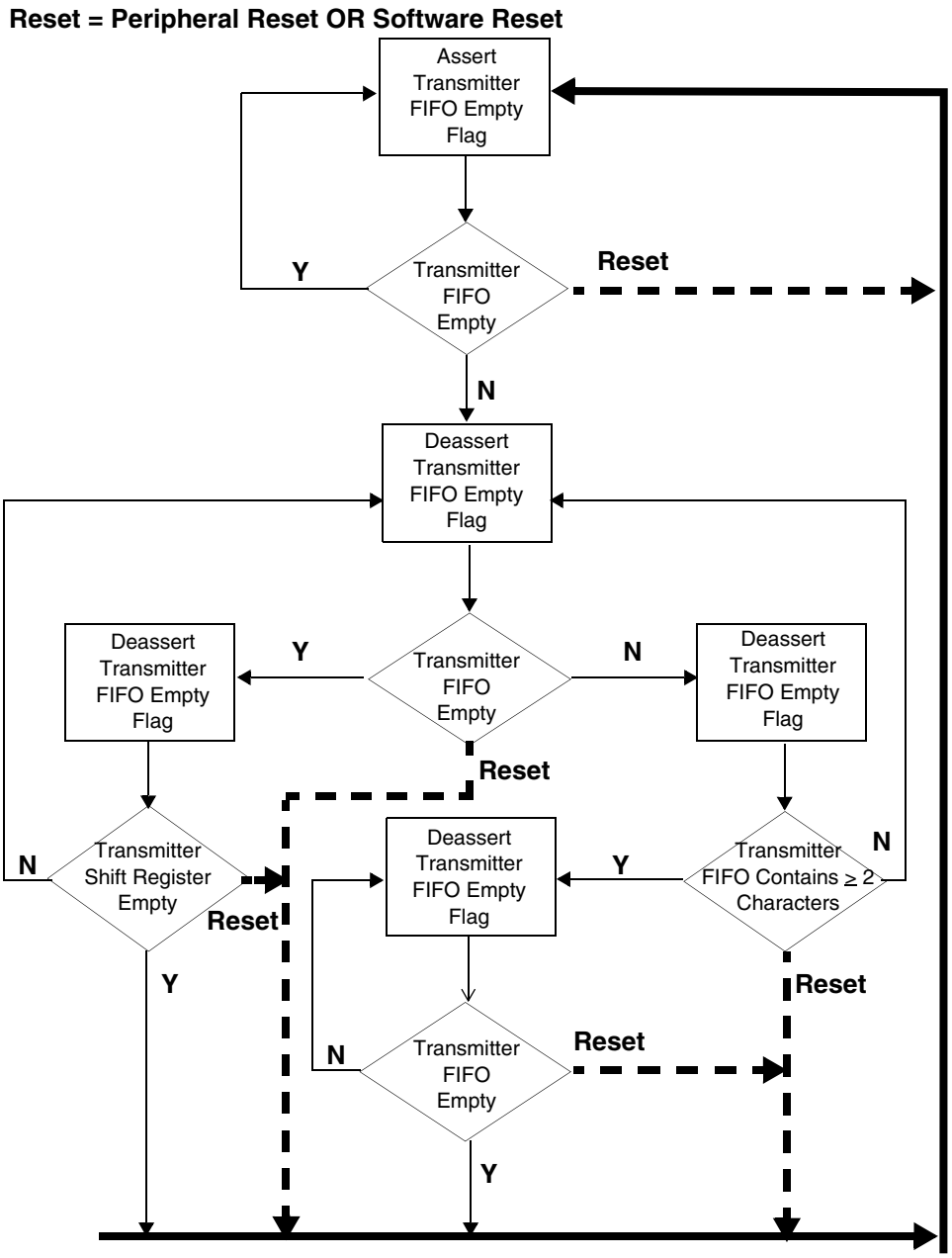


Figure 24-3. Transmitter FIFO Empty Interrupt Suppression Flow Chart

24.5.3 Receiver

The receiver accepts a serial data stream and converts it into parallel characters. When enabled, it searches for a start bit, qualifies it, and samples the following data bits at the bit-center. Jitter tolerance and noise immunity are provided by sampling at a 16x rate and using voting techniques to clean up the samples. Once the start bit is found, the data bits, parity bit (if enabled), and stop bits (either 1 or 2 depending on user selection) are shifted in. Parity is checked and its status reported in the appropriate register (either URXDn_1 or 2) when parity is enabled. Frame errors and BREAKs are also checked and reported. When a new character is ready to be received by the RxFIFO, the receive data ready (RDR) bit in the UART Status Register 2 (USR2_*x*) is asserted and an interrupt is posted (if

DREN = 1). If the receiver trigger level is set to 0, the receiver ready interrupt flag (RRDY) is asserted and an interrupt is posted if the receiver ready interrupt enable bit is set (RRDYEN = 1). If any UART Receiver Register (URXD*n_x*) is read as a word and there is only 1 character in the RxFIFO, the interrupt generated by the RRDY bit is automatically cleared and the data along with 4 status bits are read by the ARM920T processor (see the bit descriptions in Section 24.7.8, "UART Status Register 1," on page 24-40 and Section 24.7.9, "UART Status Register 2," on page 24-42). The RRDY bit is cleared when the data in the RxFIFO falls below the programmed trigger level.

Normal NRZ encoded data is expected when the IR interface is disabled.

The RxFIFO contains 32 half-words. The data is read from the RxFIFO by reading the half-word data in the [15:0] bits in the appropriate register for the UART being used (URXD*n_x*). The RxFIFO is addressed using the aforementioned register with any one of the 16 addresses and the data is written consecutively if the RxFIFO is not full, or is read consecutively if the RxFIFO is not empty. When additional data is written to the RxFIFO while it is full, the write operation cannot complete unless a read is performed. If a write is performed on the RxFIFO when it is full, the ORE bit is set in the appropriate USR2_*x* register. The ORE bit is cleared by writing 1 to it.

24.5.4 Idle Line Detect

The receiver logic block includes the ability to detect an idle line. Idle lines indicate the end or the beginning of a message. For an idle condition to occur, there must be at least 1 word in the RxFIFO and the RXD pin must be idle for more than a configured number of frames.

When the idle condition detected interrupt enable (IDEN) bit in the UART Control Register 1 (UCR1_*x*) is set and the line is idle for 4 (default), 8, 16, or 32 (maximum) frames, the detection of an idle condition flags an interrupt. When an idle condition is detected, the IDLE bit in the appropriate register (USR2_*x*) is set. Clear the IDLE bit by writing 1 to it. Writing 0 to the IDLE bit has no effect.

24.5.4.1 Idle Condition Detect Configuration

The idle condition detect (ICD [1:0]) field is located in the UART Control Register 1. If the bits are set to 00b, RXD must be idle for more than 4 frames before the IDLE bit is asserted. If the bits are set to 01b, RXD must be idle for more than 8 frames before the IDLE bit is asserted. If the bits are set to 10b, RXD must be idle for more than 16 frames before the IDLE bit is asserted. If the bits are set to 11b, RXD must be idle for more than 32 frames before the IDLE bit is asserted (see Table 24-8).

Table 24-8. IDLE Detection Truth Table

IDEN	ICD [1]	ICD [0]	IDLE	$\overline{\text{UART_MINT_RX}}$
0	X	X	0	1
1	0	0	asserted after 4 idle frames	asserted after 4 idle frames
1	0	1	asserted after 8 idle frames	asserted after 8 idle frames
1	1	0	asserted after 16 idle frames	asserted after 16 idle frames
1	1	1	asserted after 32 idle frames	asserted after 32 idle frames
Note: This table assumes that no other interrupt is set at the same time this interrupt is set for the $\overline{\text{MINT_RX}}$ signal. This table shows how this interrupt affects the $\overline{\text{MINT_RX}}$ signal.				

During a normal message there is no idle time between frames. When all of the information bits in a frame are logic 1s, the start bit ensures that at least one logic 0 bit time occurs for each frame so that the IDLE bit is not asserted.

24.5.5 Receiver Wake

The receiver logic block includes the WAKE bit in the USR2_*x* register that is set when the receiver detects the start bit. The WAKE bit is not set until the start bit is qualified. When the wake interrupt enable (WKEN) bit is enabled, the receiver flags an interrupt (UART_MINT_UARTC) if the WAKE status bit is set. The WAKE bit is cleared by writing 1 to it. Writing 0 to the WAKE bit has no effect.

When the asynchronous wake interrupt (AWAKE) is enabled (AWAKEN = 1) and the ARM920T processor is in STOP mode, a falling edge detected on the receive pin asserts the AWAKE bit and the UART_MINT_UARTC interrupt to wake the ARM920T processor from STOP mode. Clear the AWAKE bit by writing 1 to it. Writing 0 to the AWAKE bit has no effect.

If the asynchronous IR WAKE interrupt is enabled (AIRINTEN = 1), the UART is configured for IR mode, the ARM920T processor is in STOP mode, and a falling edge is detected on the receive pin, this asserts the AIRINT bit and the UART_MINT_UARTC interrupt and wakes the ARM920T processor from STOP mode. Clear the AIRINT bit by writing 1 to it. Writing 0 to the AIRINT bit has no effect.

The recommended procedure for programming the asynchronous interrupts is to first clear them by writing 1 to the appropriate bit in the UART Status Register 1 (USR1_*x*). Poll or enable the interrupt for the Receiver IDLE Interrupt Flag (RXDS) in the appropriate register (USR1_*x*). When asserted, the RXDS bit indicates to the software that the receiver state machine is in the idle state, the next state is idle, and the RXD pin is idle (high). After following this procedure, enable the asynchronous interrupt and enter STOP mode.

24.5.6 Receiving a BREAK Condition

A BREAK condition is received when the receiver detects all 0s (including a 0 during the bit time of the stop bit) in a frame. The BREAK condition asserts the BRCD bit in the appropriate register (USR2_*x*) and writes only the first BREAK character to the RxFIFO. Clear the BRCD bit by writing 1 to it. Writing 0 to the BRCD bit has no effect. The BRCD bit remains asserted as long as a BREAK condition exists. The BREAK condition ends when the receiver detects a 0-to-1 transition.

24.5.7 Vote Logic

The vote logic block provides jitter tolerance and noise immunity by sampling with respect to VOTE_CLK and using voting techniques to clean up the samples. The voting is implemented by sampling the incoming signal constantly on the rising edge of VOTE_CLK. The receiver is provided with the majority vote value, which is 2 out of the 3 samples. Examples of the majority vote results of the vote logic are shown in Table 24-9.

Table 24-9. Majority Vote Results

Samples	Vote
000	0
101	1
001	0
111	1

The vote logic captures a sample on every rising edge of BRM_CLK, however the receiver uses 16x oversampling to take its value in the middle of the sample frame. The idle character may be longer or shorter than 16 counts, however the receiver looks for a 1-to-0 transition. The receiver starts to count when the Start bit is set however it does not capture the contents of the RxFIFO at the time the Start bit is set. The start bit is validated when 0s are received for 7 consecutive bit times following the 1-to-0 transition. Once the counter reaches 0xF, it starts counting on the next bit and captures it in the middle of the sampling frame (see Figure 24-4). All data bits are captured in the same manner. Once the stop bit is detected, the receiver shift register (SIPO_OUT) data is parallel shifted to the RxFIFO.

There is a special case when the BRM_CLK period is longer than the period of the IR 0 pulse (when the baud rate is configured for less than 31.25 Kbps). In this case, the software sets the IRSC bit so that the MC9328MXL reference clock is the clock for the voting logic. The pulse is validated by counting the length of the pulse. This logic only works with 16 MHz, 25 MHz, and 30 MHz reference clock frequencies. Enabling this bit with any other reference frequency is undefined. When setting IRSC = 1, either:

- Ignore the first character received, clear the status flags after the RDR bit is set, and proceed,
- OR enables the software reset of the UART module after the initial configuration of UART including setting of the reference frequency bit (Ref16/Ref25/Ref30), however before setting IRSC bit to high. Using this method, insures the first character received is correct.

24.5.8 Binary Rate Multiplier (BRM)

The BRM submodule generates all baud rates that required integer and non-integer division. The input and output frequency ratio is programmed in the UART BRM Incremental Register (UBIR_*x*) and UART BRM Modulator Register (UBMR_*x*). The output frequency is divided by the input frequency to produce this ratio. For integer division, set the appropriate UBIR_*x* register to equal 0x000F and write the divisor to the UBMR_*x* register. All values written to these registers must be one less than the actual value to eliminate division by 0 (undefined), and to increase the maximum range of the registers.

Updating the BRM registers requires writing to both registers. The UBIR_*x* register must be written before writing to the UBMR_*x* register. If only one register is written to by the software, the BRM continues to use the previous values.

The following examples show how to determine what values are to be programmed into UBIR and UBMR for a given reference frequency and desired baud rate. The following equation can be used to help determine these values:

$$[(\text{Desired Baud Rate})*16]/(\text{reference frequency})=\text{NUM}/\text{DENOM} \qquad \textit{Eqn. 24-1}$$

$$\text{UBIR} = \text{NUM} - 1 \qquad \textit{Eqn. 24-2}$$

$$\text{UBMR} = \text{DENOM} - 1 \qquad \textit{Eqn. 24-3}$$

$$\text{reference frequency} = \text{PERCLK1} / \text{RFDIV}\ [2:0] \qquad \textit{Eqn. 24-4}$$

Example: Integer Division ÷ 2

```
Reference Frequency = 19.44 MHz
NUM = 0x0000
DENOM = 0x0001
UBIR = NUM - 1
UBMR = DENOM - 1
```

NOTE:

Notice each value written to the registers is one less than the actual value.

Example: Non-integer Division

```
Reference Frequency = 16 MHz
Output Frequency = 920 kbps × 16 (sampling) = 14.72 MHz
Ratio --> 14.72 ÷ 16 = 0.92
NUM = 92 (decimal) = 0x5C
DENOM = 100 (decimal) = 0x64
UBIR = NUM - 1
UBMR = DENOM - 1
```

NOTE:

The ratio is derived directly from the division with no factoring (easiest). To derive the exact ratio, some factoring must be performed. Factoring the above ratio produces:

```
Factored Ratio: 23 ÷ 25
NUM = 22 (decimal) = 0x0016
DENOM = 25 (decimal) = 0x0019
```

Example: Non-integer Division

```
Reference Frequency = 25 MHz
Output Frequency = 920 kbps × 16 (sampling) = 14.72 MHz
Ratio --> 14.72 ÷ 25 = 0.5888
NUM = 5888 (decimal) = 0x1700
DENOM = 10000 (decimal) = 0x2710
UBIR = NUM - 1
UBMR = DENOM - 1
```

NOTE:

The ratio is derived directly from the division with no factoring (easiest). To derive the exact ratio, some factoring must be performed. Factoring the above ratio produces:

```
Factored Ratio = 184 ÷ 313
NUM = 183 (decimal) = 0x00B7
DENOM = 313 (decimal) = 0x0139
```

Example: Non-integer Division:

```
Reference Frequency: 30 MHz
Output Frequency: 920 kbps × 16 (sampling) = 14.72 MHz
Ratio --> 14.72 ÷ 30 = 0.4907
NUM = 4907 (decimal) = 0x132B
DENOM = 10000 (decimal) = 0x2710
```

NOTE:

The ratio is derived directly from the division with no factoring (easiest). To derive the exact ratio, some factoring must be perform. Factoring the above ratio produces:

```
Factored Ratio: 184 ÷ 325
```

```
NUM = 183 (decimal) = 0x00B7
DENOM = 325 (decimal) = 0x0144

Baud Rate = (CKIH) ÷ (16 × Divisor)
or
Divisor = (CKIH) ÷ (16 × Baud Rate)
Transmitter Clock = 16 × Baud Rate
Receiver Clock = CKIH ÷ Divisor = 16 × Baud Rate
```

24.5.9 Baud Rate Automatic Detection Logic

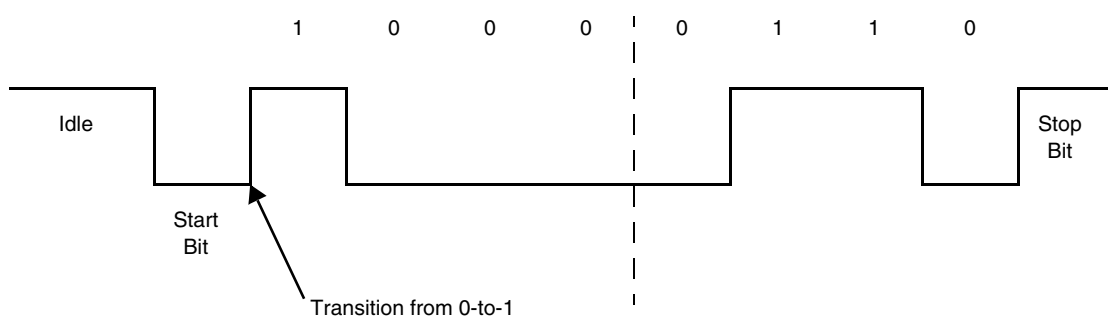
When the baud rate automatic detection logic is enabled, the UART locks onto the incoming baud rate. To enable this feature, set the automatic detection of baud rate bit (ADBR = 1) in the UART Control Register 1 and write 1 to the ADET bit in the UART Status Register 2 to clear it. When ADET=0 and ADBR=1, the UART automatically sets the INC=0x0 in the appropriate UBIR_*x* register and MOD =0x0 in the UBMR_*x* register. The UART waits for the start bit (transition from 1-to-0) and tries to lock onto the incoming baud rate. Once the start bit is detected, the length of the start bit is calculated by counting until the 0-to-1 transition (see Figure 24-4 and Section 24.5.9.1). The new baud rate is determined using this equation:

Eqn. 24-5

$$BaudRate = \langle \frac{Count}{16} \rangle$$

Table 24-10. Baud Rate Automatic Detection

ADBR	ADET	Baud Rate Detection	$\overline{\text{UART_MINT_UARTC}}$
0	X	Manual Configuration	1
1	0	Auto Detection	1
1	1	Auto Detection Complete	0
Note: This table assumes that no other interrupt is set at the same time this interrupt is set for the $\overline{\text{UART_MINT_UARTC}}$ signal.			



Note: LSB transmitted first.

Figure 24-4. Baud Rate Detection Protocol Diagram

The BRM Incremental Preset registers (BIPR1_*x* through BIPR4_*x*) and BRM Modulator Preset registers (BMPR1_*x* through BMPR4_*x*) are used when detecting the special baud rates (when BPEN = 1). These registers are written by software before the start of automatic baud sequence detection. After the auto baud

count value is divided by 16 and a remainder higher than 3 is detected, the appropriate BIPR*n_x* and BMPR*n_x* registers are selected (if the BPEN bit is asserted). This feature is available for 16 MHz, 25 MHz, or 30 MHz reference frequencies only. Enabling this feature with any other reference frequency is not supported and is undefined. The corresponding reference frequency bits in the control register (they are in different registers) must also be set (REF16, REF25, or REF30).

When the BPEN bit is not asserted (BPEN = 0), non-integer division is performed for all detections. This method works with any reference frequency. Based on the reference frequency, the UART computes the baud rate on the fly.

If any of the UART BRM registers are written to simultaneously by the baud rate automatic detection logic and peripheral data bus, the peripheral data bus has priority.

24.5.9.1 Baud Rate Automatic Detection Protocol

The receiver must receive an ASCII character “A” or “a” to verify proper detection of the incoming baud rate. When an ASCII character “A” or “a” is received and no error occurs, the Automatic Detect baud rate bit is set (ADET=1) and if the interrupt is enabled (ADEN=1),an interrupt $\overline{\text{UART_MINT_UARTC}}$ is generated.

When an ASCII character “A” or “a” is not received (because of a bit error or the transmission of another character), the value written into the associated UBIR_*x* and UBMR_*x* registers may not be accurate. When the ADET bit is not asserted after the required time-out value (based on baud rate, word size, parity, and number of stop bits), look out for the parity/frame error interrupt ($\overline{\text{UART_MINT_PFERR}}$ = 0) if enabled. After the interrupt is asserted, re-send the character “A” or “a” and repeat the above procedure until the ADET bit is set.

As long as ADET = 0 and ADBR = 1, the UART continues to try to lock onto the incoming baud rate. Once the ASCII character “A” or “a” is detected and the ADET bit is set, the receiver ignores the ADBR bit and continues normal operation with the calculated baud rate divisor.

The UART interrupt is active ($\overline{\text{UART_MINT_UARTC}}$ = 0) as long as ADET = 1 and ADBR = 1. This can be disabled by clearing the automatic baud rate detection interrupt enable bit (ADEN = 0). Before starting an automatic baud rate detection sequence, set ADET = 0 and ADBR = 1.

The RxFIFO must contain the ASCII character “A” or “a” following the automatic baud rate detection interrupt, which can be cleared by reading it. Because this UART has standard reference frequencies of 16 MHz, 25 MHz, or 30 MHz, the highest achievable baud rate is determined by the value of the reference frequency ÷ 16.

Table 24-11. Highest Baud Rates

Reference Frequency (MHz)	Highest Baud Rate (bps)
16 MHz	1 Mbps
25 MHz	1.5625 Mbps
30 MHz	1.875 Mbps

The 16-bit UART Baud Rate Count Register (UBRC_*x*) is reset to 8 and stays at 0xFFFF when an overflow occurs. The appropriate UBRC_*x* register counts the start bit of the incoming baud rate. When the start bit is detected and counted, the UART Baud Rate Count Register retains its value until the next automatic baud rate detection sequence is initiated.

The read only Baud Rate Count Register counts only when auto detection is enabled.

24.5.10 Escape Sequence Detection

An escape sequence typically consists of 3 characters entered in rapid succession (such as +++). Because these are valid characters by themselves, the time between characters determines if it is a valid escape sequence. Too much time between two of the + characters is interpreted as two + characters, and not part of an escape sequence.

The software chooses the escape character and writes its value to the associated UART Escape Character Register (UESC_*x*). The hardware compares this value to incoming characters in the RxFIFO. When an escape character is detected, the internal escape timer starts to count. The software specifies a time-out value for the maximum allowable time between escape characters. The escape timer is programmable in intervals of 2 msec to a maximum interval of 8.192 seconds.

Table 24-12. Escape Timer Scaling

UTIM_1/ UTIM_2 Register	Maximum Time Between Specified Escape Characters
0x000	2 msec
0x001	4 msec
0x002	6 msec
0x003	8 msec
0x004	10 msec
…	…
0F8	498 msec
0F9	500 msec
…	…
9C3	5 sec
…	…
FFD	8.188 sec
FFE	8.190 sec
FFF	8.192 sec
Note: To calculate the time interval: (UTIM_Value + 1) × 0.002 = Time_Interval **Example:** (09C3 + 1) × 0.002 = 5 sec	

The escape sequence detection feature is available for 16 MHz, 25 MHz, or 30 MHz reference frequencies only. Enabling this feature with any other reference frequency is not supported and is undefined.Set the corresponding reference frequency bits (REF16 in the UCR4_*x* register, REF25 or REF30 in UCR3_*x* register).

The escape sequence detection feature asserts the escape sequence interrupt flag (ESCF) bit when the escape sequence interrupt enable (ESCI) bit is set and an escape sequence is detected. Clear the ESCF bit by writing 1 to it. Writing 0 to the ESCF bit has no effect.

24.6 Infrared Interface

The IR interface converts data to be transmitted or received as specified in the IRDA Serial Infrared Physical Layer Specification.

For each 0 to be transmitted, a narrow positive pulse (3/16 of a bit time) is generated. For each 1 to be transmitted, no pulse is generated (output is low). The data is muxed out onto the TXD pin when the infrared interface enable bit is set (IREN = 1). External circuitry must be provided to drive an infrared LED.

When receiving, a narrow negative pulse is expected for each 1 transmitted and no pulse is expected for each 0 transmitted (input is high). The data from the RXD pin is demuxed to the appropriate IR decoder logic when the infrared interface enable bit is set (IREN = 1). Circuitry external to the MC9328MXL transforms the IR signal into an electrical signal.

Serial infrared mode (SIR) uses an edge triggered interrupt flag (the serial infrared interrupt flag, IRINT, in the associated USR2_*x* register) that validates 0 bits as they are received. When INVR =0, detection of a falling edge on the UART_RXD pin assets the IRINT bit. When INVR=1, detection of a rising edge on the UART_RXD pin assets the IRINT bit. When both the IRINT and ENIRI bits are asserted, the UART_MINT_UARTC interrupt is asserted. Clear the IRINT bit by writing 1 to it. Writing 0 to the IRINT bit has no effect.

In many applications, the ability to detect IrDA pulses < 2µs (micro-seconds) has caused data to be misread or missed all together. The work around to this is to allow enough oversampling of the UART clock when receiving narrow IR pulses < 2µs by configuring PerCLK1 greater than the selected reference frequency of 16 MHZ, 25 MHz, or 30 MHz as selected in the UART registers UCR3 and UCR4. The UART module will think that it is using a certain reference frequency, however, in fact it will be over driven by a higher frequency PerCLK1. However, caution must be used when employing this method as this will render features that are dependant on an accurate reference frequencies useless, such as the auto baud detection and break detection. Also, baud rates will need to be re-calculated based on the PerCLK1 clock frequency. An example suggestion is to simply set PerCLK1 to 32 MHz, while setting the UART reference frequency to 16 MHz (make sure that the RFDIV bits are set to divide by one in the UFCR register and REF16 is set in the UCR4 register). This oversampling ensures that IR pulses < 2µs will be detected. Also, make sure to remember to re-calculate the baud rates based on the value of 32 MHz and not 16 MHz when using this example.

24.7 Programming Model

The two UART modules include42 user-accessible 32-bit registers and4 user-accessible 16-bit registers. All registers use word, halfword, or byte access. The UART Receiver Registers and the UART Transmitter Registers are each mapped to 16 word-length addresses to support the load multiple registers (LDM) instruction and the store multiple registers (STM) instruction, respectively

Table 24-13 summarizes these registers and their addresses.

Table 24-13. UART Module Register Memory Map

Description	Name	Address
UART 1		
UART1 Receiver Register n	URXDn_1	0x00206000+4*n
UART1 Transmitter Register n	UTXnD_1	0x00206040+4*n
UART1 Control Register 1	UCR1_1	0x00206080
UART1 Control Register 2	UCR2_1	0x00206084
UART1 Control Register 3	UCR3_1	0x00206088
UART1 Control Register 4	UCR4_1	0x0020608C
UART1 FIFO Control Register	UFCR_1	0x00206090
UART1 Status Register 1	USR1_1	0x00206094
UART1 Status Register 2	USR2_1	0x00206098
UART1 Escape Character Register	UESC_1	0x0020609C
UART1 Escape Timer Register	UTIM_1	0x002060A0
UART1 BRM Incremental Register	UBIR_1	0x002060A4
UART1 BRM Modulator Register	UBMR_1	0x002060A8
UART1 Baud Rate Count Register	UBRC_1	0x002060AC
UART1 BRM Incremental Preset Register 1	BIPR1_1	0x002060B0
UART1 BRM Incremental Preset Register 2	BIPR2_1	0x002060B4
UART1 BRM Incremental Preset Register 3	BIPR3_1	0x002060B8
UART1 BRM Incremental Preset Register 4	BIPR4_1	0x002060BC
UART1 BRM Modulator Preset Register 1	BMPR1_1	0x002060C0
UART1 BRM Modulator Preset Register 2	BMPR2_1	0x002060C4
UART1 BRM Modulator Preset Register 3	BMPR3_1	0x002060C8
UART1 BRM Modulator Preset Register 4	BMPR4_1	0x002060CC
UART1 Test Register 1	UTS_1	0x002060D0
UART 2		
UART2 Receiver Register n	URXDn_2	0x00207000+4*n
UART2 Transmitter Register n	UTXnD_2	0x00207040+4*n
UART2 Control Register 1	UCR1_2	0x00207080
UART2 Control Register 2	UCR2_2	0x00207084
UART2 Control Register 3	UCR3_2	0x00207088

Table 24-13. UART Module Register Memory Map (continued)

Description	Name	Address
UART2 Control Register 4	UCR4_2	0x0020708C
UART2 FIFO Control Register	UFCR_2	0x00207090
UART2 Status Register 1	USR1_2	0x00207094
UART2 Status Register 2	USR2_2	0x00207098
UART2 Escape Character Register	UESC_2	0x0020709C
UART2 Escape Timer Register	UTIM_2	0x002070A0
UART2 BRM Incremental Register	UBIR_2	0x002070A4
UART2 BRM Modulator Register	UBMR_2	0x002070A8
UART2 Baud Rate Count Register	UBRC_2	0x002070AC
UART2 BRM Incremental Preset Register 1	BIPR1_2	0x002070B0
UART2 BRM Incremental Preset Register 2	BIPR2_2	0x002070B4
UART2 BRM Incremental Preset Register 3	BIPR3_2	0x002070B8
UART2 BRM Incremental Preset Register 4	BIPR4_2	0x002070BC
UART2 BRM Modulator Preset Register 1	BMPR1_2	0x002070C0
UART2 BRM Modulator Preset Register 2	BMPR2_2	0x002070C4
UART2 BRM Modulator Preset Register 3	BMPR3_2	0x002070C8
UART2 BRM Modulator Preset Register 4	BMPR4_2	0x002070CC
UART2 Test Register 1	UTS_2	0x002070D0

24.7.1 UART Receiver Registers

The read-only UART Receiver Registers contain the received character and its status. After reset, when the receiver enable bit is set (RXEN = 1), these registers contain random data and the CHARRDY bit is 0 until the first character is received. The URXD*n_x* registers are each mapped to 16 word-length addresses to support the LDM instruction.

		Addr
URXDn_1	UART1 Receiver Register n	**0x00206000+4*n**
URXDn_2	UART2 Receiver Register n	**0x00207000+4*n**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CHAR RDY	ERR	OVR RUN	FRM ERR	BRK	PR ERR			RX_DATA							
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?
								0x0000								

Note: n = (0 through 15)

Table 24-14. UART 1 and 2 Receiver Register Descriptions

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
CHARRDY Bit 15	**Character Ready**—Indicates an invalid read when the RxFIFO is empty and the software attempts to read the previously read data.	0 = The character in the RX_DATA field and its associated flags are invalid. 1 = The character in the RX_DATA field and its associated flags are valid and ready to read
ERR Bit 14	**Error Detect**—Indicates whether the character present in the RX_DATA field has an error (OVRRUN, FRMERR, BRK or PRERR) status. The ERR bit is updated and valid for each received character.	0 = No error status was detected 1 = An error status was detected
OVRRUN Bit 13	**Receiver Overrun**—Indicates whether the receiver ignored data to prevent overwriting the data in the RxFIFO. This error indicates that the software is not keeping up with the incoming data rate. OVRRUN is set for the last (32nd) character written to the RxFIFO to indicate that all characters following this character will be ignored if a read is not performed by the software. OVRRUN is updated and valid for each received character. Under normal circumstances, OVRRUN is never set.	0 = No RxFIFO overrun was detected 1 = A RxFIFO overrun was detected

Table 24-14. UART 1 and 2 Receiver Register Descriptions (continued)

Name	Description	Settings
FRMERR Bit 12	**Frame Error**—Indicates whether the current character had a framing error (a missing stop bit) and is possibly corrupted. FRMERR is updated for each character read from the RxFIFO.	0 = The current character has no framing error 1 = The current character has a framing error
BRK Bit 11	**BREAK Detect**—Indicates whether the current character was detected as a BREAK character. The data bits and the stop bit are all 0. The FRMERR bit is set when BRK is set. When odd parity is selected, PRERR is also set when BRK is set. BRK is valid for each character read from the RxFIFO.	0 = The current character is not a BREAK character 1 = The current character is a BREAK character
PRERR Bit 10	**Parity Error**—Indicates whether the current character was detected with a parity error and is possibly corrupted. PRERR is updated for each character read from the RxFIFO. When parity is disabled, PRERR always reads as 0.	0 = No parity error was detected for data in the RX_DATA field 1 = A parity error was detected for data in the RX_DATA field
Reserved Bits 9–8	Reserved—These bits are reserved and should read 0.	
RX_DATA Bits 7–0	**Received Data**—Holds the received character. In 7-bit mode, the most significant bit (MSB) is forced to 0. In 8-bit mode, all bits are active.	

24.7.2 UART Transmitter Registers

The write-only UART Transmitter Registers are where the ARM920T processor writes the data to be transmitted. When these registers are read, the TX_DATA bits are always read as 0. The UTXnD_*x* registers are each mapped to 16 word-length addresses to support the STM instruction.

		Addr
UTXnD_1	UART1 Transmitter Register n	**0x00206040+4*n**
UTXnD_2	UART2 Transmitter Register n	**0x00207040+4*n**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									TX_DATA							
TYPE	r	r	r	r	r	r	r	r	w	w	w	w	w	w	w	w
RESET	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?
								0x0000								

Note: n = (0 through 15)

Table 24-15. UART 1 and 2 Transmitter Register Descriptions

Name	Description
Reserved Bits 31–8	Reserved—These bits are reserved and should read 0.
TX_DATA Bits 7–0	**Transmit Data**—Holds the parallel transmit data inputs. In 7-bit mode, D7 is ignored. In 8-bit mode, all bits are used. Data is transmitted least significant bit (LSB) first. A new character is transmitted when the TX_DATA field is written. The TX_DATA field must be written only when the TRDY bit is high to ensure that corrupted data is not sent.

24.7.3 UART Control Register 1

UART Control Register 1 enables the respective UART features and controls the transmit and receive blocks. This register also controls the TxFIFO and RxFIFO levels and enables the TRDY and RRDY interrupts.

		Addr
UCR1_1	UART1 Control Register 1	**0x00206080**
UCR1_2	UART2 Control Register 1	**0x00207080**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
									0x0000							

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ADEN	ADBR	TRDY EN	IDEN	ICD		RRDY EN	RDMA EN	IREN	TXMPTY EN	RTSD EN	SND BRK	TDMA EN	UARTCLK EN	DOZE	UART EN
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
									0x0004							

Table 24-16. UART 1 and 2 Control Register 1 Descriptions

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
ADEN Bit 15	**Automatic Baud Rate Detection Interrupt Enable**—Enables/Disables the automatic baud rate detect complete (ADET) bit to generate an interrupt ($\overline{\text{UART_MINT_UARTC}}$ = 0).	0 = Disable the automatic baud rate detection interrupt 1 = Enable the automatic baud rate detection interrupt
ADBR Bit 14	**Automatic Detection of Baud Rate**—Enables/Disables automatic baud rate detection. When the ADBR bit is set and the ADET bit is cleared, the receiver detects the incoming baud rate automatically. The ADET flag is set when the receiver verifies that the incoming baud rate is detected properly by detecting an ASCII character “A” or “a” (0x61 or 0x41).	0 = Disable automatic detection of baud rate 1 = Enable automatic detection of baud rate
TRDYEN Bit 13	**Transmitter Ready Interrupt Enable**—Enables/Disables the transmitter Ready Interrupt (TRDY) when the transmitter has one or more slots available in the TxFIFO. The fill level in the TXFIFO at which an interrupt is generated is controlled by TxTL bits. When TRDYEN is negated, the transmitter ready interrupt is disabled.	0 = Disable the transmitter ready interrupt 1 = Enable the transmitter ready interrupt
IDEN Bit 12	**Idle Condition Detected Interrupt Enable**—Enables/Disables the IDLE bit to generate an interrupt ($\overline{\text{UART_MINT_RX}}$ = 0).	0 = Disable the IDLE bit 1 = Enable the IDLE bit

Table 24-16. UART 1 and 2 Control Register 1 Descriptions (continued)

Name	Description	Settings
ICD Bits 11–10	**Idle Condition Detect**—Controls the number of frames RXD is allowed to be idle before an idle condition is reported.	00 = Idle for more than 4 frames 01 = Idle for more than 8 frames 10 = Idle for more than 16 frames 11 = Idle for more than 32 frames
RRDYEN Bit 9	**Receiver Ready Interrupt Enable**—Enables/Disables the RRDY interrupt when the RxFIFO contains data. The fill level in the RXFIFO at which an interrupt is generated is controlled by the RXTL bits. When RRDYEN is negated, the receiver ready interrupt is disabled.	0 = Disables the RRDY interrupt 1 = Enables the RRDY interrupt
RDMAEN Bit 8	**Receive Ready DMA Enable**—Enables/Disables the receive DMA request $\overline{\text{UART_RX_DMAREQ}}$ when the receiver has data in the RxFIFO. The fill level in the RxFIFO at which a DMA request is generated is controlled by the RXFL bits. When negated, the receive DMA request is disabled.	0 = Disable $\overline{\text{UART_RX_DMAREQ}}$ DMA request 1 = Enable $\overline{\text{UART_RX_DMAREQ}}$ DMA request
IREN Bit 7	**Infrared Interface Enable**—Enables/Disables the IR interface. See the IR interface description in Section 24.6 for more information.	0 = Disable the IR interface 1 = Enable the IR interface
TXMPTYEN Bit 6	**Transmitter Empty Interrupt Enable**—Enables/Disables the transmitter FIFO empty (TXFE) interrupt. $\overline{\text{UART_MINT_TX}}$. When negated, the TXFE interrupt is disabled.	0 = Disable the transmitter FIFO empty interrupt 1 = Enable the transmitter FIFO empty interrupt
RTSDEN Bit 5	**RTS Delta Interrupt Enable**—Enables/Disables the RTSD interrupt. The current status of the $\overline{\text{UARTx_RTS}}$ pin is read in the RTSS bit.	0 = Disable RTSD interrupt 1 = Enable RTSD interrupt
SNDBRK Bit 4	**Send BREAK**—Forces the transmitter to send a BREAK character. The transmitter finishes sending the character in progress (if any) and sends BREAK characters until SNDBRK is reset. Because the transmitter samples SNDBRK after every bit is transmitted, it is important that SNDBRK is asserted high for a sufficient period of time to generate a valid BREAK. After the BREAK transmission completes, the UART transmits 2 mark bits. The user can continue to fill the TxFIFO and any characters remaining are transmitted when the BREAK is terminated.	0 = Do not send a BREAK character 1 = Send a BREAK character (continuous 0s)
TDMAEN Bit 3	**Transmitter Ready DMA Enable**—Enables/Disables the transmit DMA request $\overline{\text{UART_TX_DMAREQ}}$ when the transmitter has one or more slots available in the TxFIFO. The fill level in the TxFIFO that generates the $\overline{\text{UART_TX_DMAREQ}}$ is controlled by the TXTL bits.	0 = Disable transmit DMA request 1 = Enable transmit DMA request
UARTCLKEN Bit 2	**UART Clock Enable**—Enables/Disables all of the internal clocks of the UART module	0 = Disable UART clocks 1 = Enable UART clocks

Table 24-16. UART 1 and 2 Control Register 1 Descriptions (continued)

Name	Description	Settings
DOZE Bit 1	**DOZE**—Determines the UART enable condition in the DOZE state. When the ARM9 core executes a DOZE instruction and the system is placed in the DOZE state, the DOZE bit affects the operation of the UART. When the system is in the DOZE state and the DOZE bit is asserted, the UART is disabled. See Section 24.8, “UART Operation in Low-Power System States,” on page 24-52 for more information.	0 = The UART is enabled when in DOZE state 1 = The UART is disabled when in DOZE state
UARTEN Bit 0	**UART Enable**—Enables/Disables the UART. If UARTEN is negated in the middle of a transmission, the transmitter stops and pulls the TXD line to a logic 1.	0 = Disable the UART 1 = Enable the UART

24.7.4 UART Control Register 2

The UART Control Register 2 controls the overall operation of its respective UART. The bits in these registers set the BITSEL and its source, specify the number of bits per character, enable or disable parity generation and checking, and control the $\overline{RTS}$ and $\overline{CTS}$ pins.

		Addr
UCR2_1	UART1 Control Register 2	**0x00206084**
UCR2_2	UART2 Control Register 2	**0x00207084**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ESCI	IRTS	CTSC	CTS	ESC EN	RTEC		PREN	PROE	STPB	WS	RTS EN		TXEN	RXEN	$\overline{SRST}$
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	0x0001															

Table 24-17. UART 1 and 2 Control Register 2 Descriptions

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
ESCI Bit 15	**Escape Sequence Interrupt Enable**—Enables/Disables the ESCF bit to generate an interrupt.	0 = Disable the escape sequence interrupt 1 = Enable the escape sequence interrupt
IRTS Bit 14	**Ignore $\overline{\text{UARTx_RTS}}$ Pin**—Forces the $\overline{\text{UARTx_RTS}}$ input signal presented to the transmitter to always be asserted, effectively ignoring the external pin. When in this mode, the $\overline{\text{UARTx_RTS}}$ pin serves as a general purpose input.	0 = Transmit only when the $\overline{\text{UARTx_RTS}}$ pin is asserted 1 = Ignore the $\overline{\text{UARTx_RTS}}$ pin
CTSC Bit 13	**$\overline{\text{UARTx_CTS}}$ Pin Control**—Controls the operation of the $\overline{\text{UARTx_CTS}}$ output pin. When CTSC is asserted, the $\overline{\text{UARTx_CTS}}$ output pin is controlled by the receiver. When the RxFIFO is filled to the level of the programmed trigger level and the start bit of the overflowing character (TRIGGER LEVEL + 1) is validated, the $\overline{\text{UARTx_CTS}}$ output pin is negated to indicate to the far-end transmitter to stop transmitting. When the trigger level is programmed for less than 32, the receiver continues to receive data until the RxFIFO is full. When the CTSC bit is negated, the $\overline{\text{UARTx_CTS}}$ output pin is controlled by the CTS bit. On reset, because CTSC is cleared to 0, the $\overline{\text{UARTx_CTS}}$ pin is controlled by the CTS bit, which again is cleared to 0 on reset. This means that on reset the $\overline{\text{UARTx_CTS}}$ signal is negated.	0 = The $\overline{\text{UARTx_CTS}}$ pin is controlled by the CTS bit 1 = The $\overline{\text{UARTx_CTS}}$ pin is controlled by the receiver

Table 24-17. UART 1 and 2 Control Register 2 Descriptions (continued)

Name	Description	Settings
CTS Bit 12	**Clear to Send**—Controls the $\overline{\text{UARTx_CTS}}$ pin when the CTSC bit is negated. CTS has no function when CTSC is asserted.	0 = The $\overline{\text{UARTx_CTS}}$ pin is high (inactive) 1 = The $\overline{\text{UARTx_CTS}}$ pin is low (active)
ESCEN Bit 11	**Escape Enable**—Enables/Disables the escape sequence detection logic.	0 = Disable escape sequence detection 1 = Enable escape sequence detection
RTEC Bits 10–9	**Request to Send Edge Control**—Selects the edge that triggers the RTS interrupt. This has no effect on the RTS delta interrupt. RTEC has an effect only when RTSEN = 1 (see Table 24-4).	00 = Trigger interrupt on a rising edge 01 = Trigger interrupt on a falling edge 1X = Trigger interrupt on any edge
PREN Bit 8	**Parity Enable**—Enables/Disables the parity generator in the transmitter and parity checker in the receiver. When PREN is asserted, the parity generator and checker are enabled, and disabled when PREN is negated.	0 = Disable parity generator and checker 1 = Enable parity generator and checker
PROE Bit 7	**Parity Odd/Even**—Controls the sense of the parity generator and checker. When PROE is high, odd parity is generated and expected. When PROE is low, even parity is generated and expected. PROE has no function if PREN is low.	0 = Even parity 1 = Odd parity
STPB Bit 6	**Stop**—Controls the number of stop bits transmitted after a character. When STPB is high, 2 stop bits are sent. When STPB is low, 1 stop bit is sent. STPB has no effect on the receiver, which expects 1 or more stop bits.	0 = 1 stop bit transmitted 1 = 2 stop bits transmitted
WS Bit 5	**Word Size**—Controls the character length. When WS is high, the transmitter and receiver are in 8-bit mode. When WS is low, they are in 7-bit mode. The transmitter ignores bit 7 and the receiver sets bit 7 to 0. WS can be changed in-between transmission (reception) of characters, however not when a transmission (reception) is in progress, in which case the length of the current character being transmitted (received) is unpredictable.	0 = 7-bit transmit and receive character length (not including START, STOP or PARITY bits) 1 = 8-bit transmit and receive character length (not including START, STOP or PARITY bits)
RTSEN Bit 4	**Request to Send Interrupt Enable**—Controls the RTS edge sensitive interrupt. When RTSEN is asserted and the programmed edge is detected on the $\overline{\text{UARTx_RTS}}$ pin, the RTSF bit is asserted (see Table 24-4).	0 = Disable request to send interrupt 1 = Enable request to send interrupt
Reserved Bit 3	Reserved—This bit is reserved and should read 0.	
TXEN Bit 2	**Transmitter Enable**—Enables/Disables the transmitter. When TXEN is negated the transmitter is disabled and idle. When the UARTEN and TXEN bits are set the transmitter is enabled. If TXEN is negated in the middle of a transmission, the UART disables the transmitter immediately, and starts marking 1s.	0 = Disable the transmitter 1 = Enable the transmitter

Table 24-17. UART 1 and 2 Control Register 2 Descriptions (continued)

Name	Description	Settings
RXEN Bit 1	**Receiver Enable**—Enables/Disables the receiver. When the receiver is enabled, if the RXD input is already low, the receiver does not recognize BREAK characters, because it requires a valid 1-to-0 transition before it can accept any character.	0 = Disable the receiver 1 = Enable the receiver
$\overline{\text{SRST}}$ Bit 0	**Software Reset**—Resets the transmitter and receiver state machines, all FIFOs, and all status registers. Once the software writes 0 to $\overline{\text{SRST}}$, the software reset remains active for 4 clock cycles of CKIH before the hardware deasserts $\overline{\text{SRST}}$. The software can only write 0 to $\overline{\text{SRST}}$. Writing 1 to $\overline{\text{SRST}}$ is ignored.	0 = Reset the transmit and receive state machines, all FIFOs and all status registers 1 = No reset

24.7.5 UART Control Register 3

The UART Control Register 3 controls the features and operation of UART 1.

24.7.5.1 UART1 Control Register 3

UCR3_1 UART1 Control Register 3 **Addr 0x00206088**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				PARERREN	FRAERREN					RXDS EN	AIRINT EN	AWAK EN	REF25	REF30	INVT	BPEN
TYPE	r	r	r	rw	rw	r	r	r	r	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 24-18. UART 1 Control Register 3 Descriptions

Name	Description	Settings
Reserved Bits 31–13	Reserved—These bits are reserved and should read 0.	
PARERREN Bit 12	**Parity Error Interrupt Enable**—Enables/Disables the interrupt. When asserted, PARERREN causes the PARITYERR bit to generate an interrupt ($\overline{\text{UART_MINT_PFERR}}$ = 0)	0 = Disable the parity error interrupt 1 = Enable the parity error interrupt
FRAERREN Bit 11	**Frame Error Interrupt Enable**—Enables/Disables the interrupt. When asserted, FRAERREN causes the FRAMERR bit to generate an interrupt ($\overline{\text{UART_MINT_PFERR}}$ = 0)	0 = Disable the frame error interrupt 1 = Enable the frame error interrupt
Reserved Bits 10–7	Reserved—These bits are reserved and should read 0.For proper operation, ensure these bits always read 0.	
RXDSEN Bit 6	**Receive Status Interrupt Enable**—Controls the receive status interrupt ($\overline{\text{UART_MINT_RX}}$). When this bit is enabled and RXDS status bit is set, the interrupt $\overline{\text{UART_MINT_RX}}$ will be generated.	0 = Disable the RXDS interrupt 1 = Enable the RXDS interrupt
AIRINTEN Bit 5	**Asynchronous IR WAKE Interrupt Enable**—Controls the asynchronous IR WAKE interrupt. An interrupt is generated when AIRINTEN is asserted and a pulse is detected on the UART_RX pin.	0 = Disable the AIRINT interrupt 1 = Enable the AIRINT interrupt

Table 24-18. UART 1 Control Register 3 Descriptions

Name	Description	Settings
AWAKEN Bit 4	**Asynchronous WAKE Interrupt Enable**—Controls the asynchronous WAKE interrupt. An interrupt is generated when AWAKEN is asserted and a falling edge is detected on the RXD pin.	0 = Disable the AWAKE interrupt 1 = Enable the AWAKE interrupt
REF25 Bit 3	**Reference Frequency 25 MHz**—Indicates to the hardware that a reference clock frequency of 25 MHz is used. The reference clock is derived from the input clock IPG_CLK via the programmable divider.	0 = 25 MHz reference clock not used 1 = 25 MHz reference clock used
REF30 Bit 2	**Reference Frequency 30 MHz**—Indicates to the hardware that a reference clock frequency of 30 MHz is used. The reference clock is derived from the input clock IPG_CLK via the programmable divider.	0 = 30 MHz reference clock not used 1 = 30 MHz reference clock used
INVT Bit 1	**Inverted Infrared Transmission**—Sets the active level for the transmission. When INVT is cleared, the infrared logic block transmits a positive IR 3/16 pulse for all 0s and 0s are transmitted for 1s. When INVT is set (INVT = 1), the infrared logic block transmits an active low or negative infrared 3/16 pulse for all 0s and 1s are transmitted for 1s.	0 = Active low transmission 1 = Active high transmission
BPEN Bit 0	**Preset Registers Enable**—Activates the BRM preset registers for use during automatic baud rate detection mode. When BPEN is deasserted, the remainder from the automatic baud count register divided by 16 is ignored. When BPEN is asserted, the remainder and the dividend chooses the appropriate preset register on the detection of special baud rates. If the criteria is not met for selecting one of the preset registers, integer division is performed by writing one less than the dividend to the UBMR_1/UBMR_2 register.	0 = The BRM preset registers are not used (normal operation) 1 = The BRM preset registers are used (auto detect special baud rates)

24.7.5.2 UART2 Control Register 3

UCR3_2	UART2 Control Register 3	Addr 0x00207088

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DPEC		DTR EN	PARERR EN	FRAERR EN	DSR	DCD	RI		RXDS EN	AIR INT EN	AWAK EN	REF25	REF30	INVT	BPEN
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 24-19. UART 2 Control Register 3 Descriptions

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
DPEC Bits 15–14	**DTR Interrupt Edge Control**—Controls the edge that generates an interrupt. An interrupt is generated only if the DTREN bit is set.	00 = Interrupt generated on rising edge 01 = Interrupt generated on falling edge 1X = Interrupt generated on either edge
DTREN Bit 13	**Data Terminal Ready Interrupt Enable**—Controls the DTR edge sensitive interrupt. When DTREN is asserted and the programmed edge is detected on the $\overline{\text{UART2_DTR}}$ pin, the DTRF bit is asserted (see Table 24-4).	0 = Disable the data terminal ready interrupt 1 = Enable the data terminal ready interrupt
PARERREN Bit 12	**Parity Error Interrupt Enable**—Enables/Disables the interrupt. When asserted, PARERREN causes the PARITYERR bit to generate an interrupt ($\overline{\text{UART_MINT_PFERR}}$ = 0)	0 = Disable the parity error interrupt 1 = Enable the parity error interrupt
FRAERREN Bit 11	**Frame Error Interrupt Enable**—Enables/Disables the interrupt. When asserted, FRAERREN causes the FRAMERR bit to generate an interrupt ($\overline{\text{UART_MINT_PFERR}}$ = 0)	0 = Disable the frame error interrupt 1 = Enable the frame error interrupt
DSR Bit 10	**Data Set Ready**—Selects the logic level for the $\overline{\text{UART_DSR}}$ pin for the modem interface.	0 = The $\overline{\text{UART_DSR}}$ pin is logic 0 1 = The $\overline{\text{UART_DSR}}$ pin is logic 1
DCD Bit 9	**Data Carrier Detect**—Selects the logic level for the $\overline{\text{UART_DCD}}$ pin for the modem interface.	0 = The $\overline{\text{UART_DCD}}$ pin is logic 0 1 = The $\overline{\text{UART_DCD}}$ pin is logic 1
RI Bit 8	**Ring Indicator**—Selects the logic level for the $\overline{\text{UART_RI}}$ pin for the modem interface.	0 = The $\overline{\text{UART_RI}}$ pin is logic 0 1 = The $\overline{\text{UART_RI}}$ pin is logic 1

Table 24-19. UART 2 Control Register 3 Descriptions (continued)

Name	Description	Settings
Reserved Bit 7	This bit is reserved and should read zero. For proper operation, ensure this bit always reads 0.	
RXDSEN Bit 6	**Receive Status Interrupt Enable**—Controls the receive status interrupt (UART_MINT_RX). When this bit is enabled and RXDS status bit is set, the interrupt UART_MINT_RX will be generated.	0 = Disable the RXDS interrupt 1 = Enable the RXDS interrupt
AIRINTEN Bit 5	**Asynchronous IR WAKE Interrupt Enable**—Controls the asynchronous IR WAKE interrupt. An interrupt is generated when AIRINTEN is asserted and a pulse is detected on the UART_RX pin.	0 = Disable the AIRINT interrupt 1 = Enable the AIRINT interrupt
AWAKEN Bit 4	**Asynchronous WAKE Interrupt Enable**—Controls the asynchronous WAKE interrupt. An interrupt is generated when AWAKEN is asserted and a falling edge is detected on the RXD pin.	0 = Disable the AWAKE interrupt 1 = Enable the AWAKE interrupt
REF25 Bit 3	**Reference Frequency 25 MHz**—Indicates to the hardware that a reference clock frequency of 25 MHz is used. The reference clock is derived from the input clock IPG_CLK via the programmable divider.	0 = 25 MHz reference clock not used 1 = 25 MHz reference clock used
REF30 Bit 2	**Reference Frequency 30 MHz**—Indicates to the hardware that a reference clock frequency of 30 MHz is used. The reference clock is derived from the input clock IPG_CLK via the programmable divider.	0 = 30 MHz reference clock not used 1 = 30 MHz reference clock used
INVT Bit 1	**Inverted Infrared Transmission**—Sets the active level for the transmission. When INVT is cleared, the infrared logic block transmits a positive IR 3/16 pulse for all 0s and 0s are transmitted for 1s. When INVT is set (INVT = 1), the infrared logic block transmits an active low or negative infrared 3/16 pulse for all 0s and 1s are transmitted for 1s.	0 = Active low transmission 1 = Active high transmission
BPEN Bit 0	**Preset Registers Enable**—Activates the BRM preset registers for use during automatic baud rate detection mode. When BPEN is deasserted, the remainder from the automatic baud count register divided by 16 is ignored. When BPEN is asserted, the remainder and the dividend chooses the appropriate preset register on the detection of special baud rates. If the criteria is not met for selecting one of the preset registers, integer division is performed by writing one less than the dividend to the UMBR_2/UMBR_3 register.	0 = The BRM preset registers are not used (normal operation) 1 = The BRM preset registers are used (auto detect special baud rates)

24.7.6 UART Control Register 4

The UART Control Register 4 provides configuration control for the UARTs and is used to enable/disable several UART interrupts.

		Addr
UCR4_1	UART1 Control Register 4	**0x0020608C**
UCR4_2	UART2 Control Register 4	**0x0020708C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
									0x0000							

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CTSTL						INVR	ENIRI	WKEN	REF16	IRSC		TCEN	BKEN	OREN	DREN
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw
RESET	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
									0x8040							

Table 24-20. UART 1 and 2 Control Register 4 Descriptions

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
CTSTL Bits 15–10	**CTS Trigger Level**—Controls the threshold at which the CTS pin is deasserted by the RxFIFO. After the trigger level is reached and the CTS pin is deasserted, the RxFIFO continues to receive data until it is full. The CTSTL bits are encoded as shown in the Settings column.	000000 = 0 characters received 000001 = 1 characters in the RxFIFO ... 100000 = 32 characters in the RxFIFO (maximum) All Other Settings Reserved
INVR Bit 9	**Inverted Infrared Reception**—Determines the logic level for the detection. When cleared, the infrared logic block expects an active low or negative IR 3/16 pulse for 0s and 1s are expected for 1s. When INVR is set (INVR = 1), the infrared logic block expects an active high or positive IR 3/16 pulse for 0s and 0s are expected for 1s.	0 = Active low detection 1 = Active high detection
ENIRI Bit 8	**Serial Infrared Interrupt Enable**—Enables/Disables the serial infrared interrupt.	0 = Serial infrared Interrupt disabled 1 = Serial infrared Interrupt enabled
WKEN Bit 7	**WAKE Interrupt Enable**—Enables/Disables the WAKE bit to generate an interrupt. The WAKE bit is set at the detection of a start bit by the receiver.	0 = Disable the WAKE interrupt 1 = Enable the WAKE interrupt
REF16 Bit 6	**Reference Frequency 16 MHz**—Indicates to the hardware that a reference clock of 16 MHz is used. The reference clock is derived from input clock IPG_CLK via the programmable divider.	0 = 16 MHz reference clock not used 1 = 16 MHz reference clock is used

Table 24-20. UART 1 and 2 Control Register 4 Descriptions (continued)

Name	Description	Settings
IRSC Bit 5	**IR Special Case**—Selects the clock for the vote logic. When set, IRSC switches the vote logic clock from the sampling clock to the UART reference clock. The IR pulses are counted a predetermined amount of time depending on the reference frequency.	0 = The vote logic uses the sampling clock (16x baud rate) for normal operation 1 = The vote logic uses the UART reference clock
Reserved Bit 4	Reserved—This bit is reserved and should read 0.	
TCEN Bit 3	**Transmit Complete Interrupt Enable**—Enables/Disables the TXDC bit to generate an interrupt ($\overline{\text{UART_MINT_TX}}$ = 0).	0 = Disable TXDC interrupt 1 = Enable TXDC interrupt
BKEN Bit 2	**BREAK Condition Detected Interrupt Enable**—Enables/Disables the BRCD bit to generate an interrupt ($\overline{\text{UART_MINT_UARTC}}$ = 0).	0 = Disable the BRCD interrupt 1 = Enable the BRCD interrupt
OREN Bit 1	**Receiver Overrun Interrupt Enable**—Enables/Disables the ORE bit to generate an interrupt ($\overline{\text{UART_MINT_UARTC}}$ = 0).	0 = Disable ORE interrupt 1 = Enable ORE interrupt
DREN Bit 0	**Receive Data Ready Interrupt Enable**—Enables/Disables the RDR bit to generate an interrupt ($\overline{\text{MINT_RX}}$ = 0).	0 = Disable RDR interrupt 1 = Enable RDR interrupt

24.7.7 UART FIFO Control Registers

The UART FIFO Control Registers control the operation of the UART FIFO trigger levels and interrupts.

		Addr
UFCR_1	UART1 FIFO Control Register	**0x00206090**
UFCR_2	UART2 FIFO Control Register	**0x00207090**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TXTL						RFDIV				RXTL					
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
	0x0881															

Table 24-21. UART 1 and 2 FIFO Control Register Descriptions

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
TXTL Bits 15–10	**Transmitter Trigger Level**—Controls the threshold at which a maskable interrupt is generated by the TxFIFO. A maskable interrupt is generated whenever the data level in the TxFIFO falls below the selected threshold. The bits are encoded as shown in the Settings column. You must ensure the corresponding Interrupt Service Routine (ISR) or DMA configuration are correct in that every FIFO update does not exceed the maximum 32 bytes or size of 32 for TXTL. Exceeding limit can lead to incorrect data or an invalid parity bit. For details refer to Section 24.5.2, “Transmitter FIFO Empty Interrupt Suppression.”	000000 = Reserved 000001 = Reserved 000010 = TxFIFO has 2 or fewer characters ... 011111 = TxFIFO has 31 or fewer characters 100000 = TxFIFO has 32 characters (maximum) All Other Settings Reserved
RFDIV Bits 9–7	**Reference Frequency Divider**—Controls the divide ratio for the reference clock. The input clock is IPG_CLK. The output from the divider must be synchronous with the bus clock IPB_CLK.	000 = Divide input clock by 6 001 = Divide input clock by 5 010 = Divide input clock by 4 011 = Divide input clock by 3 100 = Divide input clock by 2 101 = Divide input clock by 1 110 = Divide input clock by 7

Table 24-21. UART 1 and 2 FIFO Control Register Descriptions (continued)

Name	Description	Settings
Reserved Bit 6	Reserved—This bit is reserved and should read 0.	
RXTL Bits 5–0	**Receiver Trigger Level**—Controls the threshold at which a maskable interrupt is generated by the RxFIFO. A maskable interrupt is generated whenever the data level in the RxFIFO reaches the selected threshold. The RXTL bits are encoded as shown in the Settings column.	000000 = 0 characters received 000001 = RxFIFO has 1 character ... 011111 = RxFIFO has 31 characters 100000 = RxFIFO has 32 characters (maximum) All Other Settings Reserved

24.7.8 UART Status Register 1

The UART Status Register 1 reports errors and status flags for both UARTs.

		Addr
USR1_1	UART1 Status Register 1	**0x00206094**
USR1_2	UART2 Status Register 1	**0x00207094**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PARITY ERR	RTSS	TRDY	RTSD	ESCF	FRAM ERR	RRDY			RXDS	AIR INT	AWAKE				
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x2000															

Table 24-22. UART 1 and 2 Status Register Descriptions

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
PARITYERR Bit 15	**Parity Error Interrupt Flag**—Indicates a parity error is detected. PARITYERR is cleared by writing 1 to it. Writing 0 to PARITYERR has no effect. When parity is disabled, PARITYERR always reads 0. At reset, PARITYERR is set to 0	0 = No parity error detected 1 = Parity error detected
RTSS Bit 14	**$\overline{\text{RTS}}$ Pin Status**—Indicates the current status of the $\overline{\text{UARTx_RTS}}$ pin. A “snapshot” of the pin is taken immediately before RTSS is presented to the data bus. RTSS cannot be cleared because all writes to RTSS are ignored. At reset, RTSS is set to 0.	0 = The $\overline{\text{UARTx_RTS}}$ pin is high (inactive) 1 = The $\overline{\text{UARTx_RTS}}$ pin is low (active)
TRDY Bit 13	**Transmitter Ready Interrupt / DMA Flag**—Indicates that the TxFIFO emptied below its target threshold and requires data. TRDY is automatically cleared when the data level in the TxFIFO goes beyond above the set threshold level by TXFL bits. At reset, TRDY is set to 1.	0 = The transmitter does not require data 1 = The transmitter requires data (interrupt posted)
RTSD Bit 12	**RTS Delta**—Indicates whether the $\overline{\text{RTS}}$ pin changed state. It (RTSD) generates a maskable interrupt. When in STOP mode, only RTS assertion sets RTSD and wakes the ARM9 core. The current state of the $\overline{\text{UARTx_RTS}}$ pin is available on the RTSS bit. Clear RTSD by writing 1 to it. Writing 0 to RTSD has no effect. At reset, RTSD is set to 0.	0 = $\overline{\text{UARTx_RTS}}$ pin did not change state since last cleared 1 = $\overline{\text{UARTx_RTS}}$ pin changed state (write 1 to clear)

Table 24-22. UART 1 and 2 Status Register Descriptions (continued)

Name	Description	Settings
ESCF Bit 11	**Escape Sequence Interrupt Flag**—Indicates if an escape sequence was detected. ESCF is asserted when the ESCEN bit is set and an escape sequence is detected in the RxFIFO. Clear ESCF by writing 1 to it. Writing 0 to ESCF has no effect.	0 = No escape sequence detected 1 = Escape sequence detected (write 1 to clear)
FRAMERR Bit 10	**Frame Error Interrupt Flag**—Indicates that a frame error is detected. The UART_PFERR interrupt generated by this. Clear FRAMERR by writing 1 to it. Writing 0 to FRAMERR has no effect.	0 = No frame error detected 1 = Frame error detected
RRDY Bit 9	**Receiver Ready Interrupt / DMA Flag**—Indicates that the RxFIFO data level is above the threshold set by the RXFL bits. (See the RXFL bits description for setting the interrupt threshold.) When asserted, RRDY generates a maskable interrupt or DMA request. In conjunction with the CHARRDY bit in the URXDn_1 or URXDn_2 register, the software can continue to read the RxFIFO in an interrupt service routine until the RxFIFO is empty. RRDY is automatically cleared when data level in the RxFIFO goes below the set threshold level. At reset, RRDY is set to 0.	0 = No character ready 1 = Character(s) ready (interrupt posted)
Reserved Bits 8-7	Reserved—This bit is reserved and should read 0.	
RXDS Bit 6	**Receiver IDLE Interrupt Flag**—Indicates that the receiver state machine is in an IDLE state, the next state is IDLE, and the receive pin is high. RXDS is automatically cleared when a character is received. RXDS is active only when the receiver is enabled.	0 = Receive in progress 1 = Receiver is IDLE
AIRINT Bit 5	**Asynchronous IR WAKE Interrupt Flag**—Indicates that the IR WAKE pulse was detected (on the RXD pin). Clear AIRINT by writing 1 to it. Writing 0 to AIRINT has no effect.	0 = No pulse was detected on the RXD pin 1 = A pulse was detected on the RXD pin
AWAKE Bit 4	**Asynchronous WAKE Interrupt Flag**—Indicates that a falling edge was detected on the RXD pin. Clear AWAKE by writing 1 to it. Writing 0 to AWAKE has no effect.	0 = No falling edge was detected on the RXD pin 1 = A falling edge was detected on the RXD pin
Reserved Bits 3–0	Reserved—These bits are reserved and should read 0.	

24.7.9 UART Status Register 2

The read-only UART Status Register 2 is used to store data about errors and status flags. There is a status register for each UART.

		Addr
USR2_1	UART1 Status Register 2	**0x00206098**
USR2_2	UART2 Status Register 2	**0x00207098**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ADET	TXFE	DTRF	IDLE				IRINT	WAKE			RTSF	TXDC	BRCD	ORE	RDR
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0
								0x4008								

Table 24-23. UART 1 and 2 Status Register 2 Descriptions

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
ADET Bit 15	**Automatic Baud Rate Detect Complete**—Indicates that an “A” or “a” was received and that the receiver detected and verified the incoming baud rate. Clear ADET by writing 1 to it. Writing 0 to ADET has no effect.	0 = ASCII “A” or “a” was not received 1 = ASCII “A” or “a” was received (write 1 to clear)
TXFE Bit 14	**Transmit Buffer FIFO Empty**—Indicates that the transmit buffer (TxFIFO) is empty. TXFE is cleared automatically when data is written to the TxFIFO. Even though TXFE is high, the transmission might still be in progress.	0 = The transmit buffer (TxFIFO) is not empty 1 = The transmit buffer (TxFIFO) is empty
DTRF (USR2_2 ONLY) Bit 13	**DTR Edge Triggered Interrupt Flag**—Indicates if a programmed edge was detected on the $\overline{\text{UART2_DTR}}$ pin. The DPEC bits select the edge that generates an interrupt (see Table 24-4). The DTRF assertion will cause an interrupt if the DTREN bit in the UCR3 is set. The DTRF interrupt ($\overline{\text{UART_MINT_DTR}}$) is cleared by writing 1 to it. Writing 0 to DTRF has no effect.	0 = Programmed edge not detected on $\overline{\text{UART2_DTR}}$ 1 = Programmed edge detected on $\overline{\text{UART2_DTR}}$ (write 1 to clear)
IDLE Bit 12	**Idle Condition**—Indicates that an idle condition has existed for more than a programmed amount frame (see Section 24.5.4.1). The $\overline{\text{UART_MINT_RX}}$ interrupt generated by this IDLE bit is cleared by writing 1 to IDLE. Writing 0 to IDLE has no effect.	0 = No idle condition detected 1 = Idle condition detected (write 1 to clear)
Reserved Bits 11–9	Reserved—These bits are reserved and should read 0.	

Table 24-23. UART 1 and 2 Status Register 2 Descriptions (continued)

Name	Description	Settings
IRINT Bit 8	**Serial Infrared Interrupt Flag**—Indicates if a valid edge was detected on the RX pin during SIR mode. When ENIRI is asserted and IRINT is asserted, this asserts the common interrupt, $\overline{\text{UART_MINT_UARTC}}$. Clear IRINT by writing 1 to it. Writing 0 to IRINT has no effect.	0 = no edge detected 1 = valid edge detected (write 1 to clear) When INVR = 1, valid edge is 0-to-1 transition edge When INVR = 0, valid edge is 1-to-0 transition edge
WAKE Bit 7	**Wake**—Indicates the start bit is detected. WAKE can generate an interrupt on $\overline{\text{UART_MINT_UARTC}}$ that can be masked using the WKEN bit. Clear WAKE by writing 1 to it. Writing 0 to WAKE has no effect.	0 = start bit not detected 1 = start bit detected (write 1 to clear)
Reserved Bits 6–5	Reserved—These bits are reserved and should read 0.	
RTSF Bit 4	**RTS Edge Triggered Interrupt Flag**—Indicates if a programmed edge is detected on the $\overline{\text{RTS}}$ pin. The RTEC bits select the edge that generates an interrupt (see Table 24-4). RTSF can generate an interrupt on $\overline{\text{UART_MINT_RTS}}$ that can be masked using the RTSEN bit. Clear RTSF by writing 1 to it. Writing 0 to RTSF has no effect.	0 = Programmed edge not detected on RTS 1 = Programmed edge detected on RTS (write 1 to clear)
TXDC Bit 3	**Transmitter Complete**—Indicates that the transmit buffer (TxFIFO) and Shift Register is empty; therefore the transmission is complete. TXDC is cleared automatically when data is written to the TxFIFO.	0 = Transmit is incomplete 1 = Transmit is complete
BRCD Bit 2	**BREAK Condition Detected**—Indicates that a BREAK condition was detected by the receiver. Clear BRCD by writing 1 to it. Writing 0 to BRCD has no effect.	0 = No BREAK condition was detected 1 = A BREAK condition was detected (write 1 to clear)
ORE Bit 1	**Overrun Error**—Indicates that the receive buffer (RxFIFO) was full when data was being received. Clear ORE by writing 1 to it. Writing 0 to ORE has no effect.	0 = No overrun error 1 = Overrun error (write 1 to clear)
RDR Bit 0	**Receive Data Ready**—Indicates that at least 1 character is received and written to the RxFIFO. If the URXDn_1 or URXDn_2 register is read and there is only 1 character in the RxFIFO, RDR is automatically cleared.	0 = No receive data ready 1 = Receive data ready

24.7.10 UART Escape Character Registers

The UART Escape Character Registers establish the software-selected escape character.

		Addr
UESC_1	UART1 Escape Character Register	**0x0020609C**
UESC_2	UART2 Escape Character Register	**0x0020709C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									ESC_CHAR							
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1
	0x002B															

Table 24-24. UART 1 and 2 Escape Character Register Descriptions

Name	Description
Reserved Bits 31–8	Reserved—These bits are reserved and should read 0.
ESC_CHAR Bits 7–0	**UART Escape Character**—Holds the selected escape character that all received characters are compared against to detect an escape sequence.

24.7.11 UART Escape Timer Registers

The UART Escape Timer Registers establish the software-selected maximum interval between escape characters. These registers are scalable in intervals of 2 msec to a maximum of 8.192 sec.

		Addr
UTIM_1	UART1 Escape Timer Register	**0x002060A0**
UTIM_2	UART2 Escape Timer Register	**0x002070A0**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					TIM											
TYPE	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 24-25. UART 1 and 2 Escape Timer Register Descriptions

Name	Description
Reserved Bits 31–12	Reserved—These bits are reserved and should read 0.
TIM Bits 11–0	**UART Escape Timer**—Holds the maximum interval allowed between escape characters.

24.7.12 UART BRM Incremental Registers

The BRM (Binary Rate Multiplier) Incremental registers hold the numerator value (minus one) of the BRM ratio. These registers can be written to at any time. Hardware updates the value in the UART BRM Modulator Registers (UBMR_*x*) at the appropriate time to avoid glitches on the BRM_CLK output (sampling clock). The BRM is not updated until both the modulator of the respective UBMR_*x* register and its incremental UBIR_*x* registers are written to by software. If only one register of the two registers is written to by the software, the BRM ignores this data until the other register is also written.

		Addr
UBIR_1	UART1 BRM Incremental Register	**0x002060A4**
UBIR_2	UART2 BRM Incremental Register	**0x002070A4**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	INC															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 24-26. UART 1 and 2 BRM Incremental Register Descriptions

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
INC Bits 15–0	**Incremental Numerator**—Holds the numerator value minus one of the BRM ratio (see Section 24.5.8). Updating this field using byte accesses is not recommended and is undefined. This register cannot be written to by the user if ADBR=1.

24.7.13 UART BRM Modulator Registers

The UART BRM (Binary Rate Multiplier) Modulator Registers hold the denominator value (minus one) of the BRM ratio. These registers can be written to at any time. Hardware updates the value in the UART BRM Modulator Registers (UBMR_*x*) at the appropriate time to avoid glitches on the BRM_CLK output (sampling clock). The BRM is not updated until both the modulator of the respective UBMR_*x* register and its incremental UBIR_*x* registers are written to by software. If only one register of the two registers is written to by the software, the BRM ignores this data until the other register is also written.

		Addr
UBMR_1	UART1 BRM Modulator Register	**0x002060A8**
UBMR_2	UART2 BRM Modulator Register	**0x002070A8**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MOD															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 24-27. UART 1 and 2 BRM Modulator Register Descriptions

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
MOD Bits 15–0	**Modulator Denominator**—Holds the value of the denominator minus one of the BRM ratio (see Section 24.5.8). Updating this register using byte accesses is not recommended and undefined.

24.7.14 UART Baud Rate Count Registers

The read-only UART Baud Rate Count Registers count the start bit of the incoming baud rate. When the start bit is detected and counted, these registers retain their value until the next automatic baud rate detection sequence is initiated. The registers are reset to 0x00000008, however they stay at 0x0000FFFF in the case of an overflow.

		Addr
UBRC_1	UART1 Baud Rate Count Register	**0x002060AC**
UBRC_2	UART2 Baud Rate Count Register	**0x002070AC**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	BCNT															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	0x0008															

Table 24-28. UART 1 and 2 Baud Rate Count Register Descriptions

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
BCNT Bits 15–0	**Baud Rate Count Register**—Counts or measures the length of the incoming baud rate start bit when in automatic baud rate detection mode. This register is clocked by the BRM_CLK.

24.7.15 UART BRM Incremental Preset Registers 1–4

The eight 32-bit UART BRM Incremental Preset Registers (BIPR*n_x*) are used in conjunction with the automatic baud detection logic. These registers are assigned preset values for the special baud rates 920 Kbps, 460 Kbps, 230 Kbps, and 115.2 Kbps. When the BPEN bit is asserted, these registers are used on the detection of a remainder. These registers must be written to prior to enabling the auto baud detection logic. This feature supports 16 MHz, 25 MHz, and 30 MHz reference frequencies only. Enabling this feature with other reference frequencies is not supported.

		Addr
BIPR1_1	UART1 BRM Incremental Preset Register 1	**0x002060B0**
BIPR1_2	UART2 BRM Incremental Preset Register 1	**0x002070B0**
BIPR2_1	UART1 BRM Incremental Preset Register 2	**0x002060B4**
BIPR2_2	UART2 BRM Incremental Preset Register 2	**0x002070B4**
BIPR3_1	UART1 BRM Incremental Preset Register 3	**0x002060B8**
BIPR3_2	UART2 BRM Incremental Preset Register 3	**0x002070B8**
BIPR4_1	UART1 BRM Incremental Preset Register 4	**0x002060BC**
BIPR4_2	UART2 BRM Incremental Preset Register 4	**0x002070BC**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	INCPI															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
	0x????															

Table 24-29. UART 1 and 2 Incremental Preset Registers 1–4 Descriptions

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
INCPI Bits 15–0	**BRM Incremental Preset Register**—Holds the appropriate numerator for the special baud rates when in automatic detect mode. BIPR1_*x* register is for the 920 Kbps BIPR2_x register is for the 460 Kbps BIPR3_*x* register is for the 230 Kbps BIPR4_*x* register is for the 115.2 Kbps

24.7.16 UART BRM Modulator Preset Registers 1–4

		Addr
BMPR1_1	UART1 BRM Modulator Preset Register 1	**0x002060C0**
BMPR1_2	UART2 BRM Modulator Preset Register 1	**0x002070C0**
BMPR2_1	UART1 BRM Modulator Preset Register 2	**0x002060C4**
BMPR2_2	UART2 BRM Modulator Preset Register 2	**0x002070C4**
BMPR3_1	UART1 BRM Modulator Preset Register 3	**0x002060C8**
BMPR3_2	UART2 BRM Modulator Preset Register 3	**0x002070C8**
BMPR4_1	UART1 BRM Modulator Preset Register 4	**0x002060CC**
BMPR4_2	UART2 BRM Modulator Preset Register 4	**0x002070CC**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MODI															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
	0x????															

Table 24-30. UART 1 and 2 Modulator Preset Registers 1–4 Descriptions

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
MODI Bits 15–0	**BRM MOD Preset Registers**—Holds the appropriate numerator for the special baud rates when in automatic detect mode. BMPR1_*x* register is for 920 Kbps BMPR2_*x* register is for 460 Kbps BMPR3_*x* register is for 230 Kbps BMPR4_*x* register is for 115.2 Kbps

24.7.17 UART Test Register 1

The UART Test Register 1 controls the various test features of the UART module.

UTS_1	UART1 Test Register 1	Addr 0x002060D0
UTS_2	UART2 Test Register 1	0x002070D0

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			FRCP ERR	LOOP						TX EMPTY	RX EMPTY	TX FULL	RX FULL			SOFT RST
TYPE	r	r	rw	rw	r	rw	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
	0x0060															

Table 24-31. UART 1 and 2 Test Register Descriptions

Name	Description	Settings
Reserved Bits 31–14	Reserved—These bits are reserved and should read 0.	
FRCPERR Bit 13	**Force Parity Error**—Forces the transmitter to generate a parity error if parity is enabled. FRCPERR is provided for system debugging.	0 = Generate normal parity 1 = Generate inverted parity (error)
LOOP Bit 12	**Loop TX and RX for Test**—Controls loopback for test purposes. When LOOP is high, the receiver input is internally connected to the transmitter and ignores the RXD pin. The transmitter is unaffected by LOOP.	0 = Normal receiver operation 1 = Internally connect the transmitter output to the receiver input
Reserved Bits 11–7	Reserved—These bits are reserved and should read 0.	
TXEMPTY Bit 6	**TxFIFO Empty**—Indicates that the TxFIFO is empty.	0 = The TxFIFO is not empty 1 = The TxFIFO is empty
RXEMPTY Bit 5	**RxFIFO Empty**—Indicates the RxFIFO is empty.	0 = The RxFIFO is not empty 1 = The RxFIFO is empty
TXFULL Bit 4	**TxFIFO FULL**—Indicates the TxFIFO is full.	0 = The TxFIFO is not full 1 = The TxFIFO is full
RXFULL Bit 3	**RxFIFO FULL**—Indicates the RxFIFO is full.	0 = The RxFIFO is not full 1 = The RxFIFO is full

Table 24-31. UART 1 and 2 Test Register Descriptions (continued)

Name	Description	Settings
Reserved Bits 2–1	Reserved—These bits are reserved and should read 0.	
SOFTRST Bit 0	**Software Reset**—Indicates the status of the software reset ($\overline{SRST}$).	0 = No software reset 1 = Generate software reset

24.8 UART Operation in Low-Power System States

The UART's serial interface will operate as long as the 16x bit clock generator is provided with the PerCLK1. The RXEN, TXEN and UARTEN bits are set by the user and provide software control of low-power modes.

When in DOZE state, the UART's behavior depends on the DOZE bit. If the DOZE bit is negated and the system is in DOZE state, the UART serial interface is operational. If the system is in DOZE state and the DOZE bit is asserted, the UART transmitter and receiver operations are halted.

If the DOZE state is entered with the DOZE bit asserted when the UART's serial interface is receiving or transmitting data, the reception/transmission of the current character is completed and signal to the far-end transmitter/receiver to stop sending/receiving. The control/status/data registers do not change when entering or exiting low-power modes.

The following UART interrupts wake the ARM920T processor from STOP mode:

- RTS
- IrDA Asynchronous WAKE (AIRINT)
- Asynchronous WAKE (AWAKE)

The UART_CLK_EN bit (UCR1_*x*) enables the clock inputs to the UART module. If the control bit is set to 0, it abruptly enables the clock inputs to the UART module. To get maximum power conservation when the module is shut off, make sure that the above clock control bit is set to 0. Setting the UARTEN (UCR1_*x*) bit to 0 only shuts off the receiver and transmitter logic and the associated clocks.

When an asynchronous WAKE interrupt exits the ARM920T processor from STOP mode, make sure that a dummy character is sent first because the first character may not be received correctly. After the settling time of the USB_PLL, actual characters can be sent to the UART. Even though the dummy character is written to RxFIFO, just ignore it.

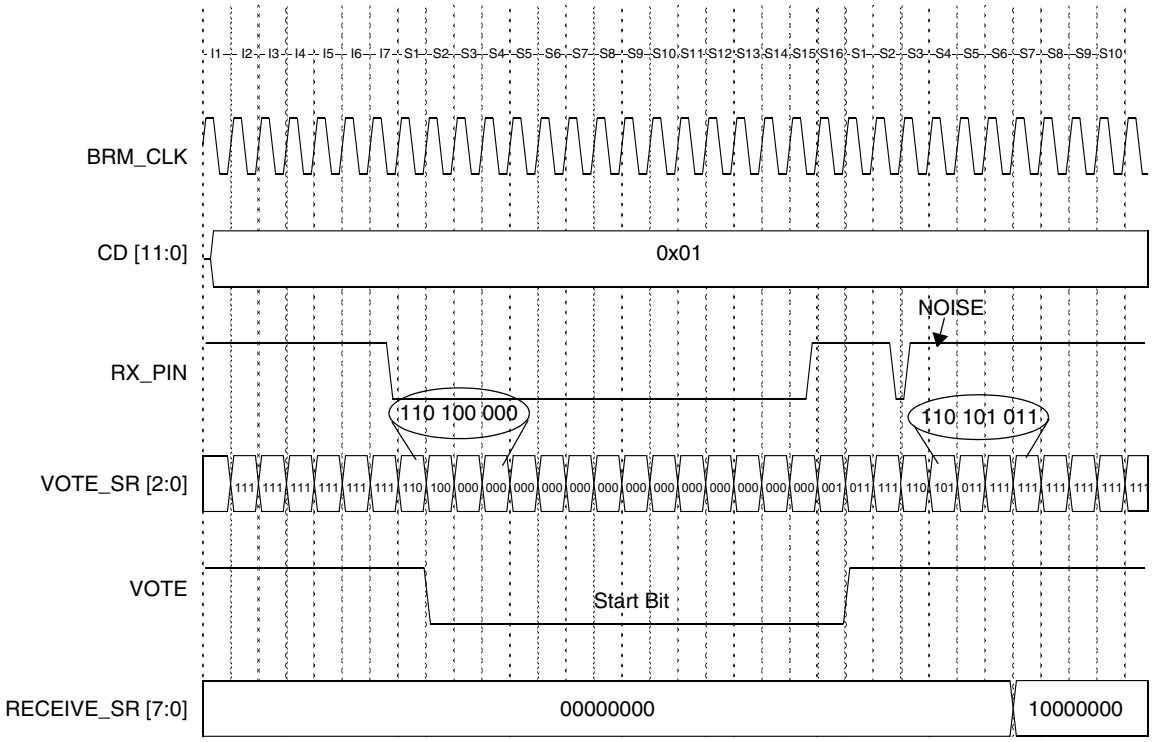


Figure 24-5. Majority Vote Results

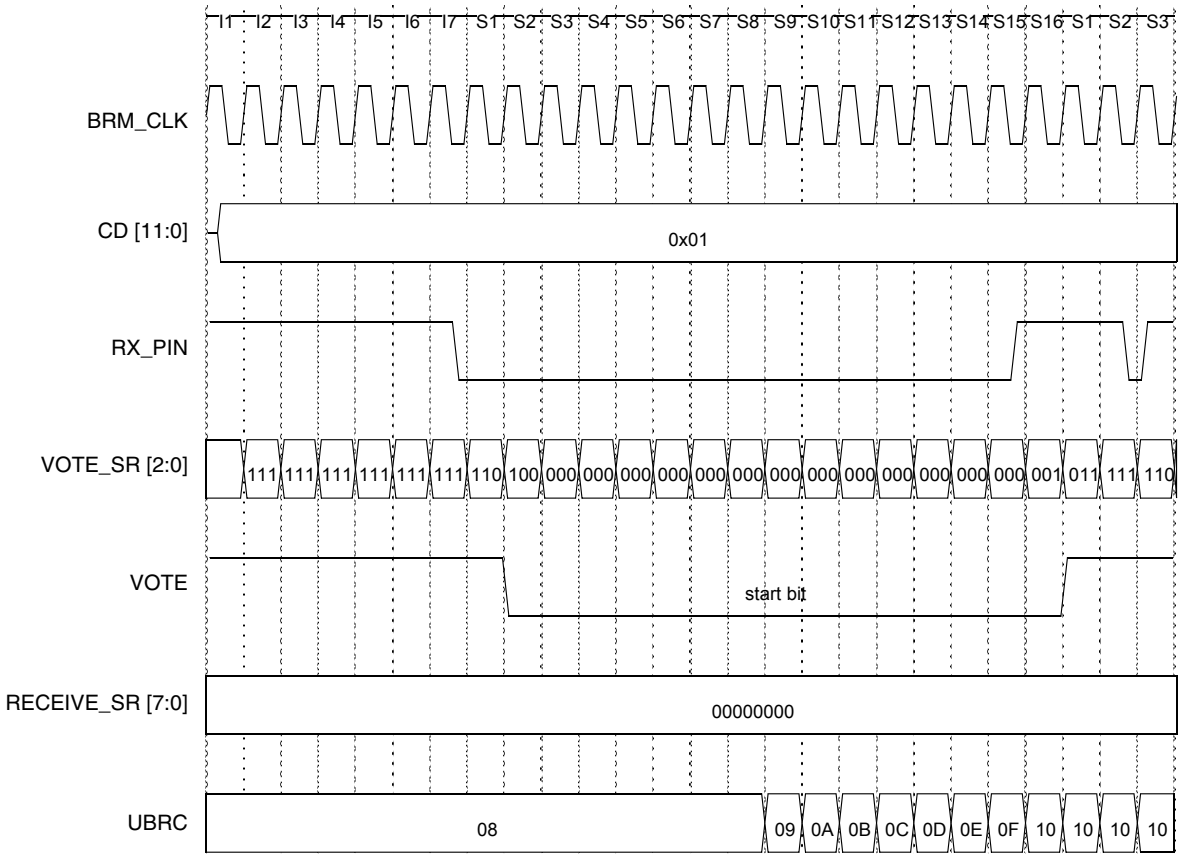


Figure 24-6. Baud Rate Detection of Divisor = 1

Chapter 25
USB Device Port

This chapter describes the USB device port of the MC9328MXL. It also provides configuration, interface description and detailed programming information for designers to achieve the optimum performance from this device.

25.1 Introduction

The Universal Serial Bus specification describes a USB system as having the following three parts:

- A USB host—The bus master that periodically polls peripherals to initiate data transfers. There is only one host on the bus.
- A USB device—A bus slave that communicates only with a USB host. It does not generate bus traffic and only responds to requests from the host.
- A USB Interconnect—A special class of USB devices that add additional connection points to the bus for more USB devices.

From the user's perspective, the USB module hides all direct interaction with the USB protocol. The registers allow the user to enable or disable the module, control the characteristics of individual endpoints, and monitor traffic flow through the module without ever seeing the low level details of the USB protocol.

Even though this module hides all direct interaction with the protocol, some knowledge of the USB is required to properly configure the device for operation on the bus. Programming requirements are covered in this chapter.

25.1.1 Features

The USB device module on the MC9328MXL provides the following USB features:

- Complies with Universal Serial Bus Specification Revision 1.1.
- Endpoint configurations are shown in Table 25-1. Six pipes are available for mapping. Endpoint 0 is required by the USB specification, however all other endpoints are optional. While endpoints 1, 2, 3, 4, and 5 can be configured as Bulk, Interrupt or Isochronous pipes (IN or OUT).
- Control, bulk and interrupt pipes are supported. The packet sizes are limited to 8, 16, 32, or 64 bytes, and the maximum packet size depends on the endpoint's FIFO size.
- Isochronous communications pipes are also supported. A frame match interrupt feature that notifies the user when a specific USB frame occurs is supported. For DMA access, the maximum packet size for the isochronous endpoint is restricted by the endpoint's FIFO size. For programmed I/O, isochronous data packets can take any size from 0 to 1023 bytes.
- Remote wake-up feature is supported through a register bit.
- The USB module operation is operates in both bus-powered and self-powered mode.
- Full speed (12 Mbps) operation.

Table 25-1. Endpoint Configurations

Endpoint	Type	Physical FIFO Size	Endpoint Configuration	Comments
0	IN and OUT	32 bytes	Control	Mandatory
1	IN or OUT	64 bytes	Ctrl, Int, Bulk, or Iso	Optional
2	IN or OUT	64 bytes	Ctrl, Int, Bulk, or Iso	Optional
3	IN or OUT	32 bytes	Ctrl, Int, Bulk, or Iso	Optional
4	IN or OUT	32 bytes	Ctrl, Int, Bulk, or Iso	Optional
5	IN or OUT	32 bytes	Ctrl, Int, Bulk, or Iso	Optional

Table 25-2. USB Specific Interrupts

Name of Interrupt (from Interrupt Assignment Table)	Description	Associated Status Register
$\overline{\text{USBD_INT[0]}}$	Endpoint 0 Interrupt	USB_EP0_INTR
$\overline{\text{USBD_INT[1]}}$	Endpoint 1 Interrupt	USB_EP1_INTR
$\overline{\text{USBD_INT[2]}}$	Endpoint 2 Interrupt	USB_EP2_INTR
$\overline{\text{USBD_INT[3]}}$	Endpoint 3 Interrupt	USB_EP3_INTR
$\overline{\text{USBD_INT[4]}}$	Endpoint 4 Interrupt	USB_EP4_INTR
$\overline{\text{USBD_INT[5]}}$	Endpoint 5 Interrupt	USB_EP5_INTR
$\overline{\text{USBD_INT[6]}}$	USBd Interrupt	USB_INTR

25.2 Module Components

A block diagram of the complete USB Device is shown in Figure 25-1. This section briefly describes each of the components within the module.

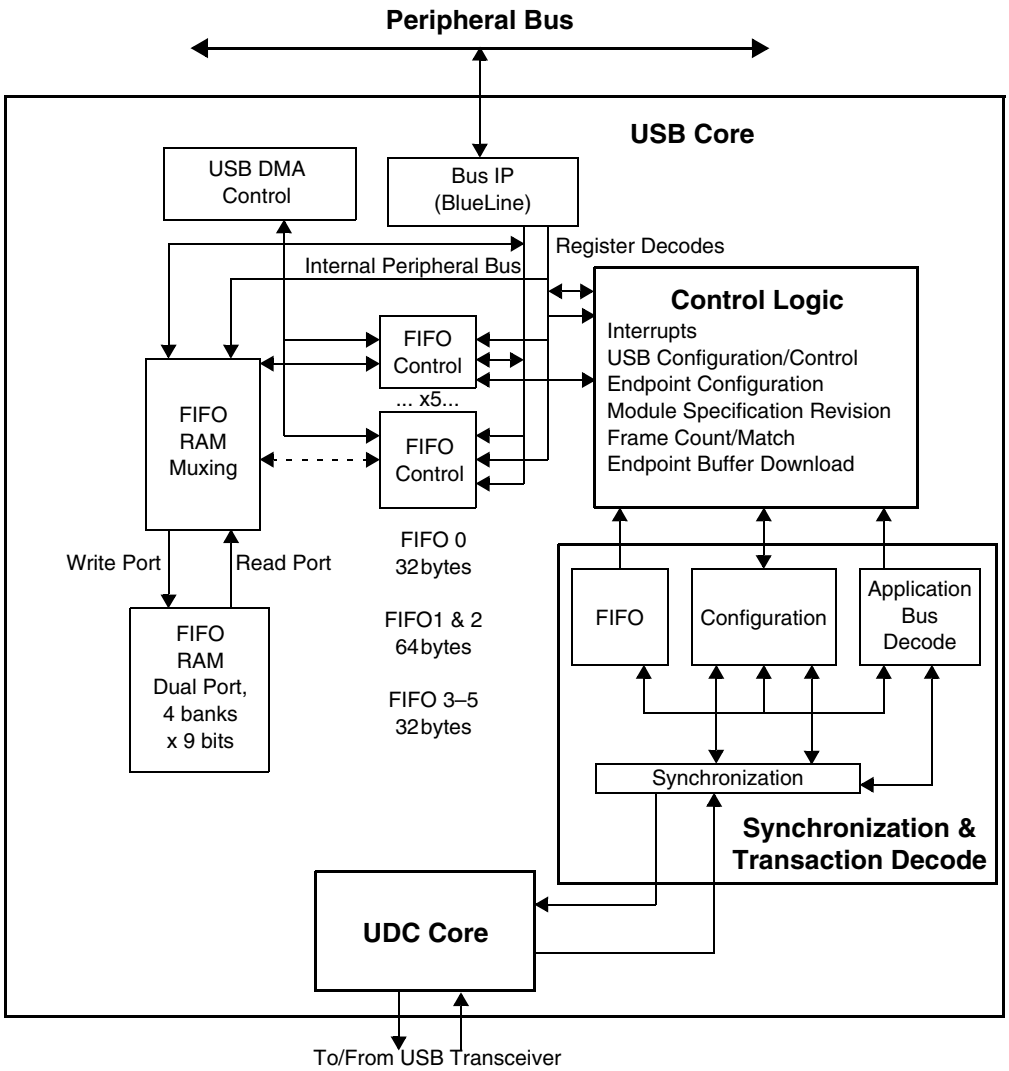


Figure 25-1. USB Device Module Block Diagram

25.2.1 Universal Serial Bus Device Controller

The Universal Serial Bus Device Controller Core (UDC) interfaces the USB function device to the Universal Serial Bus. The UDC handles all the USB protocol and provides a simple read/write protocol on the function interface (application bus). The UDC handles all details of managing the USB protocol and presents a simple set of handshakes to the application for managing data flow, vendor commands, and configuration information. It provides the following features:

- Complies with USB Specification revision 1.1
- Supports USB protocol handling
- Requires no microcontroller or firmware support
- Provides USB device state handling
- Enables clock and data recovery from USB
- Supports bit stripping and bit stuffing functions

- Supports CRC5 checking and CRC16 generation and checking
- Provides serial to parallel data conversion
- Maintains data synchronization bits (DATA0/DATA1) toggle bits
- Understands and decodes standard USB commands to endpoint 0

25.2.2 Synchronization and Transaction Decode

The synchronization and transaction decode block performs two functions:

- It synchronizes the front-end logic timing to the UDC's application bus timing. The front-end logic is targeted for a maximum of 96 MHz operation, while the UDC's application bus runs at 12 MHz for full-speed devices.
- The synchronization layer contains a transaction decoder. The application bus protocol is very simple and makes no distinction between RAM, FIFO, and configuration access. The decoder examines the type of transaction requested by the UDC and generates control signals appropriate to that transaction type (RAM, FIFO, and so on).

The transaction decoder ensures that all packet transfers occur in units of the maximum packet size for the selected endpoint. This block decodes the buffer address from the UDC's application bus to determine the maximum packet size for the current endpoint. It also looks at bytes free and end-of-frame (EOF) information from the FIFO module to determine when a packet transfer occurs.

In general, all transfers are of the maximum packet size *except* when all of these conditions apply:

- The endpoint is isochronous
- The transmit FIFO has less than the maximum packet size worth of data available
- There is an end-of-frame indicator in the FIFO

The transaction decoder also handles hardware retries of USB packets containing errors. The hardware is capable of retransmitting an IN packet to the host or discarding an OUT packet from the host.

The synchronization and transaction decode section contains logic related to the UDC module's clock enable. The UDC module is designed with low-power operation in mind. It includes a gated clock and part of the enable logic for that clock with low-power operation in mind.

25.2.3 Endpoint FIFO Architecture

The USB protocol has some specialized requirements that affect FIFO implementation and essentially require one FIFO per USB endpoint. The USB host can access any endpoint on the function in any order, even the same endpoint in back-to-back transactions, however there is a latency requirement that means the USB device must respond to the USB host within a certain number of USB bit times or the device loses its time slice on the USB until some point in the future. To achieve maximum USB bandwidth use, the USB device must provide full packets of data to the USB host immediately on request and receive full packets from the host on request. This requirement results in one FIFO per USB endpoint.

Depending on the traffic requirements, the FIFO sizes are adjustable to support double buffering. Typically, bulk and isochronous endpoints are double buffered, while interrupt and control endpoints usually are single buffered.

Six endpoint FIFOs are available in the USB device module:

- FIFO 0, 32 bytes
- FIFO 1, 64 bytes

- FIFO 2, 64 bytes
- FIFO 3, 32 bytes
- FIFO 4, 32 bytes
- FIFO 5, 32 bytes

25.2.4 Control Logic

The USB module's control logic section implements the logic and registers that allow the user to control and communicate with the USB module. The register functions include interrupt status/mask, USB configuration download, FIFO control and access, and device request processing from the control endpoint.

25.2.5 USB Transceiver Interface

The MC9328MXL provides support for a low-cost external USB transceiver interface. Generic USB transceivers that support 3.0 V I/O levels are supported. The transceiver interface signals are illustrated in Figure 25-2 and each signal is described in the following section.

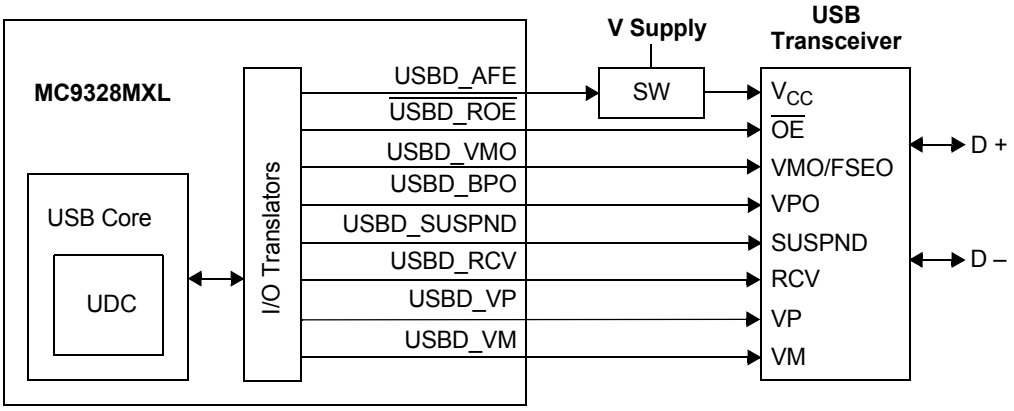


Figure 25-2. USB Module Transceiver Interface

25.2.6 Signal Description

The USB module has seven signals that can be used to interface with the external USB driver.

- **USBD_AFE**—Analog Front-End Enable. This signal is used to enable the front-end transceiver (optional).
- **USBD_ROE**—Reverse Output Enable. This active low signal is used to control the transceiver to drive its D+/D- signal (to the host connection) according to the signal in USBD_VPO and USBD_VMO (output signal), respectively.
- **USBD_VMO/USBD_VPO**—USB Module Data Output. These signals provide single ended data to the USB Transceiver Transmitter differential driver.
- **USBD_SUSPND**—Transceiver Suspend Enable. This signal, when high, activates a low-power state in the USB transceiver. Normally, when suspended, the transceiver drives USBD_RCV low and tri-state the USB bus signals D+ and D-.
- **USBD_RCV**—USB Module Receive Data. This signal is a CMOS level driven signal provided by the external USB transceiver. The signal is derived from the D+ and D- differential input to the transceiver.

- **USBD_VP**—Input D+ signal connected directly to the D+.
- **USBD_VM**—Input D- signal connected directly to the D-.

25.2.7 Pin Configuration for USB

Table 25-3 lists the pins used by the USB module. These pins are multiplexed with other functions on the device and must be configured for USB operation.

NOTE:

The user must ensure that the data direction bits in the GPIO are set to the correct direction for proper operation. See Section 29.5.1, “Data Direction Registers,” on page 29-8 for details.

Table 25-3. Pin Configuration for USB Module

Pin	Setting	Configuration Procedure
USBD_AFE	Primary function of GPIO Port B [20]	1.Clear bit 20 of Port B GPIO In Use Register (GIUS_B) 2.Clear bit 20 of Port B General Purpose Register (GPR_B)
USBD_SUSPND	Primary function of GPIO Port B [23]	1.Clear bit 23 of Port B GPIO In Use Register (GIUS_B) 2.Clear bit 23 of Port B General Purpose Register (GPR_B)
USBD_VMO	Primary function of GPIO Port B [27]	1.Clear bit 27 of Port B GPIO In Use Register (GIUS_B) 2.Clear bit 27 of Port B General Purpose Register (GPR_B)
USBD_VPO	Primary function of GPIO Port B [26]	1.Clear bit 26 of Port B GPIO In Use Register (GIUS_B) 2.Clear bit 26 of Port B General Purpose Register (GPR_B)
USBD_$\overline{\text{ROE}}$	Primary function of GPIO Port B [21]	1.Clear bit 21 of Port B GPIO In Use Register (GIUS_B) 2.Clear bit 21 of Port B General Purpose Register (GPR_B)
USBD_VM	Primary function of GPIO Port B [25]	1.Clear bit 25 of Port B GPIO In Use Register (GIUS_B) 2.Clear bit 25 of Port B General Purpose Register (GPR_B)
USBD_VP	Primary function of GPIOP Port B [24]	1.Clear bit 24 of Port B GPIO In Use Register (GIUS_B) 2.Clear bit 24 of Port B General Purpose Register (GPR_B)
USBD_RCV	Primary function of GPIO Port B [22]	1.Clear bit 22 of Port B GPIO In Use Register (GIUS_B) 2.Clear bit 22 of Port B General Purpose Register (GPR_B)

25.3 Programming Model

The USB module includes 20 required and 55 optional 32-bit registers. Table 25-4 summarizes these registers and their addresses.

Table 25-4. USB Module Register Memory Map

Description	Name	Address
USB Frame	USB_FRAME	0x00212000
USB Specification	USB_SPEC	0x00212004
USB Status	USB_STAT	0x00212008

Table 25-4. USB Module Register Memory Map (continued)

Description	Name	Address
USB Control	USB_CTRL	0x0021200C
USB Descriptor RAM Address	USB_DADR	0x00212010
USB Descriptor RAM/Endpoint Buffer Data	USB_DDAT	0x00212014
USB Interrupt	USB_INTR	0x00212018
USB Interrupt Mask	USB_MASK	0x0021201C
USB Enable	USB_ENAB	0x00212024
Endpoint n Status/Control	USB_EPn_STAT	0x00212030+ (n*0x30)[1]
Endpoint n Interrupt Status	USB_EPn_INTR	0x00212034+ (n*0x30)[1]
Endpoint n Interrupt Mask	USB_EPn_MASK	0x00212038+ (n*0x30)[1]
Endpoint n FIFO Data	USB_EPn_FDAT	0x0021203C+ (n*0x30)[1]
Endpoint n FIFO Status	USB_EPn_FSTAT	0x00212040+ (n*0x30)[1]
Endpoint n FIFO Control	USB_EPn_FCTRL	0x00212044+ (n*0x30)[1]
USB Endpoint n Last Read Frame Pointer	USB_EPn_LRFP	0x00212048+ (n*0x30)[1]
USB Endpoint n Last Write Frame Pointer	USB_EPn_LWFP	0x0021204C+ (n*0x30)[1]
Endpoint n FIFO Alarm	USB_EPn_FALRM	0x00212050+ (n*0x30)[1]
Endpoint n FIFO Read Pointer	USB_EPn_FRDP	0x00212054+ (n*0x30)[1]
Endpoint n FIFO Write Pointer	USB_EPn_FWRP	0x00212058+ (n*0x30)[1]

1. The parameter ‘n’ refers to the number of endpoints programmed into this device through MPP software in RTL and ranges from 0 to 5.

25.3.1 USB Frame Register

The USB Frame Number and Match register (USB_FRAME) is used to detect a specific frame.

USB_FRAME USB Frame **Addr 0x00212000**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
						MATCH										
TYPE	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						FRAME										
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 25-5. USB Frame Register Description

Name	Description
Reserved Bits 31–27	Reserved—These bits are reserved and should read 0.
MATCH Bits 26–16	**Match Field**—Sets compare value for FRAME_MATCH interrupt. When the value in the FRAME field equals the value in the MATCH field, a FRAME_MATCH interrupt is generated (if not masked).
Reserved Bits 15–11	Reserved—These bits are reserved and should read 0.
FRAME Bits 10–0	**Frame Field**—Holds the frame number decoded from the SOF (Start of Frame) packet that leads each USB frame.

25.3.2 USB Specification Register

The read-only USB Specification and Release Number register (USB_SPEC) stores information about the version of the USB specification with which the USB module complies.

USB_SPEC USB Specification **Addr 0x00212004**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					SPEC											
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
								0x0110								

Table 25-6. USB Specification Register Description

Name	Description	
Reserved Bits 31–12	Reserved—These bits are reserved and should be set to 0.	
SPEC Bits 11–0	**Specification Number**—Contains the version of USB specification with which the underlying USB core complies.	These 12 bits represent the version number of the specification. 0x110 = version 1.1

25.3.3 USB Status Register

The read-only USB Status register (USB_STAT) reports the current state of various features of the USB module. Certain bits are used only for hardware debug mode and always read 0 when debug mode is not enabled.

USB_STAT USB Status **Addr 0x00212008**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								RST	SUSP	CFG		INTF		ALTSET		
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 25-7. USB Status Register Description

Name	Description	Settings
Reserved Bits 31–9	Reserved—These bits are reserved and should read 0.	
RST Bit 8	**Reset Signaling**—Indicates type of reset signaling.	0 = Normal signaling in progress on USB 1 = Reset signaling in progress on USB
SUSP Bit 7	**Suspend**—Indicates USB suspend condition.	0 = USB is not suspended 1 = USB is suspended
CFG Bits 6–5	**Configuration**—Represents the currently selected USB configuration. See Section 25.5.1, “Configuration Download.”	
INTF Bits 4–3	**Interface**—Identifies the USB interface in the current configuration that is associated with the Alternative Interface Indicator (AINTF).	
ALTSET Bits 2–0	**Alternate Setting**—Contains the currently selected USB alternate setting.	

25.3.4 USB Control Register

The USB Control register (USB_CTRL) configures numerous features of the USB module.

USB_CTRL USB Control **Addr 0x0021200C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										CMD_ OVER	CMD_ ERROR	USB_ SPD	USB_ ENA	UDC_ RST	AFE_ ENA	RESUME
TYPE	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	w	rw	w
RESET	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	0x0010															

Table 25-8. USB Control Register Description

Name	Description	Settings
Reserved Bits 31–7	Reserved—These bits are reserved and should read 0.	
CMD_OVER Bit 6	**Command Over**—Indicates status of command processing. See Table 25-9 for more information. CMD_OVER clears automatically after the UDC core has completed the status phase of a control transfer.	0 = Command complete 1 = Command in process
CMD_ERROR Bit 5	**Command Error**—Indicates if an error was encountered during processing of a device request. See Table 25-9 for more information. CMD_OVER and CMD_ERROR combine to create the handshaking code for the status phase of a device request transaction.	0 = If the command was processed, there was no error 1 = If the command was processed, an error occurred
USB_SPD Bit 4	**USB Speed**—Sets the operating speed for the USB module. **Note:** The USB module supports only full speed operation of the device. Attempts to set to slow speed results in unpredictable operation.	0 = Low speed 1 = Full speed
USB_ENA Bit 3	**USB Enable**—Determines whether the USB module responds to requests from the USB host. The USB module comes out of reset in the disabled state. The user must ensure that the USB endpoint configuration and USB registers are programmed appropriately before enabling communications. USB_ENA does not affect the underlying UDC core, only the front-end logic's ability to communicate with the core.	0 = USB module front-end logic is disabled. All transactions to or from the UDC are ignored. 1 = USB module front-end logic is enabled and ready to communicate with the host.

Table 25-8. USB Control Register Description (continued)

Name	Description	Settings
UDC_RST Bit 2	**UDC Reset**—Executes a hard reset sequence on the UDC module in accordance with the UDC specification. It allows the system software to force a reset of the UDC's logic when the system is first connected to the USB, or for debugging purposes.	0 = No effect 1 = Hard reset of the UDC
AFE_ENA Bit 1	**Analog Front-End Enable**—Enables/Disables the analog front-end to the USB. This can be either an external chip or an on-chip transceiver module. Use AFE_ENA to power-down the AFE.	0 = AFE disabled 1 = AFE enabled
RESUME Bit 0	**Resume**—Initiates resume signaling on the USB. Automatically resets to 0 after a write. Remote wake-up capability is controlled in the UDC through the CLEAR_FEATURE request. Software must have a time-out feature that aborts the remote wake-up attempt when the RESUME interrupt does not occur after a specified time.	0 = No effect 1 = Initiate resume signaling on the bus when the remote wake-up capability is enabled for the current USB configuration. When the remote wake-up capability is disabled, RESUME has no effect.

Table 25-9. Device Request Status

Result of Transfer	CMD_OVER	CMD_ERROR
Application processed the device request successfully	0	0
Application encountered an error while processing the request	0	1
Application is busy completing the request	1	X

25.3.5 USB Descriptor RAM Address Register

This register allows user access to the USB descriptor RAM. The user programs a desired address into the 9-bit desired RAM address (DADR) field and follows it with a read or write to the USB_DDAT register to complete the access. On read/write access to the USB Descriptor RAM/Endpoint Buffer Data (USB_DDAT), the address in the DADR field increments automatically. When the CFG bit is set to 1, the DADR address is ignored.

USB_DADR USB Descriptor RAM Address **Addr 0x00212010**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CFG	BSY														
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x8000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								DADR								
TYPE	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 25-10. USB Descriptor RAM Address Register Description

Name	Description	Settings
CFG Bit 31	**Configuration**—Determines the function of USB_DDAT Register. The Configuration versus Descriptor access indicator is set automatically at power-on or hard reset and clears after the last byte of endpoint buffer configuration data is downloaded into the UDC.	0 = The USB_DDAT register is set to access the descriptor storage RAM. Configuration load has completed. The USB_DDAT writes have no effect. 1 = The USB_DDAT register is set to download endpoint buffer configuration data to the UDC.
BSY Bit 30	**Busy**—Indicates if a write is in progress to the endpoint buffer. Because the front-end logic and the UDC module operate on different clocks, the configuration download interface busy signal is provided to ensure that writes from the USB_DDAT register have sufficient time to successfully enter the UDC's clock domain.	0 = No write is in progress. 1 = A write is in progress to the UDC module's endpoint buffer. Attempt no other operations on the USB Core module until BSY has cleared.
Reserved Bits 29–9	Reserved—These bits are reserved and should read 0.	
DADR Bits 8–0	**Desired RAM Address**—Holds the desired RAM address. The user programs a desired descriptor RAM address into the DADR field and follows it with a read or write to the USB_DDAT register to complete the access.	

25.3.6 USB Descriptor RAM/Endpoint Buffer Data Register

This register allows user access to the endpoint buffer download facility. For endpoint buffer access, when the CFG bit in the USB_DADR register is set, writes to this register cause the data to be loaded into the UDC module's endpoint buffers, and reads are undefined. When the CFG bit in the USB_DADR register is cleared, writes have no effect and reads are zero.

Access to this register is allowed only when the USB_ENA bit in the USB Control (USB_CTRL) is set to 0. Access at other times is ignored and reads are undefined.

USB_DDAT USB Descriptor RAM/Endpoint Buffer Data **Addr 0x00212014**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									DDAT							
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 25-11. USB Descriptor RAM/Endpoint Buffer Data Register Description

Name	Description
Reserved Bits 31–8	Reserved—These bits are reserved and should read 0.
DDAT Bits 7–0	**DDAT**—Allows user access to the endpoint buffer in the UDC module when the endpoint configuration is happening. (When the CFG bit in the USB_DADR is set) Writing to this field loads the data written into the UDC endpoint buffers.

25.3.7 USB Interrupt Register

The USB Interrupt register (USB_INTR) maintains the status of interrupt conditions that pertain to USB functions.

An interrupt, when set, remains set until 1 is written to the corresponding bit. Interrupts do not clear automatically when the event that caused them goes away. For example, when reset signaling comes and goes with no intervention from software, both RESET_START and RESET_STOP are set and stay set. Writing 0 to this register has no effect.

If a register write occurs at the same time an interrupt is received, the interrupt takes precedence over the write.

USB_INTR USB Interrupt **Addr 0x00212018**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	WAKEUP															
TYPE	rw	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
										0x0000						

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									MSOF	SOF	RESET _STOP	RESET _START	RES	SUSP	FRAME _MATCH	CFG_ CHG
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
										0x0000						

Table 25-12. USB Interrupt Register Description

Name	Description	Settings
WAKEUP Bit 31	**Wakeup**—Indicates a state change from suspend to resume (wakeup) in the UDC module. Clearing the interrupt has no effect on the actual state of the USB module (WAKEUP can be written to even when the module is disabled.) **Note:** Use the asynchronous WAKEUP interrupt to power-down the module clocks and to power-up the USB module.	0 = No state change since last interrupt 1 = USB has changed state from suspend to resume (wakeup)
Reserved Bits 30–8	Reserved—These bits are reserved and should read 0.	
MSOF Bit 7	**Missed Start-of-Frame Interrupt**—Indicates if a start-of-frame was missed.	0 = No missed start-of-frame 1 = A SOF interrupt was set, however not cleared before the next one was received
SOF Bit 6	**Start-of-Frame Interrupt**—Indicates if a start-of-frame was received.	0 = No start-of-frame received 1 = The UDE module received a start-of-frame token (the USB frame number is current)

Table 25-12. USB Interrupt Register Description (continued)

Name	Description	Settings
RESET_STOP Bit 5	**Reset Signaling Stop**—Indicates the end of reset signaling on the USB.	0 = Reset signaling has not stopped (does not imply that reset signaling is occurring, just that no end-of-reset event has occurred) 1 = Reset signaling has stopped
RESET_START Bit 4	**Reset Signaling Start**—Indicates start of reset signaling on the USB.	0 = Reset signaling has not started (does not imply that reset signaling is occurring, but only that no start of reset event has occurred) 1 = Reset signaling in progress
RES Bit 3	**Suspend to Resume**—Indicates a change of state from suspend to resume in the UDC module (indicates only the change from suspended to active mode). Clearing the interrupt has no effect on the actual state of the USB.	0 = The USB has not left the suspended state (does not imply that the bus is, or ever was, suspended) 1 = USB has left the suspend state
SUSP Bit 2	**Active to Suspend**—Indicates the suspend state of the UDC module (indicates only the change from active to suspended mode). Clearing the interrupt has no effect on the actual state of the USB.	0 = USB is not suspended, or the interrupt was cleared 1 = USB is suspended
FRAME_MATCH Bit 1	**Frame Match**—Indicates whether there is a match between the USB frame number and the value in the USB Frame.	0 = No match occurred 1 = Match occurred
CFG_CHG Bit 0	**Configuration Change**—Indicates if a change occurred in the USB configuration (configuration, interface, alternate) which requires the software to reread the USB Status.	0 = No configuration change occurred 1 = Configuration change occurred

25.3.8 USB Interrupt Mask Register

The USB Interrupt Mask register (USB_MASK) is used to mask the corresponding interrupt in the USB_INTR register.

USB_MASK USB Interrupt Mask **Addr 0x0021201C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	WAKEUP															
TYPE	rw	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x8000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									MSOF	SOF	RESET_ STOP	RESET_ START	RES	SUSP	FRAME_ MATCH	CFG_ CHG
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	0x00FF															

Table 25-13. USB Interrupt Mask Register Description

Name	Description	Settings
WAKEUP Bit 31	**Wakeup Mask**—Enables/Disables the WAKEUP interrupt.	0 = Interrupt enabled (unmasked) 1 = Interrupt disabled (masked)
Reserved Bits 30–8	Reserved—These bits are reserved and should read 0.	
MSOF Bit 7	**Missed Start-of-Frame Mask**—Enables/Disables the MSOF interrupt.	0 = Interrupt enabled (unmasked) 1 = Interrupt disabled (masked)
SOF Bit 6	**Start-of-Frame Mask**—Enables/Disables the SOF interrupt.	0 = Interrupt enabled (unmasked) 1 = Interrupt disabled (masked)
RESET_STOP Bit 5	**Reset Signaling Stop Mask**—Enables/Disables the RESET_STOP interrupt.	0 = Interrupt enabled (unmasked) 1 = Interrupt disabled (masked)
RESET_START Bit 4	**Reset Signaling Start Mask**—Enables/Disables the RESET_START interrupt.	0 = Interrupt enabled (unmasked) 1 = Interrupt disabled (masked)
RES Bit 3	**Suspend to Resume Mask**—Enables/Disables the RES interrupt.	0 = Interrupt enabled (unmasked) 1 = Interrupt disabled (masked)
SUSP Bit 2	**Suspend Mask**—Enables/Disables the SUSP interrupt.	0 = Interrupt enabled (unmasked) 1 = Interrupt disabled (masked)
FRAME_MATCH Bit 1	**Frame Match Mask**—Enables/Disables the FRAME_MATCH interrupt.	0 = Interrupt enabled (unmasked) 1 = Interrupt disabled (masked)
CFG_CHG Bit 0	**Configuration Change Mask**—Enables/Disables the CFG_CHG interrupt.	0 = Interrupt enabled (unmasked) 1 = Interrupt disabled (masked)

25.3.9 USB Enable Register

The USB Enable register (USB_ENAB) controls the operation of the USB functions.

USB_ENAB USB Enable **Addr 0x00212024**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RST	ENAB	SUSPEND	ENDIAN_MODE												
TYPE	rw	rw	r	r/w	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	0x1000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																PWRMD
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	0x0001															

Table 25-14. USB Enable Register Description

Name	Description	Settings
RST Bit 31	**Reset**—Indicates the USB's software reset state. Automatically clears after the USB module is reset. **Note:** Setting RST automatically sets the ENAB bit.	0 = No USB reset in progress 1 = USB reset in progress
ENAB Bit 30	**Enable**—Indicates the USB's enable state. When the USB is disabled, all write access to the USB registers is ignored except for writes to ENAB and to the WAKEUP bit in the USB_INTR register.	0 = Disable the USB 1 = Enable the USB
SUSPEND Bit 29	**Suspend**—Indicates whether the UDC module is in the suspend state.	0 = Module is in resume/active state 1 = Module is in suspend state
ENDIAN_MODE Bit 28	**Endian Mode Select**—Selects whether the USB module is in Big Endian mode or Little Endian mode.	1 = USBd module is in Little Endian mode 0 = USBd module is in Big Endian mode
Reserved Bits 27–1	Reserved—These bits are reserved and should read 0.	
PWRMD Bit 0	**Power Mode—**Determines the power mode of USBD. The default power mode is self powered.	0 = Bus powered mode 1 = Self-powered mode

25.3.10 Endpoint n Status/Control Registers

The Endpoint n Status/Control registers allow the user to configure the endpoints and contains a field for maintaining a count of the FIFO contents.

The number of Endpoint n Status/Controls in the MC9328MXL depends on the number of endpoints configured.

		Addr
USB_EP0_STAT	Endpoint 0 Status/Control Register	**0x00212030**
USB_EP1_STAT	Endpoint 1 Status/Control Register	**0x00212060**
USB_EP2_STAT	Endpoint 2 Status/Control Register	**0x00212090**
USB_EP3_STAT	Endpoint 3 Status/Control Register	**0x002120C0**
USB_EP4_STAT	Endpoint 4 Status/Control Register	**0x002120F0**
USB_EP5_STAT	Endpoint 5 Status/Control Register	**0x00212120**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
										BYTE_COUNT						
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								SIP	DIR	MAX		TYP		ZLPS	FLUSH	FORCE_ STALL
TYPE	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	w	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 25-15. Endpoint n Status/Control Register Description

Name	Description	Settings
Reserved Bits 31–23	Reserved—These bits are reserved and should read 0.	
BYTE_COUNT Bits 22–16	**Byte Count**—Represents the number of bytes currently stored in the associated FIFO.	
Reserved Bits 15–9	Reserved—These bits are reserved and should read 0.	
SIP Bit 8	**Setup Packet In Progress**—Indicates whether the setup packet is being transferred from host to device.	0 = Setup data is not being transferred from host to device 1 = Setup data is currently being transferred from host to device
DIR Bit 7	**Transfer Direction**—Sets the endpoint direction (DIR is ignored for control endpoints).	0 = OUT endpoint (from host to device) 1 = IN endpoint (from device to host)

Table 25-15. Endpoint n Status/Control Register Description (continued)

Name	Description	Settings
MAX Bits 6–5	**Maximum Packet Size**—Sets the maximum packet size for the endpoint (MAX is ignored for -ronous endpoints). **Note:** The maximum packet size cannot be greater than the endpoint's FIFO size. See Table 25-1 for FIFO sizes.	00 = Packet is 8 Bytes 01 = Packet is 16 Bytes 10 = Packet is 32 Bytes 11 = Packet is 64 Bytes
TYP Bits 4–3	**Endpoint Type**—Sets up the type of endpoint being used. Endpoint 0 is defined as a control endpoint, however all other endpoints can be any type.	00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt
ZLPS Bit 2	**Zero Length Packet Send**—Determines if a zero length packet will be sent to the host. If the FIFO is empty and the USB host requests an IN transaction, the USB module can send a zero length packet in response. ZLPS automatically clears after the transaction completes successfully. ZLPS signifies to the USB host that the end of data was reached in a data transmission when the end of data lands on a packet boundary and there is no short packet to signify the end of data.	0 = No zero length packet to send 1 = A zero length packet to send
FLUSH Bit 1	**Flush**—Flushes the associated FIFO to its empty state.	0 = Do nothing 1 = Initiate flush operation
FORCE_STALL Bit 0	**Force a Stall Condition**—Causes the endpoint to respond with a STALL to the next poll, and automatically clears when the stall takes effect. **Note:** There is no endpoint stalled indicator because one is not returned from the UDC. The USB host is expected to communicate with the USB device through device requests to fix the stall condition. The USB host sends a CLEAR_FEATURE request to the UDC module to clear the stall and resume normal operations.	0 = Do nothing 1 = Force a stall condition

25.3.11 Endpoint n Interrupt Status Registers

The Endpoint n Interrupt Status registers monitor the status of a specific endpoint and generates CPU interrupts each time a monitored event occurs.

When an interrupt is set, it remains set until cleared by writing 1 to the corresponding bit. Interrupts do not clear automatically when the event that caused them goes away (for example, when reset signaling comes and goes with no intervention from software, both RESET_START and RESET_STOP are set). Writing 0 has no effect.

If a register write occurs at the same time an interrupt is received, the interrupt takes precedence over the write.

The number of Endpoint n Interrupt Statuss in the MC9328MXL depends on the number of endpoints configured.

		Addr
USB_EP0_INTR	Endpoint 0 Interrupt Status Register	**0x00212034**
USB_EP1_INTR	Endpoint 1 Interrupt Status Register	**0x00212064**
USB_EP2_INTR	Endpoint 2 Interrupt Status Register	**0x00212094**
USB_EP3_INTR	Endpoint 3 Interrupt Status Register	**0x002120C4**
USB_EP4_INTR	Endpoint 4 Interrupt Status Register	**0x002120F4**
USB_EP5_INTR	Endpoint 5 Interrupt Status Register	**0x00212124**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
									0x0000							

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								FIFO_ FULL	FIFO_ EMPTY	FIFO_ ERROR	FIFO_ HIGH	FIFO_ LOW	MDEV REQ	EOT	DEV REQ	EOF
TYPE	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
									0x0080							

Table 25-16. Endpoint n Interrupt Status Register Description

Name	Description	Settings
Reserved Bits 31–9	Reserved—These bits are reserved and should read 0.	
FIFO_FULL Bit 8	**FIFO Full**—Indicates whether the FIFO is full or not.	0 = The FIFO is not full 1 = The FIFO is full
FIFO_EMPTY Bit 7	**FIFO Empty**—Indicates whether the FIFO is empty or not.	0 = The FIFO is not empty 1 = The FIFO is empty
FIFO_ERROR Bit 6	**FIFO Error**—Indicates an error condition in the FIFO controller. The specific error condition can be checked by reading the Endpoint n FIFO Status (USB_EPn_FSTAT).	0 = No error condition pending 1 = Error condition pending
FIFO_HIGH Bit 5	**FIFO High**—Asserts when the FIFO has hit a high level alarm. FIFO_HIGH applies only when the FIFO is in receive mode (USB OUT).	0 = The number of data bytes in the FIFO is less than GR value from the Endpoint n FIFO Control 1 = The number of free bytes in the FIFO is less than ALRM value from the Endpoint n FIFO Alarm

Table 25-16. Endpoint n Interrupt Status Register Description (continued)

Name	Description	Settings
FIFO_LOW Bit 4	**FIFO Low**—Asserts when the FIFO has hit a low level alarm. FIFO_LOW applies only when the FIFO is in transmit mode (USB IN).	0 = The number of free bytes in the FIFO is less than 4xGR value from the Endpoint n FIFO Control 1 = The number of data bytes in the FIFO is less than ALRM value from the Endpoint n FIFO Alarm
MDEVREQ Bit 3	**Multiple Device Requests**—Asserts when a DEVREQ interrupt is pending and another setup packet is received. MDEVREQ asserts only for control endpoints.	0 = Multiple setup packets are not pending 1 = Multiple setup packets pending
EOT Bit 2	**End-of-Transfer**—Asserts when the last packet of a USB data transfer has crossed into, or out of, the UDC module. The last packet is identified by its length. Any packet shorter than the maximum packet size for the associated endpoint is considered to be an end-of-transfer marker. The EOT interrupt also asserts (for control endpoints only) when the number of bytes specified in the wLength field of the setup packet has been transferred.	0 = Last packet of data not sent/received 1 = Last packet of data sent/received
DEVREQ Bit 1	**Device Request**—Indicates whether there is a device request on the current endpoint. DEVREQ asserts only for control endpoints.	0 = No request pending 1 = Request pending
EOF Bit 0	**End-of-Frame**—Indicates whether there is end-of-frame activity for this endpoint. EOF monitors the data flow between the FIFO and the UDC and indicates when the end of a USB packet is written into the FIFO or the UDC as the end of a frame.	0 = End-of-frame (USB packet) not sent/received 1 = End-of-frame (USB packet) sent/received

25.3.12 Endpoint n Interrupt Mask Register

The Endpoint n Interrupt Mask registers allow the user to mask individual interrupts for each endpoint. Writing 1 to a bit in this register masks the corresponding interrupt in the USB_EPn_STAT register. Writing 0 unmasks the interrupt.

The number of Endpoint n Interrupt Masks in the MC9328MXL depends on the number of endpoints configured.

		Addr
USB_EP0_MASK	Endpoint 0 Interrupt Mask Register	**0x00212038**
USB_EP1_MASK	Endpoint 1 Interrupt Mask Register	**0x00212068**
USB_EP2_MASK	Endpoint 2 Interrupt Mask Register	**0x00212098**
USB_EP3_MASK	Endpoint 3 Interrupt Mask Register	**0x002120C8**
USB_EP4_MASK	Endpoint 4 Interrupt Mask Register	**0x002120F8**
USB_EP5_MASK	Endpoint 5 Interrupt Mask Register	**0x00212128**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
									0x0000							

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								FIFO_ FULL	FIFO_ EMPTY	FIFO_ ERROR	FIFO_ HIGH	FIFO_ LOW	MDEV REQ	EOT	DEV REQ	EOF
TYPE	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
									0x01FF							

Table 25-17. Endpoint n Interrupt Mask Register Description

Name	Description	Settings
Reserved Bits 31–9	Reserved—These bits are reserved and should read 0.	
FIFO_FULL Bit 8	**FIFO Full Mask**—Enables/Disables the FIFO Full interrupt.	0 = Interrupt enabled (unmasked) 1 = Interrupt disabled (masked)
FIFO_EMPTY Bit 7	**FIFO Empty Mask**—Enables/Disables the FIFO Empty interrupt.	0 = Interrupt enabled (unmasked) 1 = Interrupt disabled (masked)
FIFO_ERROR Bit 6	**FIFO Error Mask**—Enables/Disables the FIFO Error interrupt.	0 = Interrupt enabled (unmasked) 1 = Interrupt disabled (masked)
FIFO_HIGH Bit 5	**FIFO High Alarm Mask**—Enables/Disables the FIFO High Alarm interrupt.	0 = Interrupt enabled (unmasked) 1 = Interrupt disabled (masked)
FIFO_LOW Bit 4	**FIFO Low Alarm Mask**—Enables/Disables the FIFO Low Alarm interrupt.	0 = Interrupt enabled (unmasked) 1 = Interrupt disabled (masked)

Table 25-17. Endpoint n Interrupt Mask Register Description (continued)

Name	Description	Settings
MDEVREQ Bit 3	**Multiple Device Requests Mask**—Enables/Disables the Multiple Device Requests interrupt.	0 = Interrupt enabled (unmasked) 1 = Interrupt disabled (masked)
EOT Bit 2	**End-of-Transfer Mask**—Enables/Disables the End-of-Transfer interrupt.	0 = Interrupt enabled (unmasked) 1 = Interrupt disabled (masked)
DEVREQ Bit 1	**Device Request Mask**—Enables/Disables the Device Request interrupt.	0 = Interrupt enabled (unmasked) 1 = Interrupt disabled (masked)
EOF Bit 0	**End-of-Frame Mask**—Enables/Disables the end-of-frame interrupt.	0 = Interrupt enabled (unmasked) 1 = Interrupt disabled (masked)

25.3.13 Endpoint n FIFO Data Registers

The Endpoint n FIFO Data registers are the main interface port for each FIFO. Data to be buffered in the FIFO, or currently buffered in the FIFO, is accessed through this register. This register can access data from the FIFO independent of the transmit or receive configuration in byte, word, or longword formats, however each access must be aligned with the most significant byte (big endian) of the data port. Byte 0 is bits 31:24, byte 1 is bits 23:16, byte 2 is bits 15:8, and byte 3 is bits 7:0. Byte transfers must access byte 0. Word transfers must access bytes 0 and 1, and longword transfers must access all four bytes. USB FIFO data write is in big-endian format.

The direction of the FIFO is determined by the value of the DIR field in the Endpoint n Status/Control.

The number of Endpoint n FIFO Datas in the MC9328MXL depends on the number of endpoints configured.

		Addr
USB_EP0_FDAT	Endpoint 0 FIFO Data Register	**0x0021203C**
USB_EP1_FDAT	Endpoint 1 FIFO Data Register	**0x0021206C**
USB_EP2_FDAT	Endpoint 2 FIFO Data Register	**0x0021209C**
USB_EP3_FDAT	Endpoint 3 FIFO Data Register	**0x002120CC**
USB_EP4_FDAT	Endpoint 4 FIFO Data Register	**0x002120FC**
USB_EP5_FDAT	Endpoint 5 FIFO Data Register	**0x0021212C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RXDATA [31:16] (Read) or TXDATA [31:16] (Write)															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0X0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RXDATA [15:0] (Read) or TXDATA [15:0] (Write)															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 25-18. Endpoint n FIFO Data Register Description

Name	Description
TXDATA [31:0] Bits 31–0	**Transmit Data**—Contains the transmit FIFO write data.
RXDATA [31:0] Bits 31–0	**Read Data**—Contains the receive FIFO read data.

25.3.14 Endpoint n FIFO Status Registers

The Endpoint n FIFO Status registers are the main status registers for the FIFOs, and contain information about frame status, underflow, overflow, alarm, and FIFO content.

The number of Endpoint n FIFO Status registers in the MC9328MXL depends on the number of endpoints configured.

		Addr
USB_EP0_FSTAT	Endpoint 0 FIFO Status Register	**0x00212040**
USB_EP1_FSTAT	Endpoint 1 FIFO Status Register	**0x00212070**
USB_EP2_FSTAT	Endpoint 2 FIFO Status Register	**0x002120A0**
USB_EP3_FSTAT	Endpoint 3 FIFO Status Register	**0x002120D0**
USB_EP4_FSTAT	Endpoint 4 FIFO Status Register	**0x00212100**
USB_EP5_FSTAT	Endpoint 5 FIFO Status Register	**0x00212130**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
					FRAME0	FRAME1	FRAME2	FRAME3		ERROR	OF	UF	FR	FULL	ALARM	EMPTY
TYPE	r	r	r	r	r	r	r	r	r	rw	rw	rw	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	0x0001															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 25-19. Endpoint n FIFO Status Register Description

Name	Description	Settings
Reserved Bits 31–28	Reserved—These bits are reserved and should read 0.	
FRAME0 Bit 27	**Frame Status Bit 0**—Indicates whether a frame boundary exists in bus bits [31:24] for non-DMA applications.	0 = No frame boundary on the [31:24] byte 1 = A frame boundary has occurred on the [31:24] byte of the bus
FRAME1 Bit 26	**Frame Status Bit 1**—Indicates whether a frame boundary exists in bus bits [23:16] for non-DMA applications.	0 = No frame boundary on the [23:16] byte 1 = A frame boundary has occurred on the [23:16] byte of the bus
FRAME2 Bit 25	**Frame Status Bit 2**—Indicates whether a frame boundary exists in bus bits [15:8] for non-DMA applications.	0 = No frame boundary on the [15:8] byte 1 = A frame boundary has occurred on the [15:8] byte of the bus

Table 25-19. Endpoint n FIFO Status Register Description (continued)

Name	Description	Settings
FRAME3 Bit 24	**Frame Status Bit 3**—Indicates whether a frame boundary exists in bus bits [7:0] for non-DMA applications.	0 = No frame boundary on the [7:0] byte 1 = A frame boundary has occurred on the [7:0] byte of the bus
Reserved Bit 23	Reserved—This bit is reserved and should read 0.	
ERROR Bit 22	**FIFO Error**—Signifies that an error condition has happened in the FIFO controller. Writing 1 to ERROR clears it. Writing 0 has no effect.	0 = No error 1 = Underflow, overflow, pointer out of bounds, or other error condition occurred
UF Bit 21	**FIFO Underflow**—Indicates FIFO underflow status. Writing 1 to UF clears it. Writing 0 has no effect.	0 = No underflow 1 = Read pointer has passed the write pointer
OF Bit 20	**FIFO Overflow**—Indicates FIFO overflow status. Writing 1 to OF clears it. Writing 0 has no effect.	0 = No overflow 1 = Write pointer has passed the read pointer
FR Bit 19	**Frame Ready**—Indicates frame ready status. FR is inactive when the FIFO is not programmed for frame mode.	0 = No complete frames exist in the FIFO 1 = One or more complete frames exists in the FIFO
FULL Bit 18	**FIFO Full**—Indicates FIFO full status. Read FULL to clear it.	0 = The FIFO is not full 1 = The FIFO is full
ALARM Bit 17	**FIFO Alarm**—Indicates FIFO alarm status. The specific alarm condition detected depends on the FIFO direction. The signal relies on the values of the alarm (ALRM) field of the Endpoint n FIFO Alarm and the granularity (GR) field of the Endpoint n FIFO Control. For IN (transmit) FIFOs, the ALARM bit indicates a low level. It asserts when there are less than ALRM bytes of data remaining in the FIFO, and deasserts when there are less than 4 times GR free bytes remaining. When the FIFO is configured to receive (OUT FIFOs), the ALARM bit indicates a high level. It asserts when there are less than ALRM bytes free in the FIFO, and deasserts when there are less than GR bytes of data remaining. This signal is cleared by reading or writing (as appropriate) the FIFO, or by manipulating the FIFO pointers.	0 = The alarm not set 1 = The alarm is set
EMPTY Bit 16	**FIFO Empty**—Indicates FIFO empty status. Write 1 to EMPTY to clear it.	0 = The FIFO is not empty 1 = The FIFO is empty
Reserved Bits 15–0	Reserved—These bits are reserved and should read 0.	

25.3.15 Endpoint n FIFO Control Registers

The Endpoint n FIFO Control registers are the primary control registers regarding data in the FIFO. This includes frame mode and low and high data service requests.

The number of Endpoint n FIFO Controls in the MC9328MXL depends on the number of endpoints configured.

		Addr
USB_EP0_FCTRL	Endpoint 0 FIFO Control Register	**0x00212044**
USB_EP1_FCTRL	Endpoint 1 FIFO Control Register	**0x00212074**
USB_EP2_FCTRL	Endpoint 2 FIFO Control Register	**0x002120A4**
USB_EP3_FCTRL	Endpoint 3 FIFO Control Register	**0x002120D4**
USB_EP4_FCTRL	Endpoint 4 FIFO Control Register	**0x00212104**
USB_EP5_FCTRL	Endpoint 5 FIFO Control Register	**0x00212134**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			WFR		FRAME	GR										
TYPE	r	r	rw	r	rw	rw	rw	rw	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	0x0100															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 25-20. Endpoint n FIFO Control Register Description

Name	Description	Settings
Reserved Bits 31–30	Reserved—These bits are reserved and should read 0.	
WFR Bit 29	**Write Frame End**—Determines the end of the current data frame in the FIFO. Setting WFR means that the next write is the end of the current data frame.	0 = Next write to FIFO data register is not the end-of-frame 1 = Next write to FIFO data register is the end-of-frame
Reserved Bit 28	Reserved—This bit is reserved and should read 0.	

Table 25-20. Endpoint n FIFO Control Register Description (continued)

Name	Description	Settings
FRAME Bit 27	**Frame Mode**—Indicates whether the USB module is in FRAME mode. In FRAME mode, the FIFO uses its internal frame pointer and information from the peripheral to transfer only full frames of data, as defined by the peripheral. Because the controller only keeps a pointer to the end of the last complete frame, a read request can contain more than one frame of data. The MC9328MXL supports only FRAME = 1.	0 = Frame mode disabled 1 = Frame mode enabled
GR Bits 26–24	**Granularity**—Defines the deassertion point for the “high level” and “low level” service requests. See the ALARM field of the Endpoint n FIFO Status for more information. A “high level” service request is deasserted when there are less than GR data bytes remaining in the FIFO. A “low level” service request is deasserted when there are less than four times GR free bytes remaining in the FIFO. The direction, type, and packet size are defined in the Endpoint n Status/Control.	000 = FIFO has 1 data byte or 1 free location 001 = FIFO has 2 data bytes or 2 free locations 010 = FIFO has 3 data bytes or 3 free locations 011 = FIFO has 4 data bytes or 4 free locations 100 = FIFO has 5 data bytes or 5 free locations 101 = FIFO has 6 data bytes or 6 free locations 110 = FIFO has 7 data bytes or 7 free locations 111 = FIFO has 8 data bytes or 8 free locations
Reserved Bits 23–0	Reserved—These bits are reserved and should read 0.	

25.3.16 USB Endpoint n Last Read Frame Pointer Registers

The USB Endpoint n Last Read Frame Pointer registers hold the location of the most recently read frame or the start of the frame currently in transmission.

The number of USB Endpoint n Last Read Frame Pointers in the MC9328MXL depends on the number of endpoints configured.

		Addr
USB_EP0_LRFP	Endpoint 0 Last Read Frame Pointer Register	**0x00212048**
USB_EP1_LRFP	Endpoint 1 Last Read Frame Pointer Register	**0x00212078**
USB_EP2_LRFP	Endpoint 2 Last Read Frame Pointer Register	**0x002120A8**
USB_EP3_LRFP	Endpoint 3 Last Read Frame Pointer Register	**0x002120D8**
USB_EP4_LRFP	Endpoint 4 Last Read Frame Pointer Register	**0x00212108**
USB_EP5_LRFP	Endpoint 5 Last Read Frame Pointer Register	**0x00212138**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											LRFP					
TYPE	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 25-21. USB Endpoint n Last Read Frame Pointer Register Description

Name	Description
Reserved Bits 31–6	Reserved—These bits are reserved and should read 0.
LRFP Bits 5–0	**Last Read Frame Pointer**—Indicates the start of the most recently read frame or the start of the frame currently in transmission. The LRFP can be read and written for debugging purposes. For the frame retransmit function, the LRFP indicates the point to begin retransmission of the data frame. There are no safeguards to prevent retransmitting data that was overwritten. When the FRAME bit is not set, this pointer has no meaning.

25.3.17 USB Endpoint n Last Write Frame Pointer Registers

The USB Endpoint n Last Write Frame Pointer registers hold the location of the most recently written frame.

The number of USB Endpoint n Last Write Frame Pointers in the MC9328MXL depends on the number of endpoints configured.

		Addr
USB_EP0_LWFP	Endpoint 0 Last Write Frame Pointer Register	**0x0021204C**
USB_EP1_LWFP	Endpoint 1 Last Write Frame Pointer Register	**0x0021207C**
USB_EP2_LWFP	Endpoint 2 Last Write Frame Pointer Register	**0x002120AC**
USB_EP3_LWFP	Endpoint 3 Last Write Frame Pointer Register	**0x002120DC**
USB_EP4_LWFP	Endpoint 4 Last Write Frame Pointer Register	**0x0021210C**
USB_EP5_LWFP	Endpoint 5 Last Write Frame Pointer Register	**0x0021213C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											LWFP					
TYPE	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 25-22. USB Endpoint n Last Write Frame Pointer Register Description

Name	Description
Reserved Bits 31–6	Reserved—These bits are reserved and should read 0.
LWFP Bits 5–0	**Last Write Frame Pointer**—Indicates the start of the last frame written into the FIFO. The LWFP can be read and written for debugging purposes. For the frame retransmit function, the LRFP indicates the point to begin retransmission of the data frame. For the frame discard function, the LWFP divides the valid data region of the FIFO (the area in-between the read and write pointers) into framed and unframed data. Data between the LRFP and write pointer is an incomplete frame, while data between the read pointer and the LWFP is received as whole frames. When the FRAME bit is not set, this pointer has no meaning.

25.3.18 Endpoint n FIFO Alarm Registers

The Endpoint n FIFO Alarm registers define the high/low level alarm setting. The number of Endpoint n FIFO Alarms in the MC9328MXL depends on the number of endpoints configured.

		Addr
USB_EP0_FALRM	Endpoint 0 FIFO Alarm Register	**0x00212050**
USB_EP1_FALRM	Endpoint 1 FIFO Alarm Register	**0x00212080**
USB_EP2_FALRM	Endpoint 2 FIFO Alarm Register	**0x002120B0**
USB_EP3_FALRM	Endpoint 3 FIFO Alarm Register	**0x002120E0**
USB_EP4_FALRM	Endpoint 4 FIFO Alarm Register	**0x00212110**
USB_EP5_FALRM	Endpoint 5 FIFO Alarm Register	**0x00212140**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											ALRM					
TYPE	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 25-23. Endpoint n FIFO Alarm Register Description

Name	Description	Settings
Reserved Bits 31–6	Reserved—These bits are reserved and should read 0.	
ALRM Bits 5–0	**Alarm Information**—Provides the assertion point for the “high level” and “low level” service requests. See the ALARM field of the Endpoint n FIFO Status for more information. A low level alarm reports lack of data while a high level alarm reports lack of space. The integrator must decide which alarm is necessary for each application. When the amount of data or space in the FIFO is above the indicated amount, the alarm sets in non-frame mode. In frame mode, the alarm sets when the amount of data or space in the FIFO is above the amount indicated by the ALRM setting *or* when there are end-of-frame bytes in the FIFO (see frame mode operation). A “high level” service request is asserted when there are less than ALRM bytes free in the FIFO. A “low level” service request asserts when there are less than ALRM bytes of data in the FIFO.	0x00 = 1 data byte left or 1 free byte available 0x01 = 2 data bytes left or 2 free bytes available ... 0x3F = 64 data bytes left or 64 free bytes available

25.3.19 Endpoint n FIFO Read Pointer Registers

The Endpoint n FIFO Read Pointer registers hold the location of the next FIFO location to read.

The number of Endpoint n FIFO Read Pointers in the MC9328MXL depends on the number of endpoints configured.

		Addr
USB_EP0_FRDP	Endpoint 0 FIFO Read Pointer Register	**0x00212054**
USB_EP1_FRDP	Endpoint 1 FIFO Read Pointer Register	**0x00212084**
USB_EP2_FRDP	Endpoint 2 FIFO Read Pointer Register	**0x002120B4**
USB_EP3_FRDP	Endpoint 3 FIFO Read Pointer Register	**0x002120E4**
USB_EP4_FRDP	Endpoint 4 FIFO Read Pointer Register	**0x00212114**
USB_EP5_FRDP	Endpoint 5 FIFO Read Pointer Register	**0x00212144**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											RP					
TYPE	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 25-24. Endpoint n FIFO Read Pointer Register Description

Name	Description
Reserved Bits 31–6	Reserved—These bits are reserved and should read 0.
RP Bits 5–0	**Read Pointer**—Points to the next FIFO location to read. The physical address of this FIFO location is actually the sum of the read pointer and the FIFO base, provided through a port to the FIFO controller. This base address can vary, however if chosen properly, the FIFO RAM address can be concatenated with the read pointer instead of requiring hardware for addition. The read pointer can be both read and written. This ability facilitates the debugging of the FIFO controller and peripheral drivers. The current maximum size of the write pointer is twelve bits, however it can be reduced through parameterization.

25.3.20 Endpoint n FIFO Write Pointer Registers

The Endpoint n FIFO Write Pointer registers hold the location of the next FIFO location to write.

The number of Endpoint n FIFO Write Pointers in the MC9328MXL depends on the number of endpoints configured.

		Addr
USB_EP0_FWRP	Endpoint 0 FIFO Write Pointer Register	**0x00212058**
USB_EP1_FWRP	Endpoint 1 FIFO Write Pointer Register	**0x00212088**
USB_EP2_FWRP	Endpoint 2 FIFO Write Pointer Register	**0x002120B8**
USB_EP3_FWRP	Endpoint 3 FIFO Write Pointer Register	**0x002120E8**
USB_EP4_FWRP	Endpoint 4 FIFO Write Pointer Register	**0x00212118**
USB_EP5_FWRP	Endpoint 5 FIFO Write Pointer Register	**0x00212148**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											WP					
TYPE	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 25-25. Endpoint n FIFO Write Pointer Register Description

Name	Description
Reserved Bits 31–6	Reserved—These bits are reserved and should read 0.
WP Bits 5–0	**Write Pointer**—Points to the next FIFO location to write. The physical address of this FIFO location is actually the sum of the read pointer and the FIFO base, provided through a port to the FIFO controller. This base address can vary, however if chosen properly, the FIFO RAM address can be concatenated with the read pointer instead of requiring hardware for addition. The read pointer can be both read and written. This ability facilitates the debugging of the FIFO controller and peripheral drivers.

25.4 Programmer's Reference

The programmer's reference guide gives details on how to program the USB module. This section covers device initialization, processing vendor requests, normal datapath operations, interrupt services, and reset operation.

25.5 Device Initialization

During device initialization, user software downloads critical configuration information to the UDC module and prepares the USB module datapath for processing. This process is performed at two different times: at reset (hard reset or software reset via RST bit in the USB_DDAR register) and when the device is first connected to the USB.

At power-up, the USB module contains no configuration information—such as, how many endpoints are available or how to locate the descriptors. Device initialization consists of downloading this information to the appropriate memories and to configure the datapath to match the intended application. The following steps are involved in the initialization process:

1. Perform a hard reset or a software reset (by setting the RST bit in USB Enable-USB_ENAB).
2. Wait for the device to be properly reset (after writing to RST bit, wait for it to clear) before accessing the device registers.
3. Wait for the CFG bit in the USB_DDAR register to assert before attempting to communicate with the UDC. Setting the RST bit forces the ENAB bit to set automatically.
4. Download configuration data (ENDPTBUFs, see Table 25-26 on page 25-36) to the device's USB Descriptor RAM/Endpoint Buffer Data-USB_DDAT).
5. Program the USB Interrupt Mask (USB_MASK) to enable interrupts not associated with a particular endpoint.
6. Program each Endpoint n Status/Control (USB_EPn_STAT) and Endpoint n Interrupt Mask (USB_EPn_MASK) to support the intended data transfer modes.
7. Program endpoint type, direction, and maximum packet size in the USB_EPn_STAT register for each endpoint.
8. Program the FIFO control registers. For each enabled endpoint, program frame mode, granularity, alarm level, and so on (USB_EPn_FCTRL and USB_EPn_FALRM). Normally, all endpoints should be programmed with FRAME mode enabled. To ensure proper operation of the DMA request lines for frame mode endpoints, program the alarm level to a value equal to or a multiple of the packet size of the endpoint.
9. Enable the USB for processing (set the USB_ENA bit in the USB Control, USB_CTRL).

NOTE:

Initialization of the USB module is a time critical process. The USB host waits about 100ms after power-on or a connection event to begin enumerating devices on the bus. This device must have all of the configuration information available when the host requests it.

After the device is enumerated, the USB host selects a specific configuration and set of interfaces on the device. Software on the device must beware of USB configuration changes to maintain proper communication with the USB host. The software retains sole responsibility for always knowing the current configuration and alternate interface. The CFG_CHG interrupt in the USB_INTR register reports changes in device configuration and alternate interface settings to the software. The software is required to respond to the CFG_CHG interrupt. To prevent the state of the device from becoming out of sync with respect to the host, the device halts further traffic on the USB while this interrupt is pending.

25.5.1 Configuration Download

The configuration download process initializes 6 endpoint buffers (ENDPTBUF-see Table 25-26 on page 25-36) within the UDC module to define its personality on the USB. The first ENDPTBUF is reserved for the default pipe (Control Endpoint) and must contain the value 0x0000080000. Endpoint buffers 1–5 can have a maximum of two configurations. Because endpoint buffers 1–5 can have up to two configurations, both configurations are downloaded to the device. A total of 55 bytes will transfer because endpoint buffer 0 is restricted to one configuration.

The endpoint buffers are 40-bit data strings per physical endpoint (pipe) that are loaded directly into the UDC module. They associate “logical” endpoint numbers in the USB software stack with hardware within the UDC. Specifically, they attach each endpoint to a USB configuration, interface, and alternate setting, and they specify transfer type, packet size, data direction, and hardware FIFO numbers.

Table 25-26. ENDPTBUF—UDC Endpoint Buffers Format

Bit/Field	Type	Description
[39:36]	EPNUM	Logical Endpoint Number
[35:34]	CONFIG	Configuration Number (maximum of 2 configurations: 1 and 2)
[33:32]	INTERFACE	Interface Number (maximum of 4 interfaces)
[31:29]	ALTSETTING	Alternate Setting Number (maximum of 8 alternate settings)
[28:27]	TYPE	Type of Endpoint: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt
[26]	DIR	Direction of the Endpoint: 0 = OUT endpoint 1 = IN endpoint
[25:16]	MAXPKTSIZE	Maximum packet size for the endpoint 0x08 = 8 bytes 0x10 = 16 bytes 0x20 = 32 bytes 0x40 = 64 bytes
[15:14]	TRXTYP	These bits must be set to 00 for endpoint 0, and 11 for all other endpoints.
[13:3]	Reserved	Reserved
[2:0]	FIFONUM	This field maps the endpoint to one of the <<BLOCK NAME>> module’s hardware FIFOs. Multiple UDC endpoints can map to a single hardware FIFO. It is up to the software to monitor and control any data hazards related to this type of operation in this way: The hardware FIFOs that are available are: FIFO0 (32 bytes) FIFO1 (64 bytes) FIFO2 (64 bytes) FIFO3 (32 bytes) FIFO4 (32 bytes) FIFO5 (32 bytes)

To download the configuration data, perform the following steps:

1. Verify that the CFG bit (in the USB_DADR register) is set. This ensures that the UDC is fully reset and is ready to take data.
2. Write the endpoint buffers (EnfPtBufs) to the USB_DDAT register. (the first byte written is EPn[39:32] and the last byte written is EPn[7:0]). After writing each byte, and before performing any other operation on the peripheral, verify that the BSY bit (in the USB_DADR register) is clear.
3. After writing all endpoint buffer configuration bytes, check the CFG bit of the USB_DADR to verify that the configuration download is complete. CFG changes from 1 to 0 after the last byte is loaded into the UDC.

25.5.1.1 USB Endpoint to FIFO Mapping

The USB protocol recognizes a maximum of 31 endpoints on a USB device. The endpoint numbers available on a specific USB device are based on the functionality present in the device, and on the configuration and alternate interfaces currently selected. Regardless of the logical endpoint number programmed in the interface descriptors, some hardware must be associated with each endpoint.

The USB module supports a maximum of 6 endpoints, including endpoint 0. Each hardware endpoint consists of a single FIFO that is independently programmable for direction, transfer type, frame mode, and low/high alarms. To fit these hardware endpoints to the USB's logical endpoints, mapping is established in the UDC module's endpoint buffer (EndPtBufs). Endpoint FIFO 0 is reserved for the default pipe (Control Endpoint).

Endpoint type, direction and packet size are defined in the USB_EPn_STAT register for each endpoint. FIFO characteristics are programmed in the USB_EPn_FCTRL and USB_EPn_FALRM registers. At power-up, the USB endpoints are mapped to specific hardware FIFOs when the UDC endpoint buffers are downloaded to the device from the CPU. The endpoint buffer makes 16 bits available to identify each hardware endpoint. These bits are the UDC_BufAdrPtr field [15:0] in the USB_SPEC register (see Table 25-26). Depending on the transaction selected by the UDC, this bus includes the indicator for a setup packet, or endpoint specific encoding. The UDC_BufAdrPtr [15:0] fields defined for the USB module are shown in Table 25-26 -EndPtBuf[15:0].

25.5.1.2 USB Interrupt

If the application uses the interrupt registers, the specific interrupts must be enabled. During reset, all interrupts revert to the masked state. USB global interrupts (interrupts that affect the whole module) are programmed separately from those affecting a single endpoint.

25.5.1.3 Endpoint Registers

The characteristics of the FIFO and a number of interrupt sources can be programmed for each endpoint. The integrator must program the following registers:

- USB Endpoint Interrupt Mask (USB_EPn_MASK)
  Separate interrupt registers are provided for each hardware FIFO. Enable the interrupts pertaining to the application by writing 0 to the mask bit for that interrupt.
- Endpoint FIFO Controller Configuration (USB_EPn_FCTRL).
  Each FIFOs programming is based on the type of data transmission used by the endpoint. Normally, all endpoints are programmed with FRAME mode.
- FIFO Alarm Register (USB_EPn_FALRM).
  For bulk traffic (FRAME = 1), the alarm level is normally programmed to a multiple of the USB packet size (for 8 byte packets and a 16 byte FIFO, the alarm is programmed to 8 bytes) to allow the DMA request lines

to request full packets. For single buffered endpoints (packet size = 8, FIFO depth = 8 bytes), the alarm is normally programmed to 0. For isochronous traffic, the alarm is programmed to allow streaming operation to occur on the isochronous endpoint.

25.5.1.4 Enable the Device

The last step in initializing the USB module is to enable it for processing in the USB_CTRL register. Most applications set the USB_SPD bit, the AFE_ENA bit and the USB_ENA bit in the USB_CTRL register.

NOTE:

The USB_SPD bit must be set at 1 (high speed mode, default). The USB device does not support low speed.

25.6 Exception Handling

Exception handling occurs in two situations:

- When corrupted frames must be discarded
- When an error occurs on the USB

The hardware automatically discards corrupted frames, so no software intervention is required.

If the device cannot respond to the host in the time allotted, the hardware automatically handles retries. No software intervention is required.

There are four error situations that must be dealt with by the software:

- Unable to Complete Device Request
- Aborted Device Request
- Unable to Fill or Empty FIFO Due to Temporary Problem
- Catastrophic Error.

Explanations for each of the errors are described in the following sections.

25.6.1 Unable to Complete Device Request

In the event that the software receives a device request it cannot interpret, it asserts the CMD_ERROR and CMD_OVER bits (in the USB_CTRL register) to the UDC. This results in a STALL to the endpoint in question and requires intervention from the USB host to clear. When the CMD_OVER bit clears, it means that the USB host has cleared the stall condition.

25.6.2 Aborted Device Request

When the host sends a setup packet to the device, the ACK handshake from the device can become corrupted and lost on its way to the host. If this happens, the host retries the setup packet, and the device can wind up with two or more setup packets in its FIFO. There are two ways to detect this condition:

- The presence of a MDEVREQ interrupt (in the USB_EPn_INTR register)
- The SIP bit in the Endpoint n Status/Control (USB_EPn_STAT) is active

In either case, the presence of more than one setup packet invalidates the first one in the FIFO. When MDEVREQ is active, software must discard the first one, and process the second one. When SIP is active, software must discard the first one, clear the device request, and wait for it to reassert.

25.6.3 Unable to Fill or Empty FIFO Due to Temporary Problem

When the USB module is unable to fill or empty a FIFO due to a temporary problem (such as the OS did not service the FIFO in time and it overflowed), the software stalls the endpoint through the FORCE_STALL bit in the USB_EPn_STAT register. This aborts the transfer in progress and forces intervention from the USB host to clear the stall condition. The FORCE_STALL register bit automatically clears after the stall takes effect. The application software on the host must deal with the stall condition and notify the device on how to proceed.

25.6.4 Catastrophic Error

In the case of a catastrophic error, the software executes a hard reset, re-initializes the USB module, and waits for the USB host to re-enumerate the bus.

25.7 Data Transfer Operations

Four types of data transfer modes exist for the USB module: control transfers, bulk transfers, isochronous transfers and interrupt transfer. From the perspective of the USB module, the interrupt transfer type is identical to the bulk data transfer mode, and no additional hardware is supplied to support it. This section covers the transfer modes and how they work.

Data moves across the USB in packets. Groups of packets are combined to form data transfers. The same packet transfer mechanism applies to bulk, interrupt, and control transfers. Isochronous data is also moved in the form of packets, however, because isochronous pipes are given a fixed portion of the USB bandwidth at all times, there is no end-of-transfer (EOT).

25.7.1 USB Packets

Packets range in size from 0 to 1023 bytes, depending on the transfer mode. For bulk, control, and interrupt traffic, packet sizes are limited to 8-, 16-, 32-, or 64-bytes. Isochronous packets can range from 0 bytes to 1023 bytes. The packet size is programmed within the UDC module on an endpoint by endpoint basis. For DMA access, the maximum packet size for isochronous endpoints is restricted by the endpoint's FIFO size.

The terms *packet* and *frame* are used interchangeably within this document. While USB traffic occurs in units called packets, the FIFO mechanism uses the term frames for the same blocks of data. The only difference between frames and packets from the user's standpoint is that packets can be as little as zero bytes in length, while a frame must be at least one byte in length.

25.7.1.1 Short Packets

Each endpoint has a maximum packet size associated with it. The packet size is set with the MAX field in the Endpoint n Status/Control. In most cases, packets transferred across an endpoint are sent at the maximum size. Because the USB does not indicate a data transfer size, or include an end-of-transfer token, short packets are used

to mark the end of data. Software writes short packets into the FIFO to indicate the end of data. Incoming short packets are identified by examining the length of a received packet or by looking at the end-of-transfer and end-of-frame interrupts.

25.7.1.2 Sending Packets

To send a packet of data to the USB host using programmed I/O, use these steps:

1. For an N byte packet, write the first N-1 bytes to the Endpoint n FIFO Data (USB_EPn_FDAT). Data can be written as bytes, words, or longwords.
2. For the Nth byte, set the WFR bit in the USB_EPn_FCTRL register before writing the last data byte to the USB_EPn_FDAT register. Data is written in big-endian format. The most significant byte (byte 3) is held in bits 7 through 0.

When using the DMA controller for a memory I/O transfer to send packets, the sequence is as follows:

1. Program the DMA controller to write data to the endpoint FIFO and enable the DMA channel for the USB endpoint DMA.
2. When the FIFO byte count has dropped below the alarm level, a DMA request is generated. The DMA writes data to the FIFO until the DMA request deasserts.
3. For an N byte packet, the first N-1 bytes are written to the FIFO data register (USB_EPn_FDAT) as bytes, words, or long words.
4. On the Nth byte, to signal the end of a frame, the DMA controller signals to the USB that it is writing the final byte of the USB_EPn_FDAT register. The last byte in the transfer gets the end-of-frame tag (bits 31–24 for byte, bits 23–16 for word, and bits 7–0 for longword).

In a double-buffered system, the FIFO depth is twice the size of the USB packet size. Program the FIFO alarm level to be the same as a single packet. This causes the DMA request to assert whenever there is the equivalent to one packet of data or less in the FIFO. The system can write data until the DMA request deasserts, as long as the last byte of each USB packet is tagged as the end-of-frame.

25.7.1.3 Receiving Packets

Perform the following steps to receive a packet of data from the USB host using programmed I/O.

1. Monitor the EOF interrupt for the endpoint.
2. On receiving the EOF interrupt, prepare to read a complete packet of data. Clear the EOF interrupt so that software receives notification of the next frame.
3. Read the USB_EPn_FDAT register for the next piece of data.
4. Read the USB_EPn_FSTAT register to get the end-of-frame status bits (see note below). When the end-of-frame bit is set for the current transfer, stop reading data.
5. Go to step 3.

NOTE:

When reading end-of-frame indicators from USB_EPn_FSTAT, the USB_EPn_FSTAT (FRAME [3:0]) field contains valid frame byte lanes used on the bus (31:24, 23:16, 15:8, 7:0). Currently, more than one bit can be set when there are multiple end-of-frame bytes on word or longword transfers. Extra software might be required to determine the first valid end-of-frame maker. The value of this

field is computed directly from the frame boundary bits stored in the RAM. The user must ensure that the RAM data is valid when accessing these bits. For example, when there are only two bytes of data marked EOF in the RAM, and the user does a longword access, bits 1:0 of this field are undefined.

25.7.1.4 Programming the FIFO Controller

The FIFO controller has two modes of operation, Frame and Non-Frame. Only Frame mode is normally used for the USB application.

In Frame mode, the FIFO controller can handle automatic hardware retry of bad packets. During device initialization, the user configures the FIFOs through the USB_EPn_FCTRL register for FRAME mode. Data flow is controlled with the end-of-frame and end-of-transfer interrupts, or with the DMA request lines. For isochronous endpoints, no data retries are performed.

25.7.2 USB Transfers

Data transfers on the USB are composed of one or more packets. Instead of maintaining a transfer count, the USB host and device send groups of packets to each other in units called transfers. In a transfer, all packets are the same size, except the last one. The last packet in a transfer is a short packet, as small as 0 bytes (when the last data byte ends on a packet boundary).

This section describes how data transfers work from both the device to the host, and from the host to the device.

25.7.2.1 Data Transfers to the Host

The user starts by determining the number of packets in the data block, based on the maximum packet size of the target endpoint.

When the number of packets is an integer, the transfer ends on a packet boundary. A zero length packet is required to terminate the transfer. When the number of packets is not an integer, the last packet of the transfer is a short packet and no zero length packet is required.

For each packet in the transfer, write the data to the USB_EPn_FDAT register. The last byte in each packet must be tagged with the end-of-frame marker through the USB_EPn_FCTRL register (WFR bit) or DMA service request lines. Monitor the FIFO_LOW interrupt, EOF interrupt, or DMA service requests to determine when the FIFO can accept another packet.

When a zero length packet is required to terminate the transfer after the last byte is written to the FIFO, set the ZLPS bit in the USB_EPn_STAT register for the endpoint.

Wait for the EOT interrupt to determine when the transfer is complete.

NOTE:

For DMA operation, a zero length frame is not defined so it is necessary to have the user software monitor the EOT interrupts and use them as a basis for delineating individual transfers. USB traffic flow is halted until the EOT interrupt is serviced to ensure that data from different data transfers does not get mixed-up in the FIFOs.

25.7.2.2 Data Transfers to the Device

The length of data transfer from the host is generally not known in advance. The device receives a continuous stream of packets and uses the EOT interrupt to determine when the transfer ended.

Software on the device monitors the EOF interrupt and/or the DMA transfer requests to manage packet traffic. Each time a packet is received, the device must pull the data from the FIFO. When an end-of-frame is transferred from the UDC module into the data FIFO, the EOF interrupt asserts. At the end of a complete transfer, the EOT interrupt asserts. For a bulk endpoint, until the CPU has serviced the EOT interrupt, the device Negative Acknowledges (NAKs) any further requests to that endpoint from the host. This guarantees that data from two different transfers never get intermixed within the FIFO.

25.7.3 Control Transfers

The USB host sends commands to the device through control transfers. Control transfers can be addressed to any control endpoint. Control transfers consist of three distinct phases: beginning with a setup phase, followed by an optional data phase, and ending with a status phase. Command processing occurs in the following steps:

1. Receive the SETUP packet on a control endpoint. DEVREQ and EOF interrupts assert for that endpoint.
2. Read 8 bytes of the setup packet from the appropriate FIFO data register and decode the command.
3. Clear EOF and DEVREQ interrupts.
4. Set up and perform the data transfer when a data transfer is implied by the command. Do not send more bytes to the USB host than were requested in the wLength field of the SETUP packet. Hardware does not check for incorrect data phase length. The EOT interrupt asserts on completion of the data phase.
5. Assert the CMD_OVER and CMD_ERROR bits (in the USB_CTRL register) to indicate processing or error status. The UDC module generates appropriate handshakes on the USB to implement the status phase. CMD_OVER automatically clears at the end of the status phase.
6. Wait for CMD_OVER to clear, indicating that the device request has completed.

The USB module assumes that the UDC module handles most of the standard requests without software intervention. User software does not need to be concerned with handling any of the so-called “Chapter 9" requests listed in the USB specification, except for SYNCH_FRAME, GET_DESCRIPTOR and SET_DESCRIPTOR. The requests are passed through endpoint 0 as a device request and must be processed by the device driver software.

25.7.4 Bulk Traffic

Bulk traffic guarantees the error-free delivery of data in the order that it was sent, however the rate of transfer is not guaranteed. Bandwidth is allocated to bulk, interrupt, and control packets based on the bandwidth usage policy of the USB host.

25.7.4.1 Bulk OUT

For OUT transfers (from host to device), internal logic marks the start of packet location in the FIFO. When a transfer does not complete without errors, the logic forces the FIFO to return to the start of the current packet and try again. No software intervention is required to handle packet retries.

User software reads packets from the FIFOs as they appear and stops reading when an EOT interrupt is received. To enable further data transfers, software services and clears the pending interrupts (EOF or EOT) and waits for the next transfer to begin. For Bulk Out endpoint, until the CPU has serviced the EOT interrupt, the device NAKs any further requests to that endpoint from the host. This guarantees that data from two different transfers never get intermixed within the FIFO.

25.7.4.2 Bulk IN

For IN transfers (from device to host), software tags the last byte in a packet to mark the end-of-frame. When a transfer does not complete without errors, the hardware automatically forces the FIFO to return to the start of the current packet and re-send the data. User software is expected to write data to the FIFO data register in units of the associated endpoint's maximum packet size. The end-of-frame can be indicated through the WFR bit in the Endpoint n FIFO Control (USB_EPn_FCTRL) or via the end-of-frame tag signal from the DMA controller.

In the USB protocol, the last packet in a transfer is allowed to be short (smaller than endpoint's maximum packet size) or even zero length. To indicate a zero length packet, the software sets the ZLPS bit in the associated Endpoint n Status/Control. The ZLPS bit automatically clears after the zero length packet is successfully sent to the host.

The EOT interrupt asserts to indicate that the last packet of the IN transfer has completed. Software clears any pending interrupts (EOT and EOF) to begin the next data transfer.

25.7.5 Interrupt Traffic

Interrupt endpoints are a special case of bulk traffic. Interrupt endpoints are serviced on a periodic basis by the USB host. Interrupt endpoints are guaranteed to transfer one packet per polling interval. Therefore, an endpoint with an 8 byte packet size and serviced every 2 ms would move 16 Kb/sec across the USB.

The only difference between interrupt transfers and bulk transfers from the device standpoint is that every time an interrupt packet is transferred, regardless of size, the EOT interrupt asserts. The device driver software must service this interrupt packet before the next interrupt servicing interval to prevent the device from NAKing the poll.

Device driver software must ensure that the interrupt endpoint polling interval is longer than the device's interrupt service latency.

25.7.6 Isochronous Operations

Isochronous operations are a special case of USB traffic. Instead of guaranteeing delivery with unbounded latency, isochronous traffic flows over the bus at a guaranteed rate with no error checking.

25.7.6.1 Isochronous Transfers in a Nutshell

The USB host guarantees an endpoint exactly one isochronous packet per frame. Isochronous packets can range from 0 bytes to 1023 bytes. See the USB specification for more information on isochronous transfer.

Because isochronous packets can be as large as 1023 bytes, it is not practical to implement large FIFOs for each endpoint. Instead, the software drivers are responsible for keeping the FIFOs serviced. When an IN or OUT request is received on an isochronous endpoint, the software drivers must ensure that the correct amount of data can be transferred without emptying the FIFO. When the FIFO empties during an isochronous packet transfer, the host terminates the packet immediately and the device loses its time slot until the next USB frame.

For DMA access, the maximum packet size for isochronous endpoints is restricted by the endpoint's FIFO size. For programmed I/O, isochronous data packets can take any size from 0 to 1023 bytes.

To allow the driver software to maintain synchronization with the USB host, the <<BLOCK NAME>> maintains a register that holds the current USB frame number. An interrupt is asserted each time the frame number changes or when a specific USB frame number is received. The interrupts are maskable.

The EOT interrupt asserts for isochronous packet transfers when the UDC module reports that the packet data is error free. This can be used along with the EOF interrupt to determine when a transfer error of some sort occurs on an isochronous endpoint.

25.7.6.2 The SYNCH_FRAME Standard Request

The SYNCH_FRAME standard request allows synchronization between the USB host and device during isochronous operations. The command is passed through endpoint 0 as a device request and must be processed by the device driver software.

25.8 Interrupt Services

The USB module generates a number of interrupts to indicate situations requiring attention from the host. The types of interrupts are broken into two categories: USB general interrupts and Endpoint specific interrupts. These interrupts are discussed in this section, as well as missed interrupts and their behaviors.

25.8.1 USB General Interrupts

The USB general interrupts indicate such things as global configuration and status changes pertaining to the USB. All of these interrupts are maskable. When an event causes an interrupt condition to occur, and the corresponding bit in the interrupt mask register is zero, an interrupt signal asserts on the module's interface. Writing 1 to the associated bit in the interrupt register clears the interrupt. After a hard reset, all interrupts are masked by default. This section describes each of the USB General Interrupts.

25.8.1.1 MSOF—Missed Start-of-Frame

The MSOF interrupt is used to inform software that it did not service the SOF interrupt before another SOF interrupt was received.

25.8.1.2 SOF—Start-of-Frame

The SOF interrupt means that a start-of-frame token was received by the device. The current USB frame number can be read from the USB_FRAME register. The start-of-frame interrupt is usually used by isochronous devices to provide a stable timebase.

25.8.1.3 RESET_STOP—End of USB Reset Signaling

The RESET_STOP interrupt means that reset signaling on the USB has stopped.

25.8.1.4 RESET_START—Start of USB Reset Signaling

The RESET_START interrupt indicates that reset signaling on the USB has begun. When reset signaling is active on the USB, the device does not expect to receive any transactions on the bus. This interrupt results in a reset of the UDC into the powered state, however it does not cause any specific actions in the <<BLOCK NAME>> front-end logic. User software must clear any pending interrupts and ensure that the module is configured properly after the reset. This interrupt indicates the start of reset signaling. Status of the USB at any given moment can be verified by examining the USB_STAT register.

Presence of USB reset signaling invalidates any transaction in progress. On detection of RESET_START, user software reads any remaining valid data from the receive FIFOs and flushes all others. When reset signaling is detected, user software must clear any pending interrupts and ensure that the module is configured properly after the reset.

25.8.1.5 WAKEUP—Resume (Wakeup) Signaling Detected

This interrupt asserts when resume signaling is detected on the USB. The device uses this interrupt to wake up from suspend mode and resume normal operations. This interrupt is independent of the clock to the USB module.

25.8.1.6 SUSP—USB Suspended

This interrupt asserts when the USB goes into suspend mode. Suspend mode occurs when the device fails to receive any traffic from the USB for a period of 6 ms. When suspend mode is detected, the device puts itself into a low power standby mode.

25.8.1.7 FRAME_MATCH—Match Detected in USB_FRAME Register

This interrupt asserts when the frame number programmed into the MATCH field of the USB_FRAME register is the same as the FRAME field of the USB_FRAME register. Isochronous pipes can use this interrupt for synchronization between the data source and sink.

25.8.1.8 CFG_CHG—Host Changed USB Device Configuration

This interrupt means that the USB host selected a different configuration or alternate interface. Software reads the USB_STAT register to determine the current configuration and interfaces and reconfigures itself accordingly.

25.8.2 Endpoint Interrupts

The endpoint interrupts indicate requests for service by specific USB endpoints. All bits are maskable. Each endpoint's interrupt output is connected to a separate hardware interrupt line. When an event occurs that causes an interrupt condition to occur, and the corresponding bit in the interrupt mask register is zero, an interrupt signal asserts on the module's interface. Writing 1 to the associated bit in the interrupt register clears the interrupt.

25.8.2.1 FIFO_FULL

This interrupt asserts when the FIFO is full.

25.8.2.2 FIFO_EMPTY

This interrupt asserts when the FIFO is empty.

25.8.2.3 FIFO_ERROR

This interrupt means some abnormal condition occurred in the FIFO. The cause of the error can be verified by reading the USB_EPn_FSTAT register associated with the FIFO that had the error.

25.8.2.4 FIFO_HIGH

Each FIFO has an alarm register. The FIFO_HIGH interrupt asserts when the number of free bytes in the FIFO is below the level specified by the alarm register.

25.8.2.5 FIFO_LOW

Each FIFO has an alarm register. The FIFO_LOW interrupt asserts when the byte count in the FIFO is below the level specified by the alarm register.

25.8.2.6 EOT—End of Transfer

This interrupt asserts after the last data byte of a USB transfer crosses from the <<BLOCK NAME>> into the UDC module or vice versa. The end of a USB transfer is indicated by either a zero byte packet or by a data packet shorter than the maximum packet size for the endpoint.

The EOT never asserts along with the DEVREQ interrupt for setup packets.The EOT interrupt asserts after every interrupt packet transfer, every complete bulk data transfer and data phase of control transfer. The EOT interrupt generally asserts along with an EOF interrupt, although an EOT interrupt can occur without an EOF interrupt when a transfer terminates on a USB packet boundary. The EOT interrupt asserts for isochronous packet transfers when the UDC module reports that the packet data is error free. This can be used along with the EOF interrupt to determine when a transfer error of some sort occurs on an isochronous endpoint.

25.8.2.7 DEVREQ—Device Request

The Device Request (Setup Packet) interrupt means that the most recently received packet was a setup or device request packet. Software on the USB device must decode and respond to the packet to complete a Vendor, Class, or Standard request.

25.8.2.8 MDEVREQ—Multiple Device Request

The Multiple Device Requests indicator asserts when two or more setup packets have been received before the DEVREQ interrupt was cleared. This interrupt is used to determine when the USB host has aborted a transfer in progress. In this case, the device receives a setup packet, followed by a new setup packet before it has completed processing of the original command.

25.8.2.9 EOF—End of Frame

This interrupt means that an end-of-frame marker was sent or received on the FIFO/UDC interface. This interrupt asserts when a DEVREQ is received for bulk, control, isochronous, and interrupt data. While packet retries are not supported for isochronous endpoints, the end-of-frame indicator is still valid and can be used along with the SOF interrupt to control data flow.

25.8.3 Interrupts, Missed Interrupts, and the USB

Improper operation of the device can result when interrupts are not serviced in a timely manner. For example, a CFG_CHG interrupt is received, the device does not service it, and another CFG_CHG interrupt is received. This could leave the device in an incorrect operating mode. The interrupts of concern in this manner are SOF, CFG_CHG, EOT, and DEVREQ. The missed-interrupt behaviors are discussed in the following subsections.

25.8.3.1 SOF

When the device misses a start-of-frame interrupt, the MSOF bit asserts in the USB_INTR register.

25.8.3.2 CFG_CHG

When the device receives a CFG_CHG interrupt, the module NAKs all traffic from the USB host until software clears the interrupt bit. This prevents the device configuration from getting out of sync with what the host has requested.

25.8.3.3 EOT

When an end-of-transfer is received on a BULK OUT endpoint, the device NAKs all traffic on that endpoint until software clears the interrupt bit. This prevents data from two different transfers from becoming mixed up in a FIFO.

25.8.3.4 DEVREQ

When a device request is received, the device NAKs all IN/OUT traffic on the affected endpoint until software clears the interrupt bit. This ensures that the device correctly identifies the setup packet in the FIFO and can clear the FIFO before the data phase is allowed to begin. When multiple setup packets are received, the MDEVREQ interrupt asserts.

25.9 Reset Operation

The USB module includes four reset modes: Hard Reset, Software Reset, UDC reset and USB Reset signaling.

The UDC reset allows software to force a hard reset of the UDC module only, leaving all register bits in the front-end logic intact. A UDC reset is normally used only as a debug option, however it can also be used in the event of a connect/disconnect bus event. A hard reset requires that MCU PLL and System PLL be locked.

25.9.1 Hard Reset

A hard reset is generated from the USB module's bus interface, and resets all storage elements in both the front-end logic and in the UDC module. A hard reset also issues a UDC reset. Both the MCU PLL and System PLL must be locked before issuing a Software Reset.

25.9.2 USB Software Reset

The USB device allows the reset of all the storage elements in both the front-end logic and in the UDC module through the RST bit in the USB_ENAB register. On initial power-up, the user issues a Software reset. This causes the module to be enabled and the internal logic to be reset. Both the MCU PLL and USB PLL must be locked before issuing a Software Reset.

25.9.3 UDC Reset

A UDC reset is accomplished by setting the UDC_RST bit in the USB Control (USB_CTRL.) The UDC must be reset any time a connect/disconnect occurs on the USB. Any time the device is plugged in or unplugged from the USB, software must initiate either a hard reset or a UDC reset to ensure that the module can properly communicate with the USB host. Reset signaling is discussed in chapter 7 of the USB Specification. UDC Reset can invalidate data remaining in the data FIFOs. Depending on the application, software might need to flush the data FIFOs before proceeding.

25.9.4 USB Reset Signaling

Reset signaling can occur on the USB for a number of reasons. When the device receives reset signaling, it means that the host is preparing to re-enumerate the bus. All transactions in progress must be invalidated, and the device software prepared to receive configuration changes.

Because the device can contain valid, but as-yet unread, data in the FIFOs when USB reset signaling occurs, the hardware does not flush the FIFOs. When device software receives reset signaling, it completes reading any unread data from the FIFOs and execute FIFO flush operations on all of the FIFOs. This guarantees that the datapath is empty and ready for new data transfer operations when reset signaling and re-enumeration are complete.

Chapter 26
I^2C Module

26.1 Overview

I^2C is a two-wire bidirectional serial bus that provides a simple and efficient method of data exchange while minimizing the interconnection between devices. This bus is most suitable for applications requiring occasional communication between many devices in close proximity. The flexible I^2C bus allows additional devices to be connected to the bus for expansion and system development.

The I^2C system is a true multiple-master bus. It features collision and arbitration detection to prevent data corruption when multiple devices attempt to control the bus simultaneously. This allows for complex applications with multiprocessor control and can support rapid testing and alignment of end products through external connections to an assembly-line computer.

26.2 Interface Features

The following are key features of the I^2C module:

- Compatible with the I^2C bus standard
- Supports 3.3V tolerant devices
- Multiple-master operation
- Software programmable clock frequencies (supports 64 different frequencies)
- Software selectable acknowledge bit
- Interrupt driven, byte-by-byte data transfer
- Arbitration lost interrupt with automatic switching from master to slave mode
- Calling address identification interrupt
- START and STOP signal generation and detection
- Repeated START signal generation
- Acknowledge bit generation and detection
- Bus-busy detection

Figure 26-1 on page 26-2 shows a block diagram of the I^2C module.

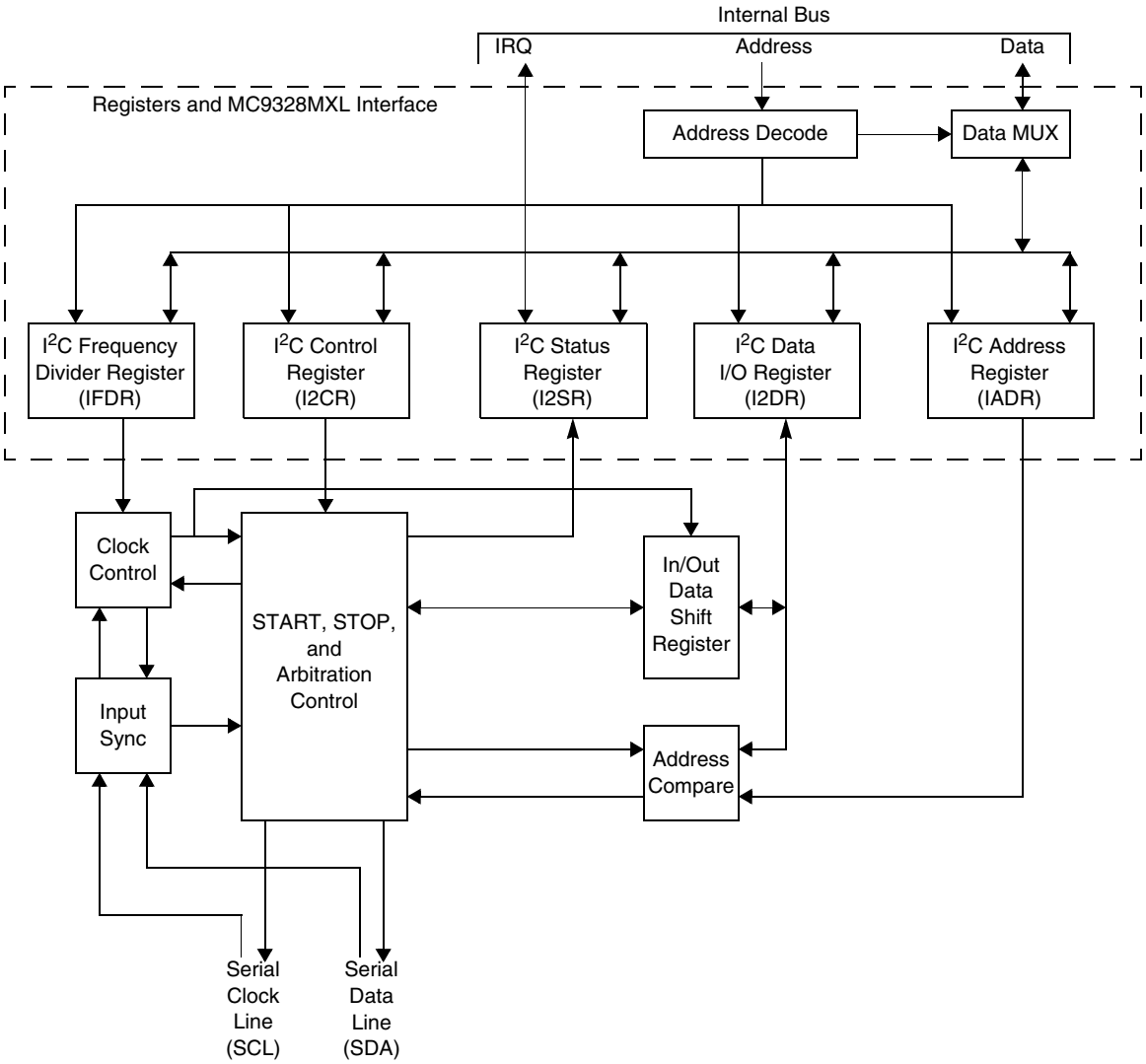


Figure 26-1. I²C Module Block Diagram

26.3 I²C System Configuration

The I²C module uses a serial data line (SDA) and a serial clock line (SCL) for data transfer. For I²C compliance, all devices connected to these two signals must have open drain or open collector outputs. (There is no such requirement for inputs.) The logic AND function is exercised on both lines with external pull-up resistors.

The default state of the I²C is slave receiver. When not programmed to be a master or responding to a slave transmit address, the I²C module always returns to the default state. Exceptions are described in Section 26.7.1, “Initialization Sequence.”

NOTE:

The I²C module is designed to be compatible with *The I²C Bus Specification,* Version 2.1 (Philips Semiconductor: 2000). For detailed information on system configuration, protocol, and restrictions, see the Philips I²C standard.

26.4 I²C Protocol

The I²C communication protocol consists of six components: START, Data Source/Recipient, Data Direction, Slave Acknowledge, Data, Data Acknowledge and STOP. These are shown in Figure 26-2 and described in the text following the figure.

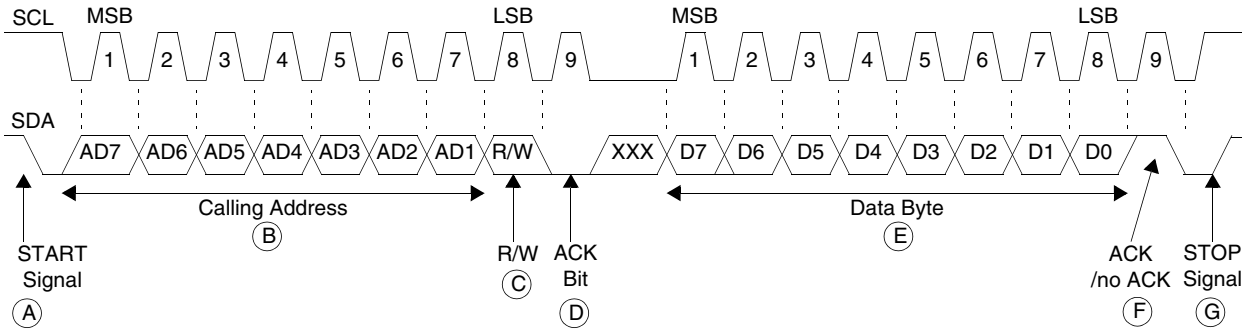


Figure 26-2. I²C Standard Communication Protocol

- **START (A)**—When the bus is free (both the SCL and the SDA lines are at logic high), a device can initiate communication by sending a START signal. This is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a data transfer and awakens all the slave devices on the bus.
- **Data Source/Recipient (B)**—The master sends out the 7-bit address of the slave that it intends to transfer data to or receive data from. Each slave must have a unique address. An I²C master cannot transmit to its own slave address—a device cannot be master and slave at the same time.
- **Data Direction (C)**—This single bit sets the data transfer direction.
- **Acknowledge (D)**—The addressed slave device responds to the master by returning an acknowledge bit. This is defined as the SDA line pulled low on the ninth clock after the START signal. This acknowledge is independent of the values of the Acknowledge Enable bit in the Control Register.
- **Data (E)**—The data transfer is done on a byte-by-byte basis in the direction specified by the Data Direction bit. Data can be changed only while the SCL is low and must be held stable while the SCL is high. The SCL is pulsed once for each data bit and the most significant bit (MSB) is sent first.
- **Acknowledge (F)**—The receiving device must acknowledge each byte by pulling the SDA low on the ninth clock, so a data byte transfer takes nine clock pulses. For multi-byte data transactions, an end-of-data condition is indicated when the receiver does not send an acknowledge.

  When the master transmits data to the slave and the slave does not acknowledge the master, this indicates an end-of-data condition to the master. The SDA line is left high and the master can generate a STOP signal to abort the data transfer or generate a START signal (repeated START, shown in Figure 26-3 on page 26-4) to restart the transfer.

  When the master receives data from the slave and the master does not acknowledge the slave, this indicates an end-of-data condition to the slave. The slave releases the SDA and the master generates a STOP or START signal.
- **STOP (G)**—A STOP signal is sent to free the bus after data is transmitted or when the master stops communication. A STOP signal is defined as a low-to-high transition of the SDA while the SCL is at logical high.

NOTE:

A master can generate a STOP even when the slave sent an acknowledge. The slave must release the bus.

- **Repeated START (H)**—Instead of sending a STOP signal, the master can repeat the START signal and the rest of the communication protocol. When a START signal is generated without a preceding STOP signal, a repeated START occurs. This is shown in Figure 26-3. The master generates a repeated START to communicate with another slave or with the same slave in a different mode (transmit/receive mode) without releasing the bus.

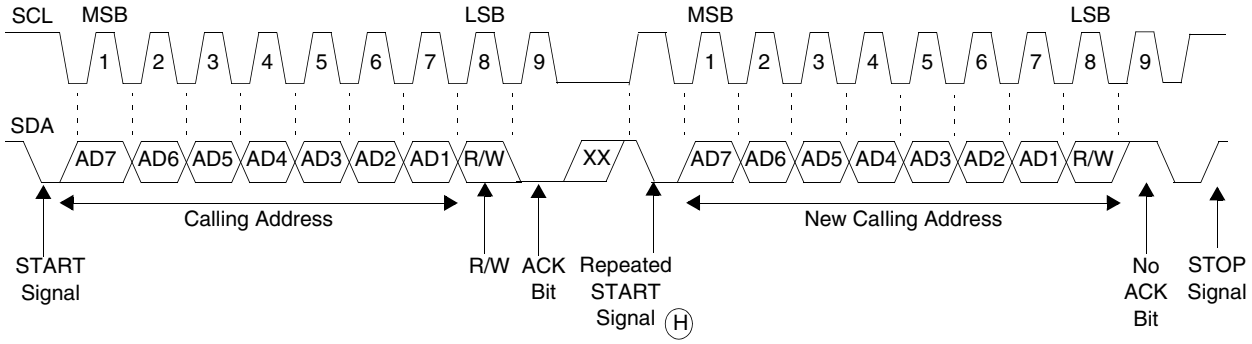


Figure 26-3. Repeated START

26.4.1 Clock Synchronization

When multiple devices simultaneously request the bus, the bus clock is determined by a synchronization procedure in which the clock lines from the individual devices are compared. The system clock line (SCL) is the result of an AND operation on all individual clock lines.

After the master drives the SCL low, the internal clocks begin counting their low periods. By performing an AND operation, the system clock stays low until ALL of the individual device clocks on the bus have transitioned to the high state. At this point, the individual clocks begin counting their high periods. The system clock remains high until ANY of the device clocks on the bus transition to the low state.

As a result, the devices with the longest low periods and shortest high periods control the SCL.

Devices with shorter low periods transition to high before the SCL transitions. This wait is shown in Figure 26-4.

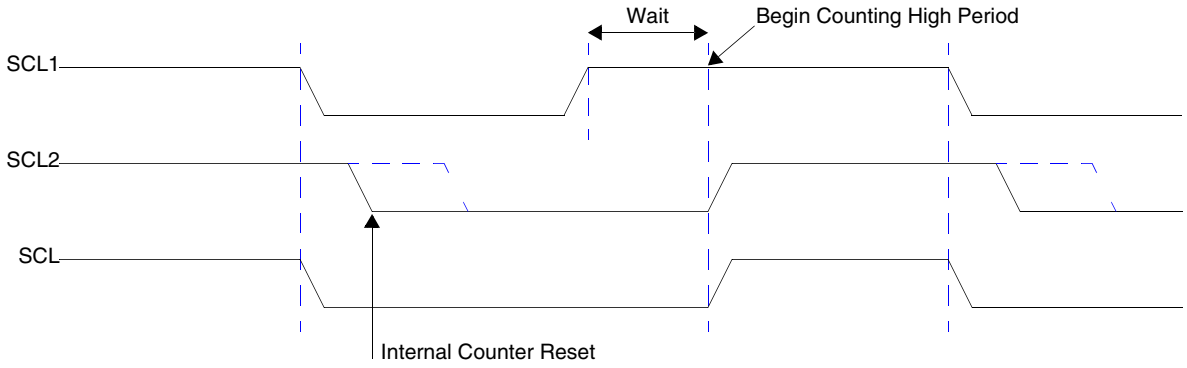


Figure 26-4. Synchronized Clock SCL

26.4.2 Arbitration Procedure

The relative priority of competing devices is determined by a data arbitration procedure. A device loses arbitration when it sends a logic high while another device sends a logic low. The device that loses arbitration immediately switches to slave-receive mode, stops driving SDA, and sets the Arbitration Lost (IAL) bit in its I2C Status Register (I2SR). In this case, the transition from master to slave mode does not generate a STOP condition.

26.4.3 Handshaking

Clock synchronization can function as a handshake in data transfers. When slave devices hold the SCL low after completing a one byte transfer (9 bits), the clock synchronization feature halts the bus clock and forces the master clock into a wait state until the slave releases the SCL.

26.4.4 Clock Stretching

Clock synchronization allows slaves to slow down the transfer bit rate. After the master drives the SCL low, the slave can hold the SCL low. When the slave SCL low period is longer than the master SCL low period, the SCL bus signal low period is stretched.

26.5 Pin Configuration for I2C

Two pins are available for the I2C module. These pins are multiplexed with other functions on the device and must be configured for SPI operation.

NOTE:

The user must ensure that the data direction bits in the GPIO are set to the correct direction for proper operation. See Section 29.5.1, “Data Direction Registers,” on page 29-8 for details.

Table 26-1. Pin Configuration

Pin	Setting	Configuration Procedure
I2C_SCL	**I2C CLOCK**—This pin is open-drain when not in GPIO. This is the primary function of GPIO Port A [16].	1. Clear bit 16 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 16 of Port A General Purpose Register (GPR_A)
I2C_SDA	**I2C Data**—This pin is open-drain when not in GPIO. This is the primary function of GPIO Port A [15].	1. Clear bit 15 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 15 of Port A General Purpose Register (GPR_A)

26.6 Programming Model

The I^2C module includes five 32-bit registers. Table 26-2 summarizes these registers and their addresses.

Table 26-2. I^2C Module Register Memory Map

Description	Name	Address
I^2C Address Register	IADR	0x00217000
I^2C Frequency Divider Register	IFDR	0x00217004
I^2C Control Register	I2CR	0x00217008
I^2C Status Register	I2SR	0x0021700C
I^2C Data I/O Register	I2DR	0x00217010

26.6.1 I²C Address Register

The I²C Address Register (IADR) holds the address to which the I²C responds when addressed as a slave.

NOTE:

As part of the I²C communications protocol, the master sends out the 7-bit address of the slave it intends to transfer data to or receive data from. When the MC9328MXL is the bus master, the address it sends out is the address of its intended slave device. It is NOT the address in this register. This value is only referenced when another device is bus master and it intends to communicate with the MC9328MXL.

IADR I²C Address Register **Addr 0x00217000**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									ADR							
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 26-3. I²C Address Register Description

Name	Description
Reserved Bits 31–8	Reserved—These bits are reserved and should read 0.
ADR Bits 7–1	**Slave Address**—Contains the specific slave address to be used by the I²C module. Slave mode is the default I²C mode for an address match on the bus.
Reserved Bit 0	Reserved—This bit is reserved and should read 0.

26.6.2 I²C Frequency Divider Register

The I²C Frequency Divider Register (IFDR) holds the prescaler value that configures the clock for bit-rate selection.

IFDR I²C Frequency Divider Register **Addr 0x00217004**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											IC					
TYPE	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 26-4. IFDR Register Description

Name	Description	Settings
Reserved Bits 31–6	Reserved—These bits are reserved and should read 0.	
IC Bits 5–0	**I²C Clock Rate Divider**—Prescales the clock for bit-rate selection. Due to potentially slow rise and fall times of SCL and SDA, bus signals are sampled at the prescaler frequency. The input clock from the PLL is HCLK. The serial bit clock for the I²C module is HCLK divided by the divider shown in Table 26-5. **Note:** The IC value can be changed at any point.	See Table 26-5 on page 26-9.

Table 26-5. HCLK Dividers

IC	Divider	IC	Divider	IC	Divider	IC	Divider
0x00	30	0x10	288	0x20	22	0x30	160
0x01	32	0x11	320	0x21	24	0x31	192
0x02	36	0x12	384	0x22	26	0x32	224
0x03	42	0x13	480	0x23	28	0x33	256
0x04	48	0x14	576	0x24	32	0x34	320
0x05	52	0x15	640	0x25	36	0x35	384
0x06	60	0x16	768	0x26	40	0x36	448
0x07	72	0x17	960	0x27	44	0x37	512
0x08	80	0x18	1152	0x28	48	0x38	640
0x09	88	0x19	1280	0x29	56	0x39	768
0x0A	104	0x1A	1536	0x2A	64	0x3A	896
0x0B	128	0x1B	1920	0x2B	72	0x3B	1024
0x0C	144	0x1C	2304	0x2C	80	0x3C	1280
0x0D	160	0x1D	2560	0x2D	96	0x3D	1536
0x0E	192	0x1E	3072	0x2E	112	0x3E	1792
0x0F	240	0x1F	3840	0x2F	128	0x3F	2048

26.6.3 I²C Control Register

The I²C Control Register (I2CR) enables the I²C module and the I²C interrupt. This register also contains bits that select whether the device operates as a slave or a master.

I2CR — I²C Control Register — **Addr 0x00217008**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									IEN	IIEN	MSTA	MTX	TXAK	RSTA		
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	$\overline{rw}$	$\overline{r}$	$\overline{r}$
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	$\overline{0}$	$\overline{0}$	$\overline{0}$
	0x0000															

Table 26-6. I²C Control Register Description

Name	Description	Settings
Reserved Bits 31–8	Reserved—These bits are reserved and should read 0.	
IEN Bit 7	**I²C Enable**—Controls the software reset of the entire I²C module. When the module is enabled in the middle of a byte transfer, slave mode ignores the current bus transfer and begins operating whenever a subsequent START condition is detected. Master mode is not aware that the bus is busy, so when a START cycle is initiated, the current bus cycle can become corrupted and cause either the current bus master or the I²C module to lose arbitration, after which bus operation returns to normal.	0 = Disable the I²C module (registers can still be accessed) 1 = Enable the I²C module (this bit must be set before any other I2CR bits have any effect)
IIEN Bit 6	**I²C Interrupt Enable**—Enables the I²C interrupts.	0 = Disable the I²C module interrupts (currently pending interrupt conditions are not cleared) 1 = Enable the I²C module interrupts (an I²C interrupt occurs when the I²C interrupt (IIF) bit in the I2SR Register is also set)
MSTA Bit 5	**Master/Slave Mode Select**—Selects master or slave mode operation. When the device loses arbitration while running in master mode, MSTA is cleared without generating a STOP signal.	0 = Slave mode (changing MSTA from 1 to 0 generates a STOP and selects slave mode) 1 = Master mode (changing MSTA from 0 to 1 signals a START on the bus and selects master mode)

Table 26-6. I²C Control Register Description (continued)

Name	Description	Settings
MTX Bit 4	**Transmit/Receive Mode Select**—Selects the direction of master and slave transfers. **Note:** When a slave is addressed during transmit mode, software sets MTX according to the Slave Read/Write (SRW) bit in the I2SR Register. In master mode, MTX is set according to the type of transfer required. Therefore, for address cycles, MTX is always set for master mode and cleared for slave mode.	0 = Receive 1 = Transmit
TXAK Bit 3	**Transmit Acknowledge Enable**—Specifies the value driven onto SDA during data acknowledge cycles for both master mode and slave mode receivers. **Note:** TXAK applies only when the I²C bus is a receiver.	0 = Sends an acknowledge signal to the bus at the ninth clock bit after receiving one byte of data 1 = No acknowledge signal response is sent (the data acknowledge bit in the protocol = 1)
RSTA Bit 2	**Repeated START**—Generates a repeated START condition. This bit always reads 0. Attempting a repeated START without bus mastership causes loss of arbitration.	0 = No repeated START 1 = Generates a repeated START
Reserved Bits 1–0	Reserved—These bits are reserved and should read 0.	

26.6.4 I²C Status Register

The I²C Status Register (I2SR) indicates transaction direction and status.

I2SR — I²C Status Register — **Addr 0x0021700C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									ICF	IAAS	IBB	IAL		SRW	IIF	RXAK
TYPE	r	r	r	r	r	r	r	r	r	r	r	rw	r	r	rw	r
RESET	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
	0x0081															

Table 26-7. I2SR Register Description

Name	Description	Settings
Reserved Bits 31–8	Reserved—These bits are reserved and should read 0.	
ICF Bit 7	**Data Transfer**—Indicates data transfer condition. ICF is cleared while one byte of data is transferred.	0 = Transfer in progress 1 = Transfer complete (set at the falling edge of the ninth clock of a byte transfer)
IAAS Bit 6	**I^2C Addressed As a Slave**—Indicates that the device slave address matches the address sent on the data line. When the I^2C Enable (IIEN) bit in the I2CR Register is set, an interrupt is generated to the ARM9 core when this match occurs. The ARM9 core must check the Slave Read/Write (SRW) bit and set the Transmit/Receive Mode Select (MTX) bit of the I2CR Register accordingly. Writing to the I2CR Register clears this bit.	0 = Not addressed 1 = Addressed as a slave (set when the MC9328MXL slave address in the IADR Register matches the calling address)
IBB Bit 5	**I^2C Bus Busy**—Indicates the status of the bus. When a STOP signal is detected, IBB is cleared, and when a START signal is detected, IBB is set.	0 = Bus is idle 1 = Bus is busy
IAL Bit 4	**Arbitration Lost**—Indicates that this device will not operate as bus master. IAL cleared by writing 0 to it, and is set by the hardware in the following circumstances: • SDA samples low when the master drives high during an address or data-transmit cycle • SDA samples low when the master drives high during the acknowledge bit of a data-receive cycle • A START is attempted when the bus is busy • A repeated START is requested in slave mode • A STOP condition is detected when the master did not request it	
Reserved Bit 3	Reserved—This bit is reserved and should read 0.	
SRW Bit 2	**Slave Read/Write**—Indicates the value of the R/W command bit of the I^2C protocol when the device is the addressed as a slave (IAAS bit is set). SRW is valid only when a complete transfer occurs, no other transfers have been initiated, and the I^2C module is a slave with an address match.	0 = Slave receiver, master writing to slave 1 = Slave transmitter, master reading from slave
IIF Bit 1	**I^2C interrupt**—Indicates an interrupt condition. Cleared by writing 0 in the interrupt routine. Set when one of the following occurs: • Completion of one byte transfer (set at the falling edge of the ninth clock) • Calling address matches MC9328MXL slave address • Arbitration is lost	0 = No I^2C interrupt pending 1 = An interrupt is pending, which causes a processor interrupt request (when IIEN = 1).
RXAK Bit 0	**Received Acknowledge**—Indicates whether an acknowledge signal (to the data) was received on SDA.	0 = An acknowledge signal was received after the completion of 8-bit data transmission on the bus 1 = No acknowledge signal was detected at the ninth clock

26.6.5 I²C Data I/O Register

The I²C Data I/O Register (I2DR) contains the data received or the data to be transmitted during I/O operations. In transmission mode, this value is sent out after the receiving device sends an acknowledge signal. In master-receive mode, the last byte received is held in this register. Reading this register initiates the transfer of the next byte of receive data. In slave mode, the same function is available after the device is addressed.

I2DR — I²C Data I/O Register — **Addr 0x00217010**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									D							
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 26-8. I2DR Register Description

Name	Description
Reserved Bits 31–8	Reserved—These bits are reserved and should read 0.
D Bits 7–0	**I²C Data**—Holds last data byte received or next data byte to be transferred.

26.7 I²C Programming Examples

This section describes programming sequences for I²C, including initialization, START signalling, post-transfer software response, STOP signalling, and repeated START generation. The flowchart in Figure 26-5 on page 26-16 illustrates an interrupt routine.

26.7.1 Initialization Sequence

The registers must be initialized before the interface can transfer serial data. The procedure to initialize the registers is as follows:

1. Set the Clock Rate Divider (IC field) in the I²C Frequency Divider Register (IFDR) for the appropriate SCL frequency.
2. Write the device slave address in the I²C Address Register (IADR).
3. Enable the I²C module by setting the I²C Enable bit (IEN) in the I²C Control Register (I2CR).
4. Modify the bits in the I²C Control Register (I2CR) to select master/slave mode, transmit/receive mode, and interrupt enable/disable.

NOTE:

Before enabling the I²C module, ensure that another communication is not in progress by verifying that the I²C Bus Busy bit (IBB) of the I²C Status Register (I2SR) is cleared. When it is not cleared, execute the following code sequence to force the slave device into an idle condition by issuing a STOP command before enabling the I²C module.

26.7.2 Generation of START

After the initialization procedure is complete and the serial bus is free, the MC9328MXL transmits serial data by selecting the master transmitter mode. The IBB bit in the I2SR Register indicates the bus condition. When the bus is free (IBB bit in the I2SR Register = 0), the START signal and the slave address can be sent. The address of the appropriate slave is written to the I²C Data I/O Register (I2DR), where the least significant bit (LSB) indicates the transfer direction.

The bus free time between a STOP signal and the next START signal is built into the hardware that generates the START. Depending on the relative frequencies of the system clock and the SCL period, it is sometimes necessary to wait after writing the calling address to the I2DR before proceeding with the following instructions.

26.7.3 Post-Transfer Software Response

When one byte is sent or received, the Data Transfer (ICF) and the I²C Interrupt (IIF) bits in the I²C Status Register (I2SR) are set. An interrupt occurs when interrupts are enabled by the IIEN bit of the I²C Control Register (I2CR).

First, the software must clear IIF in the interrupt routine. ICF is cleared either by reading the data from the I²C Data I/O Register (I2DR) in receive mode or by writing to the I2DR in transmit mode. Clearing ICF triggers the start of the next communication byte.

The software can service the I²C I/O in the main program by disabling the interrupts (clear IIEN) and polling the IIF bit.

When an interrupt occurs at the end of the address cycle, the master is still in transmit mode. When master receive mode is required, toggle the MTX bit in the I2CR Register.

When the device is functioning as the addressed slave (the IAAS bit in the I2SR is set), the SRW bit in the I2SR Register is read to determine the direction of the next transfer, and MTX is programmed accordingly. For slave-mode data cycles (IAAS = 0), the SRW bit is invalid. The MTX bit is read to determine the current transfer direction.

The following is an example of a software response by a master transmitter in the interrupt routine (see Figure 26-5 on page 26-16).

26.7.4 Generation of STOP

A data transfer ends when the master signals a STOP, which can occur after all data is sent. When the master receiver intends to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last data byte. This is done by setting the TXAK bit in the I2CR Register before reading the next-to-last byte. Before the last byte is read, a STOP signal must be generated.

26.7.5 Generation of Repeated START

After the data transfer is complete, the master can retain control of the bus by issuing a repeated START. Instead of sending a STOP signal, the master sends another START signal and sends out another slave calling address. See Figure 26-3 on page 26-4 for more information.

26.7.6 Slave Mode

When another device initiates communication on the bus, the MC9328MXL I²C module must verify whether it is addressed as the slave device by checking the I²C Addressed as a Slave bit (IAAS) in the I2SR Register. When IAAS is set, the software reads the SRW bit in the I2SR Register and sets the Transmit/Receive Mode Select bit (MTX) in the I2CR Register accordingly. Writing to the I2CR Register clears the IAAS bit automatically. The IAAS bit is set only at the end of the address cycle, even when there are multiple data bytes transferred.

Initiate a data transfer by writing data to the I2DR Register for slave transmits, or by reading data from the I2DR Register in slave receive mode. A dummy read of the I2DR Register in slave receive mode releases the SCL, allowing the master to send data.

In the slave transmitter routine, the Receive Acknowledge bit (RXAK) in the I2SR Register must be tested before sending the next byte of data. When RXAK is high, the master receiver intends to terminate the data transfer. The software must switch the MC9328MXL from transmitter to receiver mode. Reading the I2DR Register releases the SCL so the master can generate a STOP signal.

26.7.7 Arbitration Lost

When several devices try to engage the bus at the same time, one becomes the master and the hardware immediately switches the other devices to slave receive mode. Data output to the SDA line stops, however the serial clock continues to be generated until the end of the byte during which arbitration is lost. An interrupt occurs at the falling edge of the ninth clock of this transfer.

When a device that is not a master tries to transmit or generate a START, hardware automatically clears the Master/Slave Mode Select bit (MSTA) in the I2CR Register without signalling a STOP, generates an interrupt to the ARM920T processor, and sets the Arbitration Lost bit (IAL) in the I2SR Register to indicate a failed attempt to engage the bus. The slave service routine must first test the IAL bit and clear it when it is set.

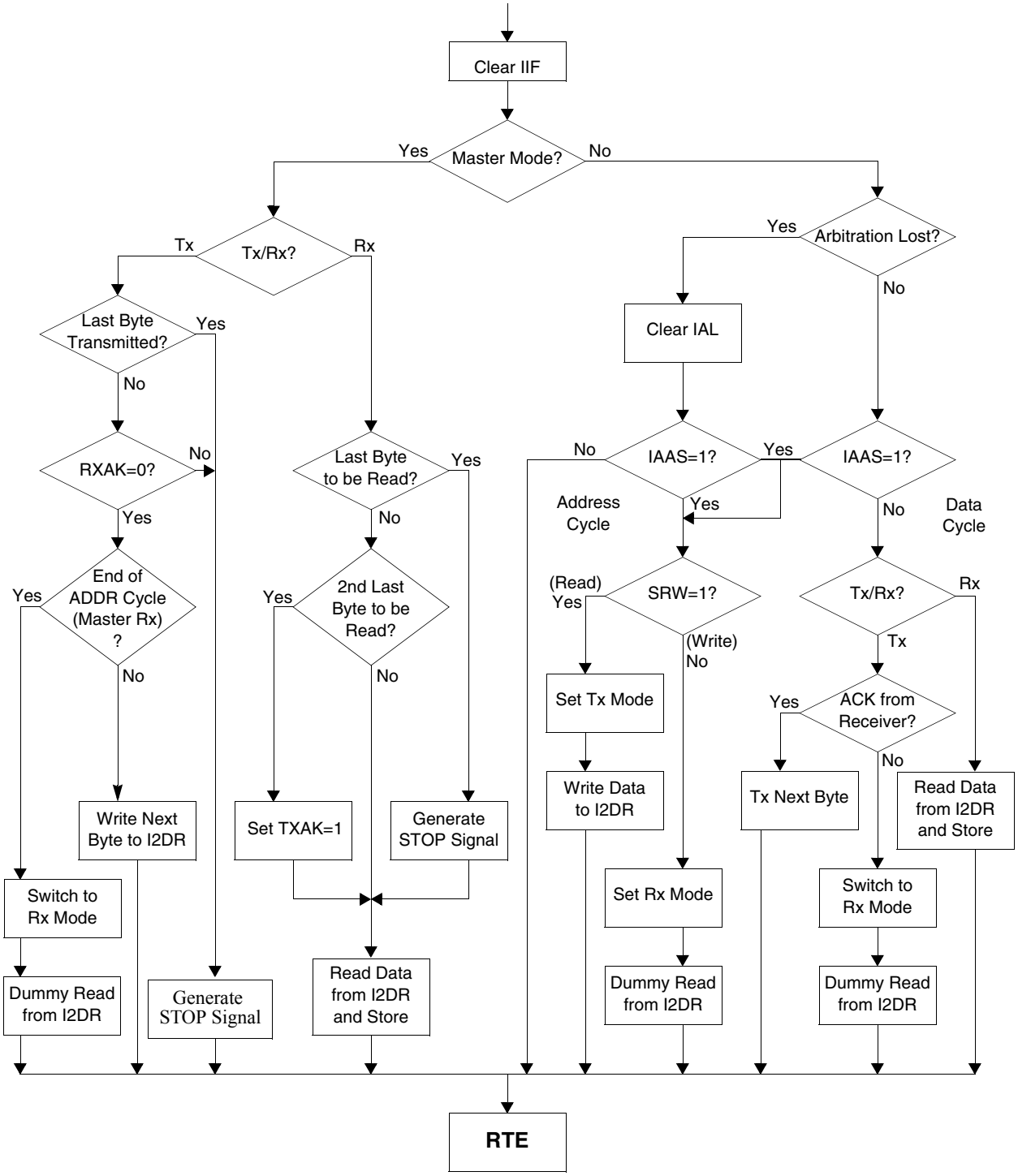


Figure 26-5. Flow Chart of Typical I²C Interrupt Routine

Chapter 27
Synchronous Serial Interface (SSI)

27.1 Introduction

This chapter describes the Synchronous Serial Interface (SSI) and provides descriptions of the architecture, programming model, operating modes, and initialization procedures of the SSI module. The SSI is a full-duplex serial port that allows the MC9328MXL to communicate with a variety of serial devices. These serial devices include standard codecs, digital signal processors (DSPs), microprocessors, peripherals that implement the Motorola Serial Peripheral Interface (SPI), and popular industry audio codecs that implement the inter-IC sound bus standard (I^2S).

The SSI typically transfers samples in a periodic manner. The SSI consists of independent transmitter and receiver sections with independent clock generation and frame synchronization functions.

The capabilities of the SSI include:

- Independent (asynchronous) or shared (synchronous) transmit and receive sections with separate or shared internal/external clocks and frame syncs, operating as master or slave
- Normal mode operation using frame sync
- Network mode operation allowing multiple devices to share the port with as many as 32 time slots
- Gated clock mode operation requiring no frame sync
- Programmable internal clock divider
- Programmable data interface modes such as I^2S, left-, and right-justified
- Programmable word length (8, 10, 12, or 16 bits)
- Program options for frame sync and clock generation
- Programmable I^2S mode (master, slave, or normal) selection
- Completely separate clock and frame sync selections for the receive and transmit sections
- Programmable oversampling output clock SYS_CLK (PerCLK3) of the sampling frequency in master mode which is available at the SSI_RXCLK pin when operated in sync mode.
- SSI power-down feature
- SSI signals are connected to Port B or Port C I/O pins

27.2 SSI Architecture

Figure 27-1 shows that the SSI functions can use Port B or Port C pins. For SSI output signals, the signal appears on both pins. However, SSI input or bidirectional signals require that the user configure one of the ports for the signal. This is done by setting up the desired pin in the Function Muxing Control Register (FMCR) in the system control module and then setting up the GPIO pins appropriately for Primary/Alternate and GPIO/Peripheral functions.For more information, see Section 27.2.3, on page 27-5.

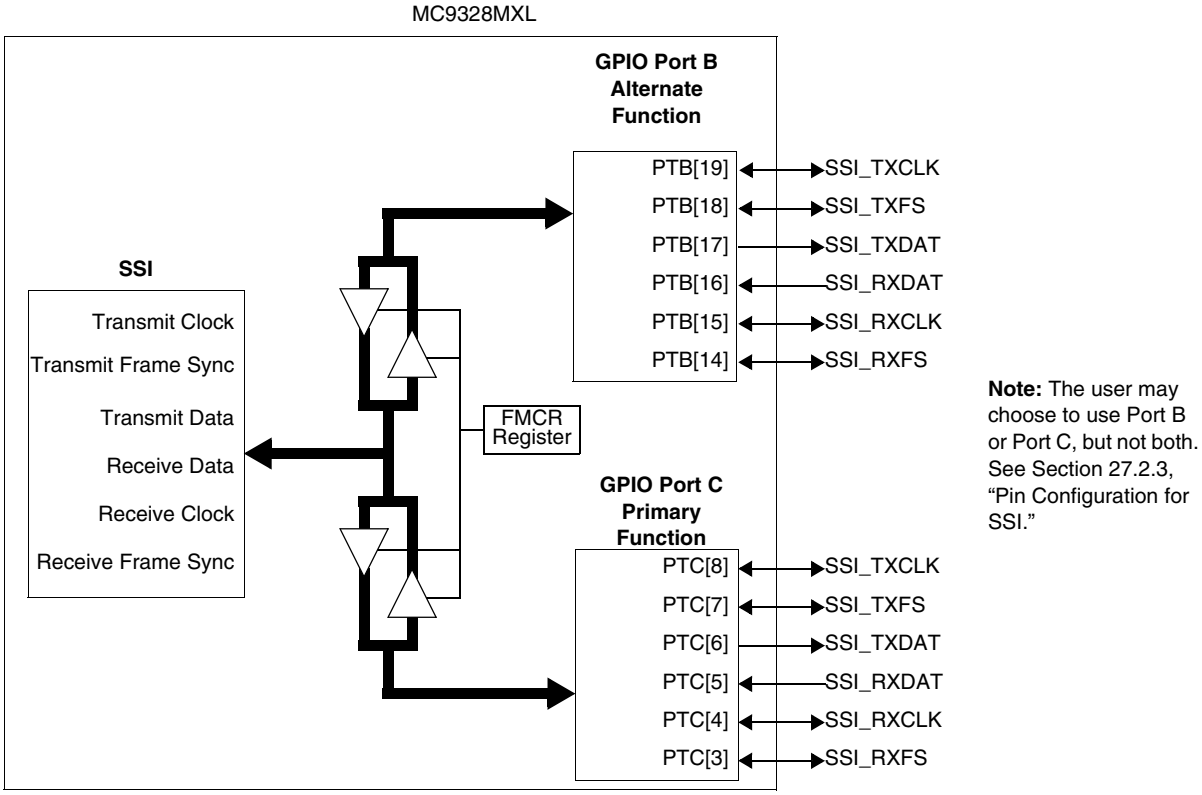


Figure 27-1. MC9328MXL SSI Input/Output Block Diagram

Figure 27-2 shows a block diagram of the SSI module. The SSI modules uses three control registers to configure the port, one status register, separate transmit and receive circuits with FIFO registers, and separate serial clock and frame sync generation for the transmit and receive sections.

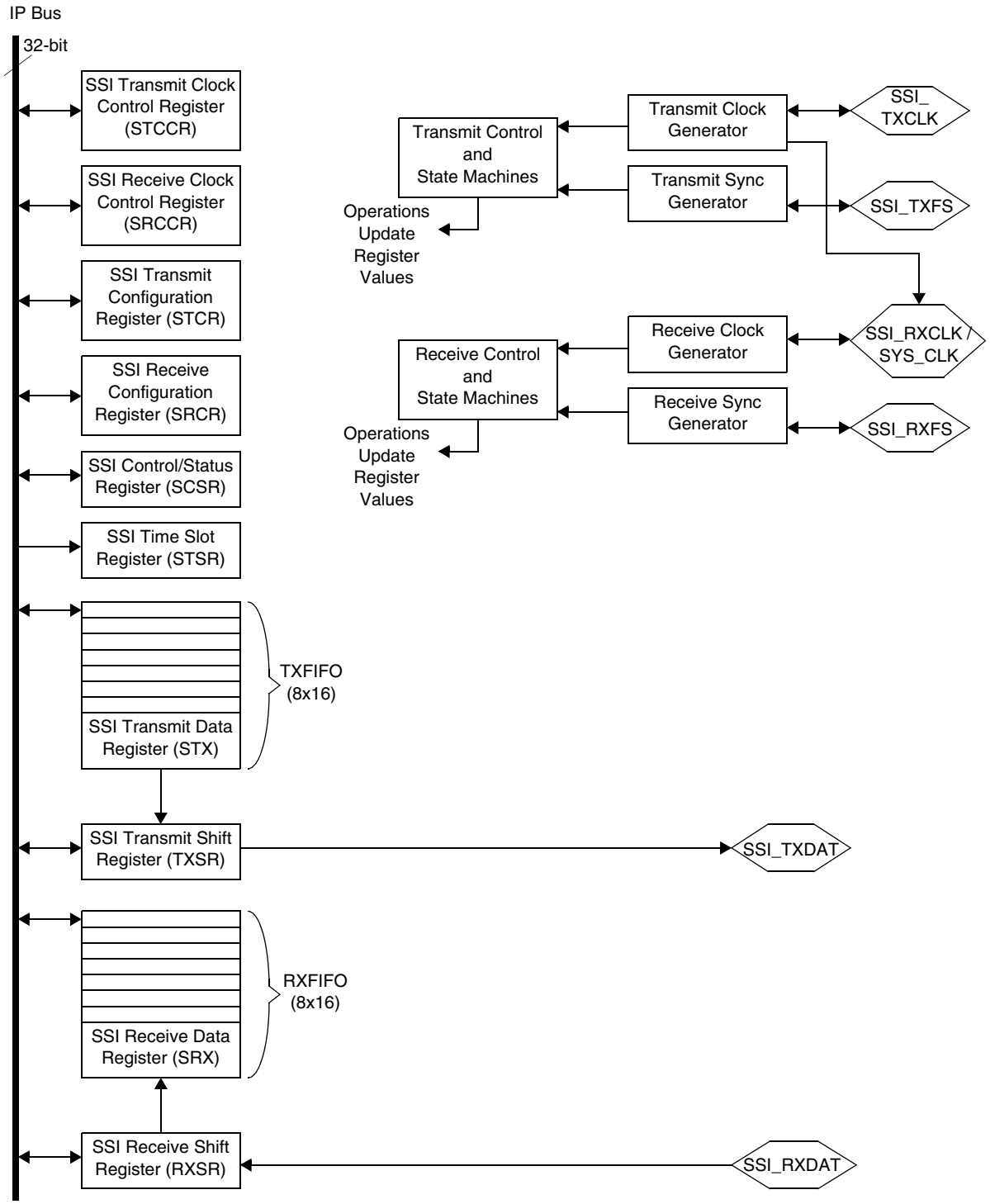


Figure 27-2. SSI Block Diagram

27.2.1 SSI Clocking

The SSI uses the following clocks:

- Serial bit clock—Serially clocks the data bits in and out of the SSI port
- Word clock—Counts the number of data bits per word (8, 10, 12, or 16 bits)
- Frame clock—Counts the number of words in a frame
- SYS_CLK—Input Clock from the PLL Clock Controller Module (PerCLK3). Made available on an output pin in synchronous master mode.

27.2.1.1 Normal Operating Mode

In normal operating mode, when the I^2S Mode Select bits (I2S_MODE1 and I2S_MODE0) in the SSI Control/Status Register (SCSR) are both clear, the serial bit clock is available on the serial transmit clock (SSI_TXCLK) and serial receive clock (SSI_RXCLK) pins. The word clock is an internal clock that determines when transmission of an 8-, 10-, 12-, or 16-bit word is complete. The word clock also clocks the frame clock, which counts the number of words in the frame. The frame sync clock is available on the SSI_TXFS and SSI_RXFS frame sync pins because a frame sync is generated after the correct number of words in the frame are transmitted/received. See Section 27.5, “SSI Operating Modes,” on page 27-38 for a detail description about the SSI operating modes.

27.2.1.2 Master / Synchronous Mode

In master mode and synchronous mode, the unused SSI_RXCLK pin outputs the serial system clock (SYS_CLK) enabled by the SYS_CLK_EN bit in the SSI Control/Status Register (SCSR). The SYS_CLK (PerCLK3) is the input clock into the SSI module. The SSI Clock Generator uses the word length (WL), prescaler range (PSR), prescaler modulus select (PM), and frame rate divider control (DC) to generate the other clocks from SYS_CLK (PerCLK3). The relationship between the clocks and the dividers is shown in Figure 27-3. A serial bit clock may be received from a SSI clock pin or can be generated internally from the PerCLK3 clock by a series of dividers, as shown in Figure 27-4.

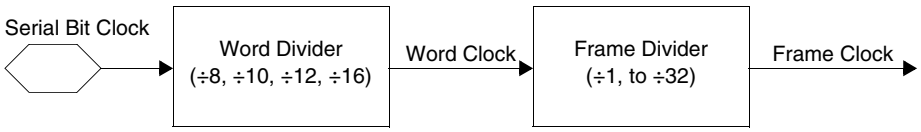


Figure 27-3. SSI Clocking

27.2.2 SSI Clock and Frame Sync Generation

Data clock and frame sync signals are generated internally by the MC9328MXL or can be obtained from external sources. When generated internally, the SSI clock generator derives bit clock and frame sync signals from an input clock signal. The SSI clock generator consists of a selectable, fixed prescaler and a programmable prescaler for bit rate clock generation. In gated clock mode, the data clock is valid only when data is being transmitted. If the pull-up is disabled for this pin in the GPIO Module’s Pull-Up Enable Register, then the clock pin is tri-stated when data is not transmitting.

A programmable frame rate divider and a word length divider are used for frame rate sync signal generation.

Figure 27-4 on page 27-5 shows a block diagram of the clock generator for the transmit section. Whether the serial bit clock is generated internally or derived from an external source depends on the transmit direction bit in the SSI Transmit Configuration Register (STCR). The receive section contains a similar clock generator circuit. However, for the receiver, because SSI_RXCLK is used for the receive clock, SYS_CLK (PerCLK3) is not made available on any output pins.

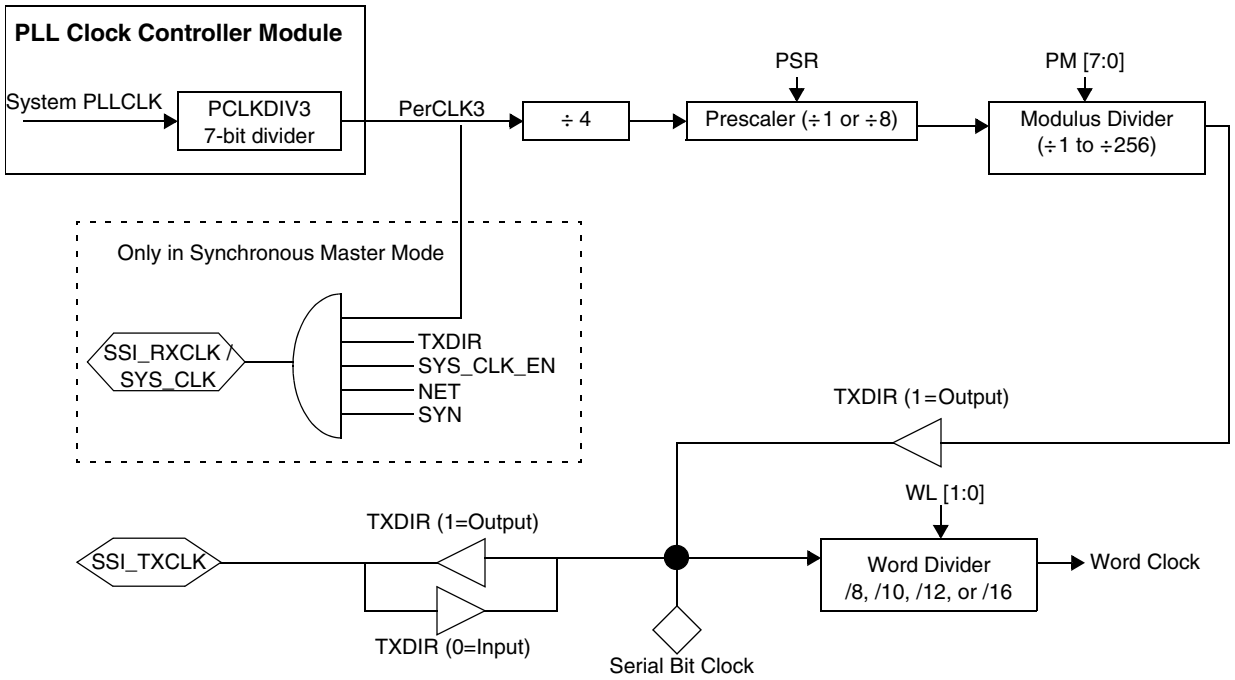


Figure 27-4. SSI Transmit Clock Generator Block Diagram

Figure 27-5 shows the frame sync generator block for the transmit section. When generated internally, both receive and transmit frame sync signals are generated from the word clock and are defined by the DC and WL bits of the SSI Transmit Clock Control Register (STCCR). The receive section contains an equivalent circuit for the frame sync generator.

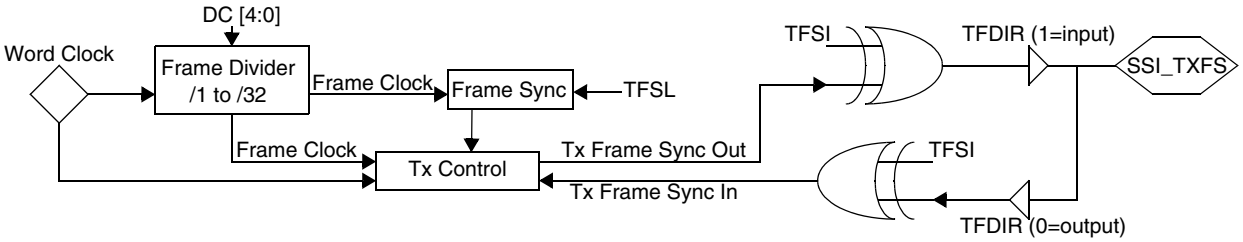


Figure 27-5. SSI Transmit Frame Sync Generator Block Diagram

27.2.3 Pin Configuration for SSI

Figure 27-1 illustrates the pins used for the SSI module. Table 27-1 on page 27-6 provides the pin configuration for the SSI module. These pins are multiplexed with other functions on the device, and must be configured for SSI operation.

NOTE:

The user must ensure that the data direction bits in the GPIO are set to the correct direction for proper operation. See Section 29.5.1, “Data Direction Registers,” on page 29-8 for details.

Table 27-1. SSI Pin Configuration

Pin	Setting[1]	Configuration Procedure
SSI_TXCLK	Primary function of GPIO Port C [8]	1. Clear bit 8 of Port C GPIO In Use Register (GIUS_C) 2. Clear bit 8 of Port C General Purpose Register (GPR_C) 3. Clear bit 3 in the Function Muxing Control Register (FMCR) in the System Control Module
	Alternate function of GPIO Port B [19]	1. Clear bit 19 of Port B GPIO In Use Register (GIUS_B) 2. Set bit 19 of Port B General Purpose Register (GPR_B) 3. Set bit 3 in the Function Muxing Control Register (FMCR) in the System Control Module
SSI_TXFS	Primary function of GPIO Port C [7]	1. Clear bit 7 of Port C GPIO In Use Register (GIUS_C) 2. Clear bit 7 of Port C General Purpose Register (GPR_C) 3. Clear bit 4 in the Function Muxing Control Register (FMCR) in the System Control Module
	Alternate function of GPIO Port B [18]	1. Clear bit 18 of Port B GPIO In Use Register (GIUS_B) 2. Set bit 18 of Port B General Purpose Register (GPR_B) 3. Set bit 4 in the Function Muxing Control Register (FMCR) in the System Control Module
SSI_TXDAT	Primary function of GPIO Port C [6]	1. Clear bit 6 of Port C GPIO In Use Register (GIUS_C) 2. Clear bit 6 of Port C General Purpose Register (GPR_C) (This is an output-only pin and therefore does not require a control signal in the Function Muxing Control Register (FMCR))
	Alternate function of GPIO Port B [17]	1. Clear bit 17 of Port B GPIO In Use Register (GIUS_B) 2. Set bit 17 of Port B General Purpose Register (GPR_B) (This is an output-only pin and therefore does not require a control signal in the Function Muxing Control Register (FMCR))
SSI_RXDAT	Primary function of GPIO Port C [5]	1. Clear bit 5 of Port C GPIO In Use Register (GIUS_C) 2. Clear bit 5 of Port C General Purpose Register (GPR_C) 3. Clear bit 5 in the Function Muxing Control Register (FMCR) in the System Control Module
	Alternate function of GPIO Port B [16]	1. Clear bit 16 of Port B GPIO In Use Register (GIUS_B) 2. Set bit 16 of Port B General Purpose Register (GPR_B) 3. Set bit 5 in the Function Muxing Control Register (FMCR) in the System Control Module

Table 27-1. SSI Pin Configuration (continued)

Pin	Setting[1]	Configuration Procedure
SSI_RXCLK	Primary function of GPIO Port C [4]	1. Clear bit 4 of Port C GPIO In Use Register (GIUS_C) 2. Clear bit 4 of Port C General Purpose Register (GPR_C) 3. Clear bit 6 in the Function Muxing Control Register (FMCR) in the System Control Module
	Alternate function of GPIO Port B [15]	1. Clear bit 15 of Port B GPIO In Use Register (GIUS_B) 2. Set bit 15 of Port B General Purpose Register (GPR_B) 3. Set bit 6 in the Function Muxing Control Register (FMCR) in the System Control Module
SSI_RXFS	Primary function of GPIO Port C [3]	1. Clear bit 3 of Port C GPIO In Use Register (GIUS_C) 2. Clear bit 3 of Port C General Purpose Register (GPR_C) 3. Clear bit 7 in the Function Muxing Control Register (FMCR) in the System Control Module
	Alternate function of GPIO Port B [14]	1. Clear bit 14 of Port B GPIO In Use Register (GIUS_B) 2. Set bit 14 of Port B General Purpose Register (GPR_B) 3. Set bit 7 in the Function Muxing Control Register (FMCR) in the System Control Module

1. Only one of the two pins should be set-up for each SSI signal.

27.2.3.1 Pin Configuration Example Software

Code Example 27-1 sets up the SSI pins as the alternate function of Port B by following the configuration in Table 27-1 on page 27-6.

Code Example 27-1. SSI Pin Setup

```
// Configure Pad to Peripheral Function instead of GPIO
LDR r1,=GIUS_B            // GPIO PORT B GIUS register
LDR r2,=0xFFF03FFF        // Pins 14-19 should be cleared
STR r2,[r1]

// Configure Pad for Alternate Function instead of Primary
LDR r1,=GPR_B             // GPIO PORT B GPR register
LDR r2,=0x000FC000        // Pins 14-19 should be set
STR r2,[r1]

// Configure FMCR to Select SSI input from SSI/SIM Pads
LDR r1,=FMCR              // CRM FMCR register
LDR r2,=0x000000F8        // Pins 7-3 should be set
STR r2,[r1]

// The SSI default is in pull-up state. Disable the pull-up, making them tristate
LDR r1,=PUEN_B            // GPIO PORTB PUEN register
LDR r2,=0xFFF03FFF        // Disable pullup for SSI input pins
STR r2,[r1]
```

27.3 Programming Model

The SSI module includes ten user-accessible 32-bit registers. It also includes two 16-bit internal registers and two 8×16 bit internal FIFOs. Table 27-2 summarizes these registers and their addresses.

Table 27-2. SSI Module Register Summary

Description	Name	Address
SSI Transmit Data Register	STX	0x00218000
SSI Receive Data Register	SRX_1	0x00218004
SSI Control/Status Register	SCSR	0x00218008
SSI Transmit Configuration Register	STCR	0x0021800C
SSI Receive Configuration Register	SRCR	0x00218010
SSI Transmit Clock Control Register	STCCR	0x00218014
SSI Receive Clock Control Register	SRCCR	0x00218018
SSI Time Slot Register	STSR	0x0021801C
SSI FIFO Control/Status Register	SFCSR	0x00218020
SSI Option Register	SOR	0x00218028
SSI Transmit and Receive FIFO Registers		
SSI Transmit Shift Register	TXSR	—
SSI Receive Shift Register	RXSR	—

NOTE:

To use SSI, the Port B General Purpose Register (GPR) and GPIO In Use (GIUS) registers must be set correctly. See Section 27.2.3, “Pin Configuration for SSI,” for more information.

27.3.1 SSI Transmit Data Register

The SSI Transmit Data Register (STX) holds the data to be transmitted. The data is transferred to the SSI Transmit Shift Register (TXSR), and when that register is empty, the data is shifted out onto the serial transmit data (SSI_TXDAT) pin.

The STXregister is implemented as the first word of the transmit FIFO. When the transmit FIFO enable (TFEN) bit in STCR is set, 8 data values are written to the STX register and these values are held in the transmit FIFO. They are then transmitted out one word at a time.

When the transmit interrupt enable (TIE) bit in the SSI Transmit Configuration Register (STCR1 is set, an interrupt occurs if no data is waiting to be transferred to the shift register (the transmit data empty (TDE) bit in the SCSR register is set) and the transmit register is empty.

When multiple writes to the STX register occur, data already in the register is not overwritten by incoming data. Multiple writes are accomplished as shown in the following examples:

- When the transmit FIFO is enabled and the user writes data1, data2, …, data9 to the STX register, data1, data2, …, data8 are stored in the FIFO, and data9 is discarded.
- When the transmit FIFO is disabled and user writes data1, data2 to the STX register, data2 is discarded.

NOTE:

Enable the SSI (SSI_EN = 1) in the SCSR before writing to the STX register.

STX SSI Transmit Data Register **Addr 0x00218000**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	HIGH BYTE								LOW BYTE							
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 27-3. SSI1 Transmit Data Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
HIGH BYTE Bits15–8	**Transmit Data High Byte**—Holds the high byte of the data to be transmitted out of the SSI.
LOW BYTE Bits 7–0	**Transmit Data Low Byte**—Holds the low byte of the data to be transmitted out of the SSI.

27.3.2 SSI Transmit FIFO Register

The SSI transmit FIFO is a 8×16-bit register that holds up to 8 words to be transmitted out by the SSI. The STX register is the first word of this FIFO, and data is transmitted first-in–first-out. The transmit FIFO is enabled by setting the Transmit FIFO Enable (TFEN) in the STCR register.

When the TIE bit in the STCR is enabled and the Transmit FIFO Empty (TFE) bit in the SCSR is set, an interrupt occurs if the data level in the transmit FIFO falls below the threshold value.

When the transmit FIFO is full, all further writes are ignored until the data is transmitted.

27.3.3 SSI Transmit Shift Register

The SSI Transmit Shift Register (TXSR) is a 16-bit shift register containing the data ready to be transmitted.

When a continuous clock is used, data is shifted out to the SSI_TXDAT pin by the selected (internal/external) bit clock when the associated (internal/external) frame sync is asserted. When a gated clock is used, data is shifted out to the SSI_TXDAT pin by the selected (internal/external) gated clock.

The WL bits in the STCCR register determine the number of bits that will be shifted out of the TXSR before it is considered empty and can be written to again. This word length can be 8, 10, 12, or 16 bits.

The Transmit Shift Direction (TSHFD) bit and Transmit Bit Position(TXBIT0) in the STCR determines how the data is transmitted. Table 27-4 displays the data bit shifting configuration and Figure 27-6 through Figure 27-9 visually the data path for each configuration.

Table 27-4. Data Bit Shifting Configuration

TXBIT0	TSHFD	WL[1:0]	Shifting from Bit
0	0	xx	Bit 15 (MSB) first
0	1	00	Bit 8 (LSB) first, bit 15 (MSB) last
0	1	01	Bit 6 (LSB) first, bit 15 (MSB) last
0	1	10	Bit 4 (LSB) first, bit 15 (MSB) last
0	1	11	Bit 0 (LSB) first, bit 15 (MSB) last
1	0	00	Bit 7 (MSB) first, bit 0 (LSB) last
1	0	01	Bit 9 (MSB) first, bit 0 (LSB) last
1	0	10	Bit 11 (MSB) first, bit 0 (LSB) last
1	0	11	Bit 15 (MSB) first, bit 0 (LSB) last
1	1	xx	Bit 0 (LSB) first

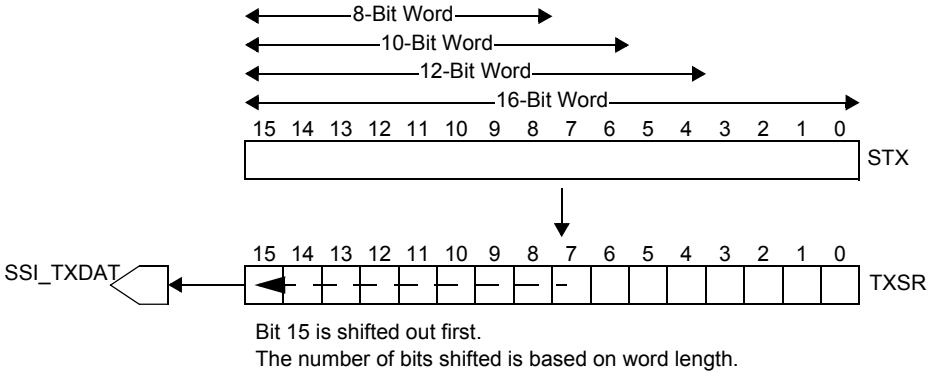


Figure 27-6. Transmit Data Path (TXBIT0 = 0, TSHFD = 0)

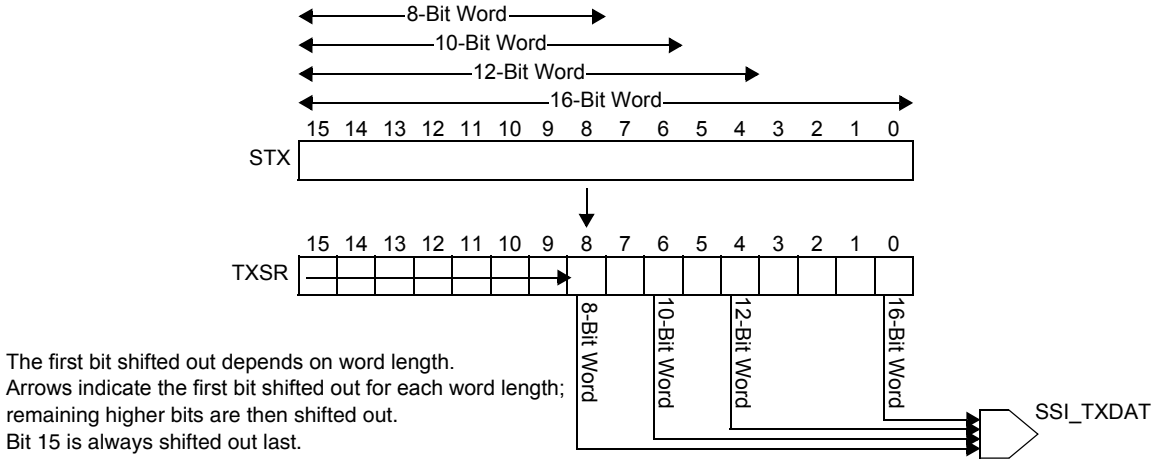


Figure 27-7. Transmit Data Path (TXBIT0 = 0, TSHFD = 1)

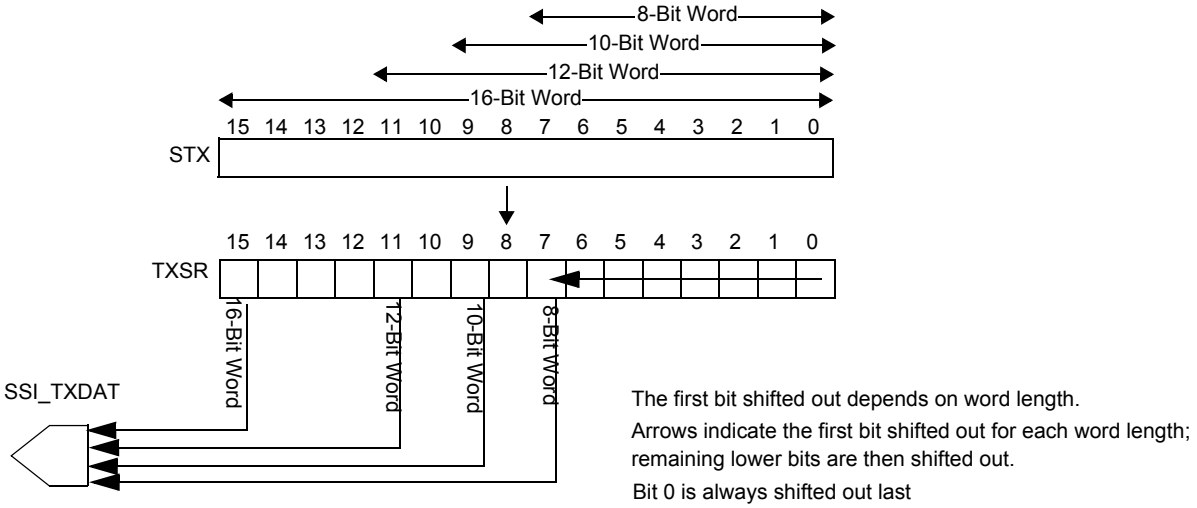


Figure 27-8. Transmit Data Path (TXBIT0 = 1, TSHFD = 0)

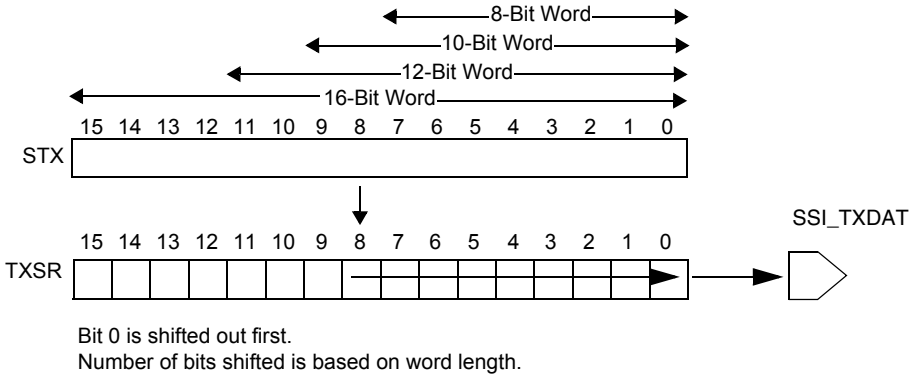


Figure 27-9. Transmit Data Path (TXBIT0 = 1, TSHFD = 1)

27.3.4 SSI Receive Data Register

The read-only SSI Receive Data Register (SRX) holds the most recently received data. Incoming data from the serial receive data (SSI_RXDAT) pin is held in the SSI Receive Shift Register (RXSR) as it is received. When the transmission is complete, the data is transferred to the SRX register.

The SRX register is implemented as the first word of the receive FIFO. When the receive FIFO is enabled in the SSI Receive Configuration Register (SRCR), 8 data values are received into the SSI receive FIFO.

When the receive interrupt enable (RIE) bit in the SRCR is set, an interrupt occurs when a new value is loaded into the SRX register from the RXSR.

SRX SSI Receive Data Register **Addr 0x00218004**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	HIGH BYTE								LOW BYTE							
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 27-5. SSI Receive Data Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
HIGH BYTE Bits 15–8	**Receive Data High Byte**—Holds the high byte of the data received from the RXSR.
LOW BYTE Bits 7–0	**Receive Data Low Byte**—Holds the low byte of the data received from the RXSR.

27.3.5 SSI Receive FIFO Register

The SSI receive FIFO is an 8×16-bit register that holds the last 8 words received by the SSI. The SRX register is the first word of this FIFO. The receive FIFO is enabled by setting the Receive FIFO Enable (RFEN) in the SRCR

When the RIE bit is enabled and the Receive FIFO Full (RFF) bit in the SCSR is set, an interrupt occurs if the data level in the receive FIFO reaches the threshold value.

When the receive FIFO is full, all further received data is ignored until the data is read out.

27.3.6 SSI Receive Shift Register

The SSI Receive Shift Register (RXSR) is a 16-bit shift register that contains the data being received from the SSI_RXDAT pin. When the register is full, received data fills the receive FIFO.

When a continuous clock is used, data is shifted in by the selected (internal/external) bit clock when the associated (internal/external) frame sync is asserted. When a gated clock is used, data is shifted in from the SSI_TXDAT pin by the selected (internal/external) gated clock.

The Receive shift direction (RSHFD) bit and Receive bit position(RXBIT0) in the SRCR determines how the data is stored. Table 27-6 contains all of the information about data bit shifting configurations and Figure 27-10 through Figure 27-13 visually the data path for each configuration.

The WL bits in the SSI Receive Clock Control Register (SRCCR) determine the number of bits to be shifted in from the SSI_RXDAT pin. This word length can be 8, 10, 12, or 16 bits. When receiving 8, 10, or 12 bits of data, 0s are appended to the end of the data string to fill the 16-bit register.

Table 27-6. Data Bit Shifting Configuration

RXBIT0	RSHFD	WL[1:0]	Shifting to bit
0	0	00	Bit 15 (MSB) first, bit 8 (LSB) last
		01	Bit 15 (MSB) first, bit 6 (LSB) last
		10	Bit 15 (MSB) first, bit 4 (LSB) last
		11	Bit 15 (MSB) first, bit 0 (LSB) last
0	1	00	Bit 8 (MSB) first, bit 15 (LSB) last
		01	Bit 6 (MSB) first, bit 15 (LSB) last
		10	Bit 4 (MSB) first, bit 15 (LSB) last
		11	Bit 0 (MSB) first, bit 15 (LSB) last
1	0	00	Bit 0 (MSB) first, bit 7 (LSB) last
		01	Bit 0 (MSB) first, bit 9 (LSB) last
		10	Bit 0 (MSB) first, bit 11 (LSB) last
		11	Bit 0 (MSB) first, bit 15 (LSB) last
1	1	00	Bit 7 (MSB) first, bit 0 (LSB) last
		01	Bit 9 (MSB) first, bit 0 (LSB) last
		10	Bit 11 (MSB) first, bit 0 (LSB) last
		11	Bit 15 (MSB) first, bit 0 (LSB) last

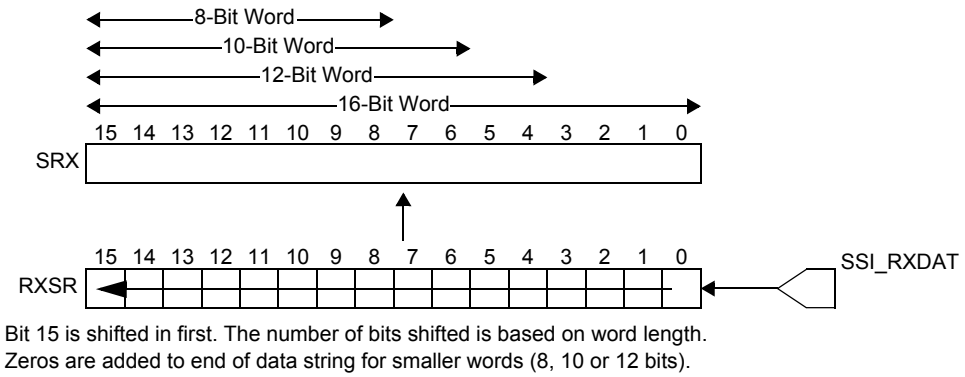


Figure 27-10. Receive Data Path (RXBIT0 = 0, RSHFD = 0)

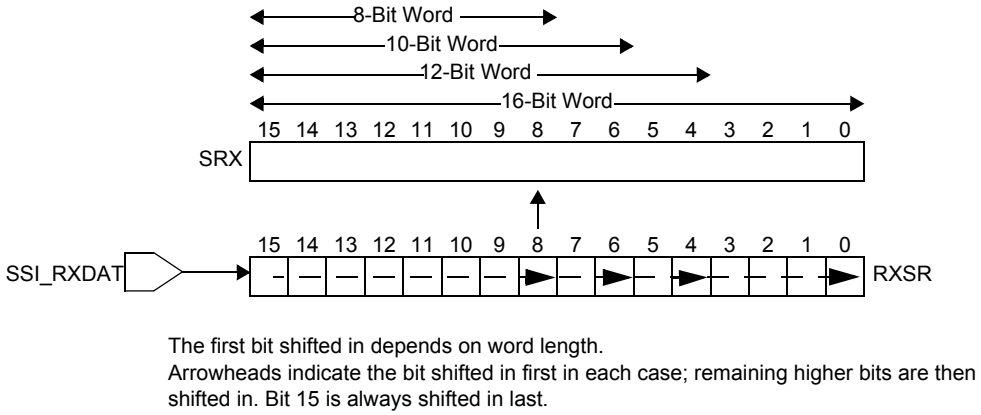


Figure 27-11. Receive Data Path (RXBIT0 = 0, RSHFD = 1)

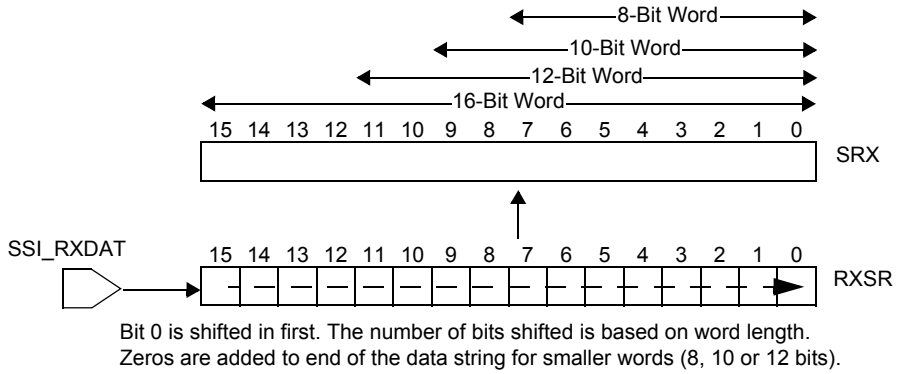
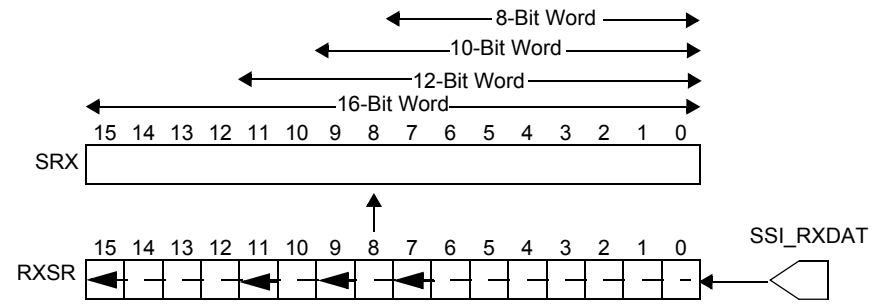


Figure 27-12. Receive Data Path (RXBIT0 = 1, RSHFD = 0)



The first bit shifted in depends on word length.
Arrowheads indicate the bit shifted in first in each case; all lower bits are then shifted in.
Bit 0 is always shifted in last.

Figure 27-13. Receive Data Path (RXBIT0 = 1, RSHFD = 1)

27.3.7 SSI Control/Status Register

The SSI Control/Status Register sets up and monitors the SSI. The SSI status bits are updated when the SSI is enabled, and then after the transmission or reception of the first bit of the next SSI word is complete. The Receive Overrun Error (ROE) and Transmitter Underrun Error (TUE) bits are cleared by reading this register, followed by a read of the SRX register (to clear ROE), or a write to the STX register (to clear TUE).

SCSR SSI Control/Status Register **Addr 0x00218008**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SYS_CLK_EN	I2S_MODE		SYN	NET	RE	TE	SSI_EN	RDR	TDE	ROE	TUE	TFS	RFS	RFF	TFE
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
	0x0041															

Table 27-7. SSI Control/Status Register Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	

Table 27-7. SSI Control/Status Register Description (continued)

Name	Description	Settings
SYS_CLK_EN Bit 15	**System Clock Enable**—Determines whether the input clock PerCLK3 is output as SYS_CLK on the SSI_RXCLK pin. When SYS_CLK_EN is set, PerCLK3 is output on the SRCK pin if the network mode, synchronous mode, and transmit internal clock mode bits are also set. The timing relationship between SYS_CLK and the internal serial bit clock is controlled by the STCCR and/or SRCCR registers.	0 = SYS_CLK is not output on the SSI_RXCLK pin 1 = SYS_CLK is output on the SSI_RXCLK pin
I2S_MODE Bits 14–13	**I^2S Mode Select**—Determines whether the SSI operates in normal mode, I^2S master mode, or I^2S slave mode.	See Section 27.3.7.1, “I2S Mode Selection.”
SYN Bit 12	**Synchronous Mode**—Enables/Disables the synchronous mode of operation. In synchronous mode, the transmit and receive sections share a common clock pin (SSI_TXCLK) and frame sync pin (SSI_TXFS).	0 = Disable synchronous mode 1 = Enable synchronous mode
NET Bit 11	**Network Mode**—Selects the operational mode of the SSI. See Section 27.5.2, “Network Mode,” for more information on Network Mode.	0 = Normal mode 1 = Network mode
RE Bit 10	**Receive Enable**—Enables/Disables the receive portion of the SSI. When disabled data transfers into the SRX register/receive FIFO are inhibited. If data is being received when the RE bit is disabled, the rest of the word is ignored. If the RE bit is re-enabled before the second-to-last bit of the same word is received, the word is received.	0 = Disable SSI receive 1 = Enable SSI receive
TE Bit 9	**Transmit Enable**—Enables/Disables the transfer of the contents of the STX register to the TXSR. When the TE bit is set and a word boundary is detected, the transmit portion of the SSI is enabled. When the TE bit is cleared, the transmitter continues to send the data currently in the TXSR and then disables the transmitter. The serial output is disabled and any data present in the STX register is not transmitted. Data can be written to the STX register with the TE bit cleared. The TDE bit is cleared, however data is not transferred to the TXSR. When the TE bit is cleared and then set again during the same transmitted word, the data continues to be transmitted. When the TE bit is set again during a different time slot, data is not transmitted until the next word boundary. The normal transmit enable sequence is to write data to the STX register or to the SSI Time Slot Register (STSR) before setting the TE bit. The normal transmit disable sequence is to clear the TE bit and the TIE bit in the STCR after the TDE bit is set. When an internal gated clock is used, the gated clock runs during valid time slots when the TE bit is set. When the TE bit is cleared, the transmitter continues to send the data currently in the TXSR until the TXSR register is empty, then the clock stops. When the TE bit is set again, the gated clock starts immediately and runs during any valid time slots.	0 = Disable SSI transmit 1 = Enable SSI transmit

Table 27-7. SSI Control/Status Register Description (continued)

Name	Description	Settings
SSI_EN Bit 8	**SSI Enable**—Enables/Disables the SSI. The SSI Enable bit must be set prior to setting other bits. When the SSI is set up for internal frame sync, enabling causes an output frame sync to be generated. When the SSI is set up for external frame sync, enabling causes the SSI to wait for the input frame sync. When the SSI is disabled, the SSI_TXCLK, SSI_TXFS, and SSI_TXFS pins are tri-stated, the status register bits hold their reset value, and all internal clocks are disabled (other than clocking for register access). Reset the SSI by clearing the SSI_EN bit. The contents of the transmit FIFO and receive FIFO are cleared when the SSI_EN bit is reset.	0 = Disable the SSI 1 = Enable the SSI
RDR Bit 7	**Receive Data Ready**—Indicates that the SRX register or the receive FIFO contains a new value. It is automatically cleared when the CPU reads the SRX register or when the receive FIFO (when enabled) is empty. When the RIE bit in the SRCR is set, an interrupt occurs when the RDR bit is set.	0 = SRX does not contain a new value or the receive FIFO is empty 1 = SRX or receive FIFO contains a new value
TDE Bit 6	**Transmit Data Register Empty**—Indicates that no data is waiting to be transferred to the TXSR. This occurs when the STX register is empty. Because the STX register is the first word of the transmit FIFO, the TDE bit is not set until the transmit FIFO is empty. When the TDE bit is set and data is not written to the STX register or to the SSI Time Slot Register (STSR) before the TXSR empties, an underrun error occurs. To clear the TDE bit, write transmission data to the STX register or write any data to the STSR register. When the TIE bit in the STCR is set, an interrupt occurs when the TDE bit is set.	0 = Data is waiting to be transmitted (transferred to the TXSR) 1 = No data is waiting to be transmitted (transferred to the TXSR)
ROE Bit 5	**Receive Overrun Error**—Indicates when the RXSR is filled and ready to transfer to the SRX register or the receive FIFO register (when enabled), and these registers are already full. This is also indicated by the RFF and Receive Data Ready (RDR) bits of this register. When ROE is set, the contents of the RXSR are not transferred. However, when the ROE bit is set, it causes a change in the interrupt vector used, allowing the use of a different interrupt handler for a receive overrun condition. When a receive interrupt occurs with the ROE bit set, the Receive Data with Exception interrupt is generated. When a receive interrupt occurs with the ROE bit cleared, the Receive Data interrupt is generated. ROE is cleared by reading this register, and then reading the SRX register.	0 = SRX and/or receive FIFO can hold more data 1 = RXSR is ready to transfer data, however SRX and/or receive FIFO is full

Table 27-7. SSI Control/Status Register Description (continued)

Name	Description	Settings
TUE Bit 4	**Transmitter Underrun Error**—Indicates that the TXSR is empty AND a transmit time slot occurs. The TDE bit in this register indicates the first condition. When a transmit underrun error occurs, the data that was in the TXSR is retransmitted. In normal mode, a transmit time slot occurs when the frame sync is asserted. In network mode, each time slot requires data transmission and is a transmit time slot (TE = 1). When the TUE bit is set, there is no data transferred to the TXSR. However, the TUE bit does cause a change in the interrupt vector used for transmit interrupts so that a different interrupt handler can be used for a transmit underrun condition. When a transmit interrupt occurs with the TUE bit set, the Transmit Data with Exception interrupt is generated. When a transmit interrupt occurs with the TUE bit cleared, the Transmit Data interrupt is generated. TUE is cleared by reading this register, and then writing to the STX register or to the SSI Time Slot Register (STSR).	0 = TXSR is empty and no transmit time slot has occurred OR the TXSR is not empty 1 = TXSR is empty and a transmit time slot has occurred
TFS Bit 3	**Transmit Frame Sync**—Indicates that a frame sync occurred during transmission of the last word written to the STX register. Data written to the STX register during the time slot when the TFS bit is set is sent during the second time slot (in network mode) or in the next first time slot (in normal mode). In network mode, the TFS bit is set during transmission of the first slot of the frame. It is then cleared when starting transmission of the next slot.	0 = No frame sync occurred during transmission 1 = A frame sync occurred during transmission
RFS Bit 2	**Receive Frame Sync**—Indicates that a frame sync occurred when receiving a word into the SRX register. In network mode, the RFS bit is set when the first slot of the frame is being received. It is cleared when the next slot of the frame begins to be received.	0 = No frame sync occurred when receiving 1 = A frame sync occurred when receiving
RFF Bit 1	**Receive FIFO Full**—Indicates that the data level in the receive FIFO reaches the Receive FIFO Full Water Mark. The Water Mark is defined in the RFWM field of the SSI FIFO Control/Status Register (SFCSR). The receive FIFO must be enabled or RFF is meaningless. When RFF is set, data can be read from the receive FIFO via the SRX register. When the RX FIFO has received 8 bytes of data, all further received data is ignored until the data is read out of the receive FIFO. **Note:** An interrupt is generated only when both the RFF and RIE bits in the SRCR are set and the receive FIFO (the RFEN bit in the SRCR) is enabled.	0 = Data level in the receive FIFO exceeds Water Mark level 1 = Data level in receive FIFO reaches Water Mark

Table 27-7. SSI Control/Status Register Description (continued)

Name	Description	Settings
TFE Bit 0	**Transmit FIFO Empty**—Indicates that the data level in the transmit FIFO reaches the Transmit FIFO Empty Water Mark. The Water Mark is defined in the TFWM field of the SSI FIFO Control/Status Register (SFCSR). The transmit FIFO must be enabled or TFE is meaningless. When TFE is set, data can be written to the transmit FIFO via the STX register. **Note:** An interrupt is generated only when both the TFE and TIE (of the STCR) are set and the transmit FIFO is enabled (the TFEN bit in the STCR is set).	0 = Data level in the transmit FIFO exceeds Water Mark 1 = Data level in the transmit FIFO below Watermark

27.3.7.1 I^{2}S Mode Selection

The SSI is able to emulate certain features of the I^{2}S standard. Both early frame sync and 1-word-wide frame sync are supported. Data can be up to 16-bits per channel. However, dummy clocks are not supported. If the data is 16-bit and there are 2 channels per frame, the number of bit clock should be exactly 32. Use of dummy clocks is forbidden.

The SCSRcontains two bits, the I^{2}S Mode Select (I2S MODE1 and I2S MODE0) bits, that determine the mode of the SSI module. This section explains how the mode of the module affects the bits in the other SSI registers. Table 27-8 summarizes the mode settings.

Table 27-8. I^{2}S Mode Selection

I2S_MODE [1]	I2S_MODE [0]	Remark
0	0	Normal mode
0	1	I^{2}S master mode
1	0	I^{2}S slave mode
1	1	Normal mode

In normal mode operation, no register bits are forced to any particular state internally and the SSI can be programmed to work in any operating condition.

When entering I^{2}S modes (I^{2}S master or I^{2}S slave), several control bits are fixed in the hardware. Attempts to write to these bits when in I^{2}S master or I^{2}S slave mode are ignored. These bits are described in Table 27-9.

The user must configure the bit clock in the STCCR and the SRCCR.

Table 27-9. I^{2}S Master or I^{2}S Slave Mode Settings

I^{2}S Mode	Bit Name	Register Location	Forced Value	Function
Master or slave	SYN	SCSR [12]	1	Synchronous mode enabled
Master or slave	NET	SCSR [11]	1	Network mode enabled

Table 27-9. I²S Master or I²S Slave Mode Settings (continued)

I²S Mode	Bit Name	Register Location	Forced Value	Function
Master or slave	TSHFD	STCR [4]	0	Transmission direction is MSB first
Master or slave	RSHFD	SRCR [4]	0	Receive direction is MSB first
Master or slave	TSCKP	STCR [3]	1	Falling edge of bit clock clocks data out
Master or slave	RSCKP	SRCR [3]	1	Rising edge of bit clock clocks data in
Master or slave	TFSI	STCR [2]	1	Transmit frame sync is active low
Master or slave	RSFI	SRCR [2]	1	Receive frame sync is active low
Master or slave	TFSL	STCR [1]	0	Transmit frame sync length is 1 word
Master or slave	RFSL	SRCR [1]	0	Receive frame sync length is 1 word
Master or slave	TEFS	STCR [0]	1	Transmit frame sync initiated sync one bit-clock before transmission
Master or slave	REFS	SRCR [0]	1	Receive frame sync initiated one bit-clock before receive
Master	TXDIR	STCR [5]	1	Clock source is generated internally bit clock
Master	TFDIR	STCR [6]	1	Transmit frame sync is generated internally
Slave	TXDIR	STCR [5]	0	Clock source is externally generated bit clock
Slave	TFDIR	STCR [6]	0	Transmit frame sync is generated externally

When the SSI is configured as an I²S master, the external codec may require an oversampling clock in addition to the sampling rate frequency and bit clock. This oversampling clock is usually required to be 256 times the sampling frequency.

As an example, if a sampling frequency of 44.1 kHz was required, and the word length was set to 16-bits. Then a frame sync would be 2×16 = 32 bit clocks. This makes the bit clock frequency 1.4112 MHz.

The frequency ratios that must hold would be the following:

$$\text{SYS_CLK (PerCLK3)} = 256 \times \text{Frame Sync} \qquad \textit{Eqn. 27-1}$$

$$\text{Frame_Sync} = \text{Bit Clock} / 32 \qquad \textit{Eqn. 27-2}$$

Therefore,

$$\text{SYS_CLK (PerCLK3)} = 8 \times \text{Bit Clock} \qquad \textit{Eqn. 27-3}$$

If the divide ratio is set to 8—that is, PM = 1 and PSR = 0, SYS_CLK_EN is set, the module is in I²S master mode and the incoming frequency of PerCLK3 is 11.2896 MHz, then SYS_CLK will meet the requirements. The external codec will not need a crystal oscillator for clocking.

27.3.8 SSI Transmit Configuration Register

The SSI Transmit Configuration Register (STCR) controls the transmit operation of the SSI. This register controls the frame synchronization signal, clocking, and data direction. It also contains enables for the DMA, transmit FIFO, and the transmit interrupt.

As with all on-chip peripheral interrupts for the MC9328MXL, the STCR must first be set to enable maskable interrupts. Next, the AITC (ARM9 Interrupt Controller) is configured to handle the SSI interrupts. For example, the SSI interrupt bits (bits 45 through 42) in the AITC's Interrupt Enable High Register (INTENABLEH) must be set to enable the interrupt. Also be sure to enable interrupts to the core (either the IRQ or FIQ interrupts). After all of these steps are completed, an interrupt is generated when any of the desired transmit status bits of the SCSR (TDE, TUE, TFS, or TFE) are set.

NOTE:

SSI reset does not affect the STCR bits. Power-on-reset clears all STCR bits.

STCR SSI Transmit Configuration Register **Addr 0x0021800C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
										0x0000						

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						TXBIT0	TDMAE	TIE	TFEN	TFDIR	TXDIR	TSHFD	TSCKP	TFSI	TFSL	TEFS
TYPE	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
										0x0000						

Table 27-10. SSI Transmit Configuration Register Description

Name	Description	Settings
Reserved Bits 31–11	Reserved—These bits are reserved and should read 0.	
TXBIT0 Bit 10	**Transmit Bit0**—This bit determines which bit in the Transmit Shift Register (TXSR) triggers the TXSR to transmit its data word. By default, shifting of data is triggered by bit position 15 of TXSR. The shifting data direction (MSB or LSB bit transmitted first) is controlled by the TSHFD bit.	0 = Bit position 15 of TXSR triggers transfer 1 = Bit position 0 of TXSR triggers transfer

Table 27-10. SSI Transmit Configuration Register Description (continued)

Name	Description	Settings
TDMAE Bit 9	**Transmit DMA Enable**—Enables DMA requests to be issued when certain conditions are met. When the transmit FIFO is enabled and the TDMAE bit is set, a DMA request is issued when the TFE bit in the SCSR is set. When the transmit FIFO is disabled and the TDMAE bit is set, a DMA request is issued when the TDE bit in the SCSR is set. The TIE bit (in this register) has higher priority than the TDMA bit. When TIE is set, an interrupt is generated to the CPU instead of the DMA.	0 = No DMA request issued 1 = DMA request issued
TIE Bit 8	**Transmit Interrupt Enable**—Enables/Disables the transmit interrupt when certain conditions are met. TIE works with the Transmit Enable (TE) bit in the SCSR. When the transmit FIFO is enabled and the TIE and TE bits are both set, an interrupt occurs when the TFE bit in the SCSR register is set. When the transmit FIFO is disabled and the TIE and TE bits are both set, an interrupt occurs when the TDE bit in the SCSR register is set. Two transmit data interrupts with separate interrupt vectors are available: transmit data with exception status and transmit data without exceptions. Table 27-11 shows the conditions that generate these interrupts and lists the interrupt vectors.	0 = Transmit interrupt disabled 1 = Transmit interrupt enabled
TFEN Bit 7	**Transmit FIFO Enable**—Enables/Disables the transmit FIFO. When the transmit FIFO is enabled, a maximum of 8 values are written to the STX register at a time. These values are then transmitted out first-in–first-out. When the transmit FIFO is disabled, only one value is written to the STX register at a time.	0 = Transmit FIFO disabled 1 = Transmit FIFO enabled
TFDIR Bit 6	**Transmit Frame Direction**—Selects the direction and source of the transmit frame sync signal. When the TFDIR bit is set, the frame sync is generated internally and output to the SSI_TXFS pin (if not configured as GPIO). When the TFDIR bit is cleared, the transmit frame sync is supplied from an external source. Table 27-14 shows the clock pin configuration.	0 = Frame sync generated externally 1 = Frame sync generated internally
TXDIR Bit 5	**Transmit Direction**—Selects the direction and source of the clock signal that clocks the TXSR. When the TXDIR bit is set, the clock is generated internally and output to the SSI_TXCLK pin (if not configured as GPIO). When the TXDIR bit is cleared, the clock source is external, the internal clock generator is disconnected from the SSI_TXCLK pin, and an external clock source can drive this pin to clock the TXSR.	0 = External source must drive SSI_TXCLK pin 1 = Clock generated internally and output to SSI_TXCLK pin
TSHFD Bit 4	**Transmit Shift Direction**—Controls whether the MSB or LSB is transmitted first for the transmit section. **Note:** The codec device labels the MSB as bit 0, however the MC9328MXL labels the LSB as bit 0. When using a standard codec, the MC9328MXL MSB (or codec bit 0) is shifted out first, and the TSHFD bit is cleared.	0 = MSB transmitted first 1 = LSB transmitted first

Table 27-10. SSI Transmit Configuration Register Description (continued)

Name	Description	Settings
TSCKP Bit 3	**Transmit Clock Polarity**—Controls the bit clock edge that clocks out data for the transmit functions.	0 = Rising edge of clock clocks data out 1 = Falling edge of clock clocks data out
TFSI Bit 2	**Transmit Frame Sync Invert**—Selects the logic of the transmit frame sync I/O.	0 = Active high 1 = Active low
TFSL Bit 1	**Transmit Frame Sync Length**—Selects the length of the frame sync signal to be generated or recognized. When the word-long frame sync is selected, this frame sync length is determined by the WL field of the STCCR.	0 = Frame sync is 1 word long 1 = Frame sync is 1 bit-clock long
TEFS Bit 0	**Transmit Early Frame Sync**—Controls when the frame sync is initiated for the transmit section. The frame sync is disabled after one bit for bit-length frame sync and after one word for word-length frame sync (based on the TFSL bit of this register).	0 = Frame sync is initiated with first data bit transmission 1 = Frame sync is initiated one bit-clock before transmission starts

Table 27-11. SSI Transmit Data Interrupts

Interrupt	Interrupt Source Register High[1]	TIE	TUE	TFE/TDE
Transmit Data with Exception	Bit 11 (IN43)	1	1	1
Transmit Data Without Exception	Bit 10 (IN42)	1	0	1

1. Refer to Chapter 10, “Interrupt Controller (AITC),” for more information about interrupts.

27.3.9 SSI Receive Configuration Register

The SSI Receive Configuration Register (SRCR) controls the receive operation of the SSI—which controls the frame synchronization signal, clocking, and data direction configurations. It also contains enables for the DMA, receive FIFO, and the receive interrupt settings.

As with all on-chip peripheral interrupts for the MC9328MXL, the SRCR must first be set to enable maskable interrupts. Next, the AITC (ARM9 Interrupt Controller) is configured to handle the SSI interrupts. For example, the SSI interrupt bits (bits 45 through 42) in the AITC’s Interrupt Enable High Register (INTENABLEH) must be set to enable the interrupt. Also be sure to enable interrupts to the core (either the IRQ or FIQ interrupts). When all of these steps are complete, then an interrupt is generated when any of the desired transmit status bits of the SCSR (RDR, ROE, RFS, or RFF) are set.

NOTE:

An SSI reset does not affect the bits in the SRCR register. Power-on-reset clears all SRCR register bits.

SRCR — SSI Receive Configuration Register

Addr 0x00218010

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
									0x0000							

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						RXBIT0	RDMAE	RIE	RFEN	RFDIR	RXDIR	RSHFD	RSCKP	RFSI	RFSL	REFS
TYPE	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
									0x0000							

Table 27-12. SSI Receive Configuration Register Description

Name	Description	Settings
Reserved Bits 31–11	Reserved—These bits are reserved and should read 0.	
RXBIT0 Bit 10	**Receive Bit 0**—This bit determines which bit in the Receive Shift Register (RXSR) triggers the RXSR to transmit its data word. By default, shifting of data is triggered by bit position 15 of TXSR. The shifting data direction (MSB or LSB bit transmitted first) is controlled by the RSHFD bit.	0 = Bit position 15 of RXSR triggers transfer 1 = Bit position 0 of RXSR triggers transfer
RDMAE Bit 9	**Receive DMA Enable**—Enables DMA requests to be issued when certain conditions are met. When the receive FIFO is enabled and the RDMAE bit is set, a DMA request is issued when the RFF bit in the SCSR is set. When the receive FIFO is disabled and the RDMAE bit is set, a DMA request is issued when the RDR bit in the SCSR is set. The RIE bit (in this register) has higher priority than the RDMAE bit. When the RIE bit is set, an interrupt is generated to the CPU instead of the DMA.	0 = No DMA request issued 1 = DMA request issued

Table 27-12. SSI Receive Configuration Register Description (continued)

Name	Description	Settings
RIE Bit 8	**Receive Interrupt Enable**—Enables/Disables the receive interrupt when certain conditions are met. The RIE bit works with the RE bit in the SCSR. When the receive FIFO is enabled and the RIE and RE bits are both set, an interrupt occurs when the RFF bit in the SCSR is set. When the receive FIFO is disabled and the RIE and RE bits are both set, an interrupt occurs when the RDR bit in the SCSR is set. Two receive data interrupts with separate interrupt vectors are available, Receive Data with Exception and Receive Data Without Exception. Table 27-13 on page 27-26 shows the conditions that generate these interrupts and lists the interrupt vectors.	0 = Disable transmit interrupt 1 = Enable transmit interrupt
RFEN Bit 7	**Receive FIFO Enable**—Enables/Disables the receive FIFO. When the receive FIFO is enabled, a maximum of 8 values can be received by the SSI without losing data. A ninth value can be received into the RXSR, however it is not transferred to the receive FIFO. When the receive FIFO is disabled, only one value can be received into the SRX register (that value must be read out before another value can be transferred in from the RXSR).	0 = Disable receive FIFO 1 = Enable transmit FIFO
RFDIR Bit 6	**Receive Frame Direction**—Selects the direction and source of the receive frame sync signal. When the RFDIR bit is set, the frame sync is generated internally and output to the SSI_RXFS pin (if not configured as a GPIO). When the RFDIR bit is cleared, the receive frame sync is supplied from an external source.	0 = Receive frame sync generated externally 1 = Receive frame sync generated internally
RXDIR Bit 5	**Receive Direction**—Selects the direction and source of the clock signal that clocks the RXSR. When the RXDIR bit is set, the clock is generated internally and output to the SSI_RXCLK pin (if not configured as a GPIO). When the RXDIR bit is cleared, the internal clock generator is disconnected from the SSI_RXCLK pin so an external clock source can drive this pin to clock the RXSR. Table 27-14 on page 27-26 shows the clock pin configuration.	0 = External source drives SSI_RXCLK pin 1 = Clock generated internally and output to SSI_RXCLK pin
RSHFD Bit 4	**Receive Shift Direction**—Controls whether the MSB or LSB is received first. **Note:** The codec device labels the MSB as bit 0, whereas the MC9328MXL labels the LSB as bit 0. When using a standard codec, the MC9328MXL MSB (or codec bit 0) is shifted in first, and the RSHFD bit is cleared.	0 = MSB received first 1 = LSB received first
RSCKP Bit 3	**Receive Clock Polarity**—Controls the bit clock edge that latches in data.	0 = Falling edge of clock latches data in 1 = Rising edge of clock latches data in
RFSI Bit 2	**Receive Frame Sync Invert**—Selects the logic of the receive frame sync I/O.	0 = Active high 1 = Active low
RFSL Bit 1	**Receive Frame Sync Length**—Selects the length of the frame sync signal to be generated or recognized. When the word-long frame sync is selected, this frame sync length is determined by the WL field of the SRCCR.	0 = Frame sync is 1 word long 1 = Frame sync is 1 bit-clock long

Table 27-12. SSI Receive Configuration Register Description (continued)

Name	Description	Settings
REFS Bit 0	**Receive Early Frame Sync**—Controls when the frame sync is initiated for the receive section. The frame sync is disabled after one bit for bit-length frame sync and after one word for word-length frame sync (based on the RFSL bit of this register).	0 = Frame sync is initiated with first data bit received 1 = Frame sync is initiated one bit-clock before data received

Table 27-13. SSI Receive Data Interrupts

Interrupt	Interrupt Source Register High[1]	RIE	ROE	RFF/RDR
Receive Data with Exception	Bit 13 (IN45)	1	1	1
Receive Data Without Exception	Bit 12 (IN44)	1	0	1

1. Refer to Chapter 10, “Interrupt Controller (AITC),” for more information about interrupts.

Table 27-14. Clock Pin Configuration

SYN	RXDIR	TXDIR	RFDIR	TFDIR	SSI_RXFS	SSI_TXFS	SSI_RXCLK	SSI_TXCLK
Asynchronous Mode[1]								
0	0	0	0	0	Receive Frame Sync (RFS) in	Transmit Frame Sync (TFS) in	Receive Clock (RCK) in	Transmit Clock (TCK) in
0	0	1	0	1	RFS in	TFS out	RCK in	TCK out
0	1	0	1	0	RFS out	TFS in	RCK out	TCK in
0	1	1	1	1	RFS out	TFS out	RCK out	TCK out
Synchronous Mode[2]								
1	0	0	x	0	GPIO	TFS in	GPIO	TCK in
1	0	1	x	1	GPIO	TFS out	GPIO	TCK out
1	1	0	x	x	GPIO	GPIO	GPIO	Gated Clock in
1	1	1	x	x	GPIO	GPIO	GPIO	Gated Clock out

1. See Figure 27-14 on page 27-36.
2. See Figure 27-15 on page 27-37.

27.3.10 SSI Transmit Clock Control Register and SSI Receive Clock Control Register

The SSI Transmit Clock Control Register (STCCR) and SSI Receive Clock Control Register (SRCCR) control the clocks for the SSI. These registers control the prescaler, word length, and frame rate for the transmit and receive clocks. The fields of this register control the dividers that generate the serial bit clock, the word clock and the frame clock from the PerCLK3 input. See Section 27.3.10.1, “Calculating the SSI Bit Clock from the Input Clock Value,” for detailed information on clock values.

The STCCR is dedicated to the transmit section, and the SRCCR register is dedicated to the receive section (except in synchronous mode, when the STCCR controls both the receive and transmit sections). The bit structure for both registers is identical, however they are two distinct registers and must be individually programmed.

NOTE:

SSI reset does not affect the STCCR and SRCCR bits. Power-on reset clears all STCCR and SRCCR bits.

		Addr
STCCR	SSI Transmit Clock Control Register	**0x00218014**
SRCCR	SSI Receive Clock Control Register	**0x00218018**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PSR	WL		DC					PM							
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 27-15. SSI Transmit Clock Control Register and SSI Receive Clock Control Register Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
PSR Bit 15	**Prescaler Range**—Selects whether the fixed divide-by-eight prescaler will be used to generate the serial bit clock. See Figure 27-4. Using this prescaler allows a 128 kHz master clock to be generated for Motorola MC1440x series codecs.	0 = Bypass divide-by-eight prescaler 1 = Use divide-by-eight prescaler

Table 27-15. SSI Transmit Clock Control Register and SSI Receive Clock Control Register Description

Name	Description	Settings
WL Bits 14–13	**Word Length**—Selects the length (8, 10, 12, or 16 bits) of the data words being transferred by the SSI. WL also controls the frame sync pulse length when the length of the frame sync (SRCR:RFSL or STCR:TFSL) is set to 1 word. The value of this field is used as a divider value to convert serial bit clock to word clock as shown in Figure 27-4.	00 = 8 Bits per word 01 = 10 Bits per word 10 = 12 Bits per word 11 = 16 Bits per word
DC Bits 12–8	**Frame Rate Divider Control**—Specifies the divide ratio for the programmable frame rate divider shown in Figure 27-5. The divider converts the word clock to the frame clock. In normal mode, this ratio determines the word transfer rate. In network mode, this ratio sets the number of words per frame. In network mode, a divide ratio of one (DC = 00000) is a special case (on-demand mode). In normal mode, a divide ratio of one (DC = 00000) provides continuous periodic data word transfer (a bit-length sync must be used).	00000 = Divide Ratio is 1 00001 = Divide Ratio is 2 … 11111 = Divide Ratio is 32 In network mode, DC = 00000 is a special case.
PM Bits 7–0	**Prescale Modulus Select**—Specifies the divide ratio for the modulus divider shown in Figure 27-4. This divider is one of three dividers that convert the PerCLK3 signal to the serial bit clock.	0x00 = Divide ratio is 1 0x01 = Divide ratio is 2 ... 0xFF = Divide ratio is 256

27.3.10.1 Calculating the SSI Bit Clock from the Input Clock Value

The serial bit clock is the result of the Clock Controller input PerCLK3 being divided by one fixed, one selectable, and one programmable divider. Serial bit clock may range from the value (PerCLK3 / 4×1×1) to (PerCLK3 / (4×8×256)). See Figure 27-4 on page 27-5. Note that WL has a value of 8, 10, 12, or 16.

$$f_{INT_BIT_CLK} = f_{PerCLK3} \div [4 \times (7 \times PSR + 1) \times (PM + 1)] \qquad \textit{Eqn. 27-4}$$

$$f_{FRAME_SYN_CLK} = (f_{INT_BIT_CLK}) \div [(DC + 1) \times WL] \qquad \textit{Eqn. 27-5}$$

Example 1: When the 7-bit clock controller divider, PCLKDIV3 in the PLL and Clock Control module, is set to 5, then PerCLK3 is 96 MHz ÷ 5 = 19.2 MHz.

The SSI is set up in normal mode with a word length of 8 (WL = 00), a frame rate divider control of 1 (DC = 0001), a prescale modulus of 74 (decimal) (PM = 01001010), and the PSR bit cleared (PSR = 0).

In this case, the bit clock rate is 19.2 MHz ÷ (4 × 1 × 75) = 64 kHz.

That means that the frame sync clock (SSI_TXFS) is 64 kHz ÷ (2 × 8) = 4 kHz.

Example 2: When the 7-bit clock controller divider, PCLKDIV3, is set to 8, then PerCLK3 is 96 MHz ÷ 8 = 12 MHz.

The SSI is set up in network mode, with a word length of 16 (WL = 11), a frame rate divider control of 1 (DC = 0001), a prescale modulus of 1 (decimal) (PM = 00000001), and the PSR bit cleared (PSR = 0).

In this case, the bit clock rate is 12 MHz ÷ (4 × 1 × 2) = 1.5 MHz.

That means that the frame sync clock (SSI_TXFS) is 1.5 MHz ÷ (2 × 16) = 46.875 kHz.

Table 27-16 shows various combinations of the PSR and PM fields that achieve different bit clock frequencies.

Table 27-16. SSI Bit and Frame Clock as a Function of PSR and PM in Normal Mode

System PLL output in the clock Controller Module (MHz)	System Clock Divider PCLKDIV3 (7 Bit)	PerCLK3 (MHz) Input to the SSI Module	PSR (0 or 1)	PM [7:0]	Actual Bit_Clk Frequency (kHz) SSI_TXCLK, SSI_RXCLK	Ideal Bit_Clk Frequency (kHz)
96	15	6.4	1	24(0x18)	8.0	8.0
96	10	9.6	1	24(0x18)	12.0	12.0
96	20	4.8	0	74(0x4A)	16.0	16.0
96	5	19.2	0	74(0x4A)	64.0	64.0
96	5	19.2	0	37(0x25)	126.3	128.0
96	2	48.0	0	46(0x2E)	255.3	256.0
96	1	96.0	0	46(0x2E)	510.6	512.0
96	1	96.0	0	23(0x0F)	1000.0	1024
96	1	96.0	0	11(0x0B)	2000.0	2048
96	1	96.0	0	5(0x05)	4000.0	4096

Table 27-17 shows the effect of changing the clock controller divider value to generate SYS_CLK and bit clock frequencies close to the ideal. In these examples, master mode was selected by setting the I^2S Mode Select (I2S_MODE1 and I2S_MODE0) bits in the SCSR to 01, or individually programming the SSI into network, synchronous, and transmit internal modes.

Table 27-17. SSI Sys, Bit and Frame Clock in Master Mode

Ideal Sampling Rate (kHz)	Ideal Bit Clock Frequency (kHz)	Ideal SYS_CLK Frequency (MHz)	CRM Clock Controller Divider Value (with 96 MHz input)	PerCLK3 (MHz)	Actual SYS_CLK Frequency (MHz) SSI_RXCLK	Actual Bit Clock Frequency (kHz) SSI_TXCLK	Actual Sampling Rate (kHz) SSI_TXFS
8.0	256.0	2.048	47	2.043	2.043	255.32	7.98
11.025	352.8	2.822	32	3.000	3.000	375.00	11.72
16.00	512.0	4.096	23	4.173	4.173	510.64	15.96
22.05	705.6	5.644	17	5.647	5.647	705.88	22.06
32.00	1024.0	8.192	12	8.000	8.000	1000.0	31.25
44.10	1411.0	11.289	9	10.677	10.677	1334.63	41.71
48.00	1536.0	12.288	8	12.000	12.000	1500.00	48.88

27.3.11 SSI Time Slot Register

The write-only SSI Time Slot Register (STSR) is used when data should not be transmitted in an available transmit time slot. For the purposes of timing, the time slot register behaves like an alternate transmit data register. Instead of transmitting data, the SSI_TXDAT pin is either tri-stated or pulled high, depending on the setting for that pin in the Pull-Up Enable Register in the GPIO Module. Using this register is important for avoiding overflow/underflow during inactive time slots.

STSR SSI Time Slot Register **Addr 0x0021801C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DUMMY															
TYPE	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 27-18. SSI Time Slot Register Description

Name	Description
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.
DUMMY Bits 15–0	**Dummy Bits**—Holds data that is not intended for transmission. This register can be used during inactive time slots.

27.3.12 SSI FIFO Control/Status Register

The SSI FIFO Control/Status Register (SFCSR) tracks the number of data words held in the transmit and receive FIFOs and holds the Water Mark levels for each FIFO. The Water Mark is a user-defined value that determines when the RFF and TFE bits in the SCSR are set.

SFCSR SSI FIFO Control/Status Register **Addr 0x00218020**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RFCNT				TFCNT				RFWM				TFWM			
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
								0X0081								

Table 27-19. SSIFIFO Control/Status Register Description

Name	Description	Settings
Reserved Bits 31–16	Reserved—These bits are reserved and should read 0.	
RFCNT Bits 15–12	**Receive FIFO Counter**—Indicates how many data words are in the receive FIFO. This count does not include words that are being received or were received into the RXSR_1 yet not transferred to the receive FIFO.	0000 = 0 data words in the receive FIFO 0001 = 1 data word in the receive FIFO 0010 = 2 data words in the receive FIFO 0011 = 3 data words in the receive FIFO 0100 = 4 data words in the receive FIFO 0101 = 5 data words in the receive FIFO 0110 = 6 data words in the receive FIFO 0111 = 7 data words in the receive FIFO 1000 = 8 data words in the receive FIFO All other settings reserved
TFCNT Bits 11–8	**Transmit FIFO Counter**—Indicates how many data words are in the transmit FIFO. This count does not include words that were transferred to the TXSR.	0000 = 0 data words in the transmit FIFO 0001 = 1 data word in the transmit FIFO 0010 = 2 data words in the transmit FIFO 0011 = 3 data words in the transmit FIFO 0100 = 4 data words in the transmit FIFO 0101 = 5 data words in the transmit FIFO 0110 = 6 data words in the transmit FIFO 0111 = 7 data words in the transmit FIFO 1000 = 8 data words in the transmit FIFO All other settings reserved

Table 27-19. SSIFIFO Control/Status Register Description (continued)

Name	Description	Settings
RFWM Bits 7–4	**Receive FIFO Full Water Mark**— Specifies the Receive FIFO Water Mark. When the data level of the receive FIFO reaches the Water Mark level, the RFF bit in the SCSR register is set.	0000 = Reserved 0001 = RFF sets when at least one data word is written to the receive FIFO (RFCNT = 1, 2, 3, 4, 5, 6, 7, 8 data words) 0010 = RFF sets when 2 or more data words are written to the receive FIFO (RFCNT = 2, 3, 4, 5, 6, 7, 8 data words) 0011 = RFF sets when 3 or more data words are written to the receive FIFO (RFCNT = 3, 4, 5, 6, 7, 8 data words) 0100 = RFF sets when 4 or more data words are written to the receive FIFO (RFCNT = 4, 5, 6, 7, 8 data words) 0101 = RFF sets when 5 or more data words are written to the receive FIFO (RFCNT = 5, 6, 7, 8 data words) 0110 = RFF sets when 6 or more data words are written to the receive FIFO (RFCNT = 6, 7, 8 data words) 0111 = RFF sets when 7 or more data words are written to the receive FIFO (RFCNT = 7, 8 data words) 1000 = RFF sets when exactly 8 data words are written to the receive FIFO (RFCNT = 8 data words) All other settings reserved
TFWM Bits 3–0	**Transmit FIFO Empty Water Mark**— Specifies the Transmit FIFO Water Mark. When the data level of the transmit FIFO falls below the Water Mark level, the TFE bit in the SCSR register is set.	0000 = Reserved 0001 = TFE sets when there are 1 or more empty slots in the transmit FIFO (TFCNT = 7, 6, 5, 4, 3, 2, 1, 0 data words) 0010 = TFE sets when there are 2 or more empty slots in the transmit FIFO (TFCNT = 6, 5, 4, 3, 2, 1, 0 data words) 0011 = TFE sets when there are 3 or more empty slots in the transmit FIFO (TFCNT = 5, 4, 3, 2, 1, 0 data words) 0100 = TFE sets when there are 4 or more empty slots in the transmit FIFO (TFCN = 4, 3, 2, 1, 0 data words) 0101 = TFE sets when there are 5 or more empty slots in the transmit FIFO (TFCNT = 3, 2, 1, 0 data words) 0110 = TFE sets when there are 6 or more empty slots in the transmit FIFO (TFCNT = 2, 1, 0 data words) 0111 = TFE sets when there are 7 or more empty slots in the transmit FIFO (TFCNT = 1, 0 data words) 1000 = TFE sets when there are 8 empty slots are in the transmit FIFO (TFCNT = 0 data words) All other settings reserved

Table 27-20. Value of Transmit FIFO Empty (TFE) and Receive FIFO Full (RFF)

Transmit FIFO Water Mark (TFWM) or Receive FIFO Water Mark (RFWM)	Number of Data Words in TXFIFO									Number of Data Words in RXFIFO								
	0	1	2	3	4	5	6	7	8	0	1	2	3	4	5	6	7	8
1 (0001)	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1
2 (0010)	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
3 (0011)	1	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1
4 (0100)	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1
5 (0101)	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1
6 (0110)	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
7 (0111)	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
8 (1000)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

27.3.13 SSI Option Register

The SSI Option Register (SOR) allows the user to clear out the receive FIFO and/or the transmit FIFO, turn the clock off when the SSI is disabled, and reset the frame synchronization and the state machine. This register also includes a user-programmable field to set wait states.

SOR SSI Option Register **Addr 0x00218028**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
									0x0000							

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										CLK OFF	RX_CLR	TX_CLR				SYN RST
TYPE	r	r	r	r	r	r	r	r	r	rw	rw	rw	r	r	r	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
									0x0000							

Table 27-21. SSI Option Register Description

Name	Description	Settings
Reserved Bits 31–7	Reserved—These bits are reserved and should read 0.	
CLKOFF Bit 6	**Clock Off**—Turns off the clocks when the SSI is disabled to further reduce power consumption.	0 = Clocks enabled when SSI disabled 1 = Clocks disabled when SSI disabled
RX_CLR Bit 5	**Receiver Clear**—Controls whether the receive FIFO is flushed. The transmit portion of the SSI is not affected. The software must clear RX_CLR before the SSI transmitter can operate.	0 = No effect 1 = Clear receive FIFO
TX_CLR Bit 4	**Transmitter Clear**—Controls whether the transmit FIFO is flushed. The receiver portion of the SSI is not affected. The software must clear TX_CLR before the SSI transmitter can operate.	0 = No effect 1 = Clear transmit FIFO
Reserved Bits 3–1	Reserved—These bits are reserved and should read 0.	
SYNRST Bit 0	**Frame Sync Reset**—Resets the accumulation of data in the SRX register and receive FIFO (RXFIFO) on frame synchronization.	0 = No effect 1 = Reset data accumulation

27.4 SSI Data and Control Pins

The SSI has six I/O pins. The pins are shared with either Port B or Port C pins, depending on the configuration of the ports. See Section 27.2.3, “Pin Configuration for SSI,” for more information. Each pin is described in detail in the sections following Table 27-22.

Table 27-22. SSI Pin Description

Pin Description	Pin Name	Port C Multiplexed Pin	Port B Multiplexed Pin
Serial Transmit Data	SSI_TXDAT	PTC [6]	PTB [17]
Serial Receive Data	SSI_RXDAT	PTC [5]	PTB [16]
Serial Transmit Clock	SSI_TXCLK	PTC [8]	PTB [19]
Serial Receive Clock	SSI_RXCLK	PTC [4]	PTB [15]
Serial Transmit Frame Sync	SSI_TXFS	PTC [7]	PTB [18]
Serial Receive Frame Sync	SSI_RXFS	PTC [3]	PTB [14]

27.4.1 SSI_TXDAT, Serial Transmit Data

The SSI_TXDAT pin transmits data from the TXSR. While data is being transmitted, the SSI_TXDAT pin is an output pin. This pin is disabled between data word transmissions and on the trailing edge of the bit clock after the last bit of a word is transmitted. This pin has an internal pull-up controlled by the GPIO’s PUEN bit. For more information, refer to the Chapter 29, “GPIO Module and I/O Multiplexer (IOMUX).”

27.4.2 SSI_RXDAT, Serial Receive Data

The SSI_RXDAT pin brings serial data into the SSI Receive Shift Register (RXSR).

27.4.3 SSI_TXCLK, Serial Transmit Clock

The SSI_TXCLK pin can be used as either an input or an output. This clock signal is used by the transmitter and can be either continuous or gated. During gated clock mode, data on the SSI_TXCLK pin is valid only during the transmission of data. If the pull-up is disabled for this pin in the GPIO Module’s Pull-Up Enable Register, then the clock pin is tri-stated when data is not transmitting. In synchronous mode, this pin is used by both the transmit and receive sections. When using gated clock mode, an external resistor is connected to this pin to prevent the signal from floating when not being driven.

27.4.4 SSI_RXCLK, Serial Receive Clock

The SSI_RXCLK pin can be used as either an input or an output. This clock signal is used by the receiver and is always continuous. During gated clock mode, the SSI_TXCLK pin is used instead for clocking in data. In I^2S master mode, this pin is used as an output pin for the oversampling clock, SYS_CLK (PerCLK3).

27.4.5 SSI_TXFS, Serial Transmit Frame Sync

The SSI_TXFS pin can be used as either an input or an output. The frame sync is used by the transmitter to synchronize the transfer of data. The frame sync signal can be 1 bit-clock or 1 word in length and can occur one bit before the transfer of data or with the transfer of data. (These configurations are set in the STCR.) In synchronous mode, this pin is used by both the transmit and receive sections. In gated clock mode, frame sync signals are not used. When SSI_TXFS is configured as input, the external device should drive SSI_TXFS at the rising or falling edge of SSI_TXCLK, depending on the setting of the TSCKP bit of the SSI Transmit Configuration Register. SSI_TXFS should sync with the rising edge of SSI_TXCLK if the data is clocking out at the rising edge of SSI_TXCLK. SSI_TXFS should sync with the falling edge of SSI_TXCLK if the data is clocking out at the falling edge of SSI_TXCLK.

27.4.6 SSI_RXFS, Serial Receive Frame Sync

The SSI_RXFS pin can be used as either an input or an output. The frame sync is used by the receiver to synchronize the transfer of data. The frame sync signal can be 1 bit-clock or 1 word in length and can occur one bit before the transfer of data or with the transfer of data. (These configurations are set in the SRCR.) When SSI_RXFS is configured as an input, the external device can drive SSI_RXFS at the rising or falling edge of SSI_RXCLK, depending on the setting of the RSCKP bit of the SRCR Register. SSI_RXFS should sync with the rising edge of SSI_RXCLK if the data is clocking out at the rising edge of SSI_RXCLK. SSI_RXFS should sync with the falling edge of SSI_RXCLK if the data is clocking out at the falling edge of SSI_RXCLK.

Figure 27-14 and Figure 27-15 on page 27-37 show the main SSI configurations. These pins support all transmit and receive functions with continuous or gated clocks as shown. Note that gated clock implementations do not require the use of the frame sync pins (SSI_TXFS and SSI_RXFS). In this case, these pins also can be used as GPIO pins.

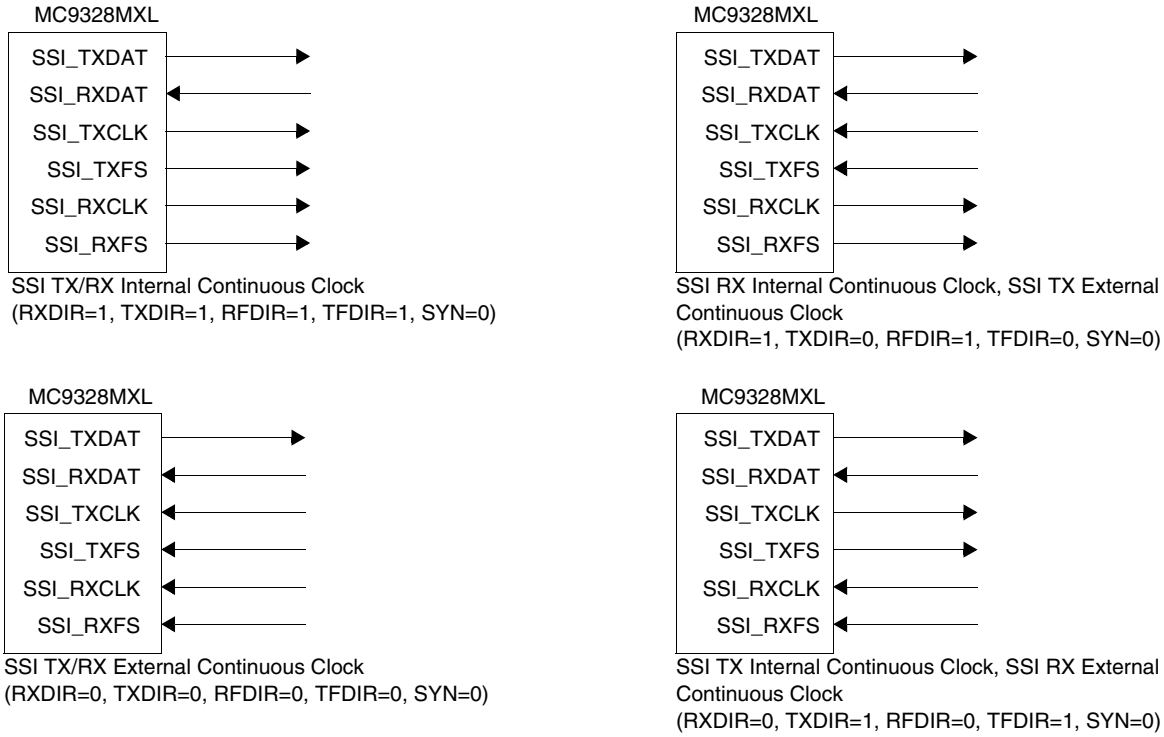


Figure 27-14. Asynchronous (SYN = 0) SSI Configurations—Continuous Clock

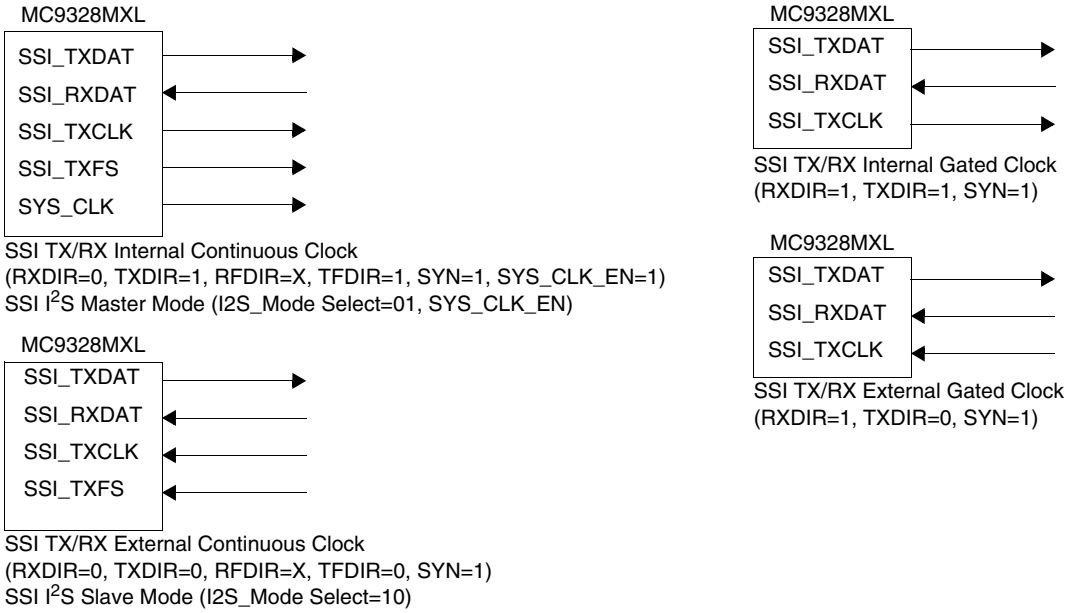


Figure 27-15. Synchronous SSI Configurations—Continuous and Gated Clock

An example of the pin signals for an 8-bit data transfer is shown in Figure 27-16. Continuous and gated clock signals are shown, as well as the bit-length frame sync signal and the word-length frame sync signal.

NOTE:

The shift direction can be defined as MSB first or LSB first.

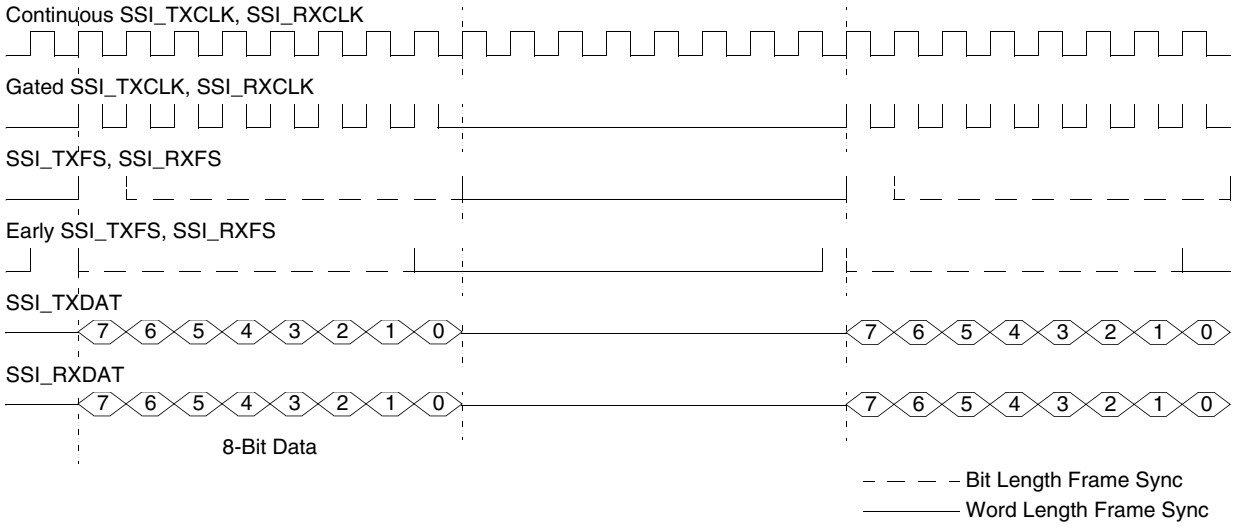


Figure 27-16. Serial Clock and Frame Sync Timing

27.5 SSI Operating Modes

The SSI module has three basic operating modes, with the option of asynchronous or synchronous protocol for most modes.

- Normal mode
  - asynchronous protocol
  - synchronous protocol
- Network mode
  - asynchronous protocol
  - synchronous protocol
- Gated clock mode
  - synchronous protocol only

The mode of the SSI is determined by several bits in the SSI control registers. Table 27-23 lists these operating modes and gives examples of applications that typically use each mode.

Table 27-23. SSI Operating Modes

TX, RX Sections	Serial Clock	Mode	Typical Application
Asynchronous	Continuous	Normal	Multiple synchronous codecs
Asynchronous	Continuous	Network	TDM codec or DSP networks
Synchronous	Continuous	Normal	Multiple synchronous codecs
Synchronous	Continuous	Network	TDM codec or DSP network
Synchronous	Gated	Normal	SPI-type devices; DSP to MCU

The transmit and receive sections of the SSI can be synchronous or asynchronous. In synchronous mode, the transmitter and the receiver use a common clock and frame synchronization signal. In asynchronous mode, the transmitter and receiver each have their own clock and frame synchronization signals. Continuous or Gated clock mode can be selected. In continuous mode, the clock runs continuously. In gated clock mode, the clock is only functioning during transmission.

Normal or network mode also can be selected. In normal mode, the SSI functions with one data word of I/O per frame. In network mode, a frame can contain between 2 and 32 data words. Network mode is typically used in star or ring-time division multiplex networks with other processors or codecs, allowing interface to time division multiplexed networks without additional logic. Use of the gated clock is not allowed in network mode. These distinctions result in the basic operating modes that allow the SSI to communicate with a wide variety of devices.

The SSI supports both normal and network modes, and these can be selected regardless of whether the transmitter and receiver are synchronous or asynchronous. Typically these protocols are used in a periodic manner, when data is transferred at regular intervals, such as at the sampling rate of an external codec. Both modes use the concept of a frame. The beginning of the frame is marked with a frame sync when programmed with continuous clock. The frame sync occurs at a periodic interval. The length of the frame is determined by the DC field in either the SRCCR or STCCR register, depending on whether data is being transferred or received. The number of words transferred per frame depends on the mode of the SSI.

In normal mode, one data word is transferred per frame. In network mode, the frame is divided into anywhere between 2 and 32 time slots, when one data word can optionally be transferred in each time slot.

27.5.1 Normal Mode

Normal mode is the simplest mode of the SSI. It transfers one word per frame. In continuous clock mode, a frame sync occurs at the beginning of each frame. The length of the frame is determined by the fields of the STCCR or SRCCR registers:

- The period of the serial bit clock. For external clocking, this is based on the frequency on the SSI_TXCLK pin. For an internal clock, this is determined by the values of the PSR bit and the PM field.
- The number of bits per sample (WL)
- The number of time slots per frame (DC)

When normal mode is configured to provide more than one time slot per frame, data is transmitted only in the first time slot. No data is transmitted in subsequent time slots.

27.5.1.1 Normal Mode Transmit

The conditions for data transmission from the SSI in normal mode are:

1. SSI enabled (SCSR:SSI_EN = 1)
2. Enable the FIFO (STCR:TFEN = 1) and configure the transmit (SFCSR:TFWM) and receive (SFCSR:RFWM) Water Marks if the FIFO is used.
3. Write data to the STX register
4. Enable transmitter (SCSR:TE = 1)
5. Active frame sync (required for continuous clock)
6. Active bit clock (the bit clock is active when the transmitter is enabled; for a gated clock, see Table 27-14)

When these conditions occur in normal mode, the next data word is transferred into the TXSR. When the transmit FIFO is enabled, the data word is transferred from the transmit FIFO. When the transmit FIFO is disabled, the data word is transferred from the STX register.

The new data word is transmitted immediately and the TDE bit is set.

When the transmit FIFO is disabled and the TIE bit in the STCR is set, the transmit interrupt occurs. When the transmit FIFO is enabled, the FIFO waits until the Transmit Water Mark level has been reached. The transmit interrupt then occurs if the TIE bit is set. In this case, an eighth data word is transferred and shifted out before the MC9328MXL writes new data to the STX register.

The SSI_TXDAT pin is disabled except during the data transmission period. For a continuous clock, the optional frame sync output and clock outputs are not tri-stated, even when both receiver and transmitter are disabled.

27.5.1.2 Normal Mode Receive

The conditions for data reception from the SSI are:

1. SSI enabled (SCSR:SSI_EN = 1)
2. Receiver enabled (SCSR:RE = 1)
3. Active frame sync (required for continuous clock)

4. Active bit clock (the bit clock is active when the receiver is enabled; for a gated clock, see Table 27-14)

With these conditions are met in normal mode with a continuous clock, each time the frame sync signal is generated (or detected), a data word is clocked in. Also given these conditions and a gated clock, each time the clock begins, a data word is clocked in.

When the receive FIFO is not enabled, after receiving a data word, the data is transferred from the RXSR to the SRX register. The RDR bit is set, and the receive interrupt occurs if it is enabled (the RIE bit is set).

When the receive FIFO is enabled, after receiving the data word, it is transferred to the receive FIFO. The RFF bit is set when the SRX register is full and receive FIFO register reaches the select threshold (Receive Water Mark), and the receive interrupt occurs if the RIE bit is set.

The MC9328MXL program must read the data from the SRX register before a new data word is transferred from the RXSR, otherwise the ROE bit is set. When the receive FIFO is enabled, the ROE bit is set when the SRX register and the receive FIFO are full of data and a new data word is ready to be transferred to the receive FIFO. Figure 27-17 shows transmitter and receiver timing for an 8-bit word with two words per time slot in normal mode, with a continuous clock and a late word length frame sync.

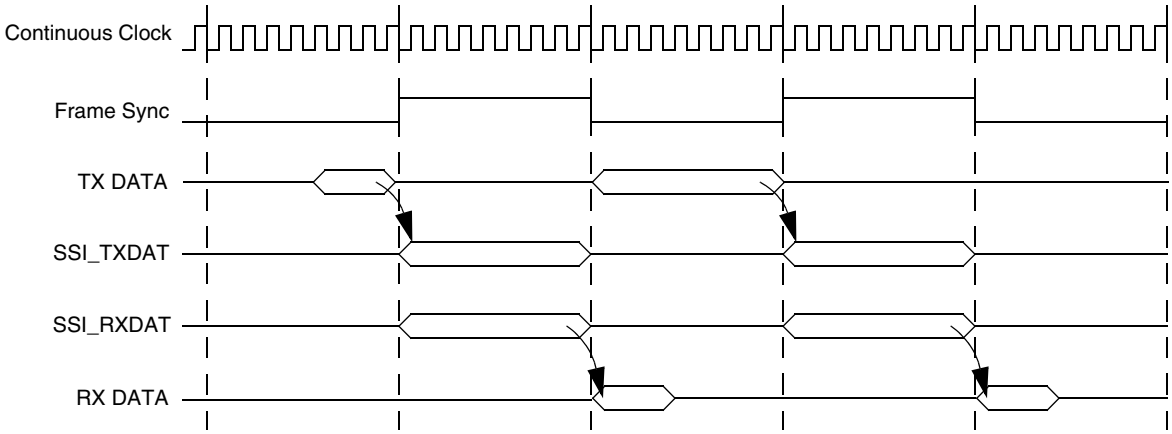


Figure 27-17. Normal Mode Timing—Continuous Clock

Figure 27-18 shows a similar case for a gated clock. A pull-down resistor is required in the gated clock example because the clock pin is tri-stated between transmissions.

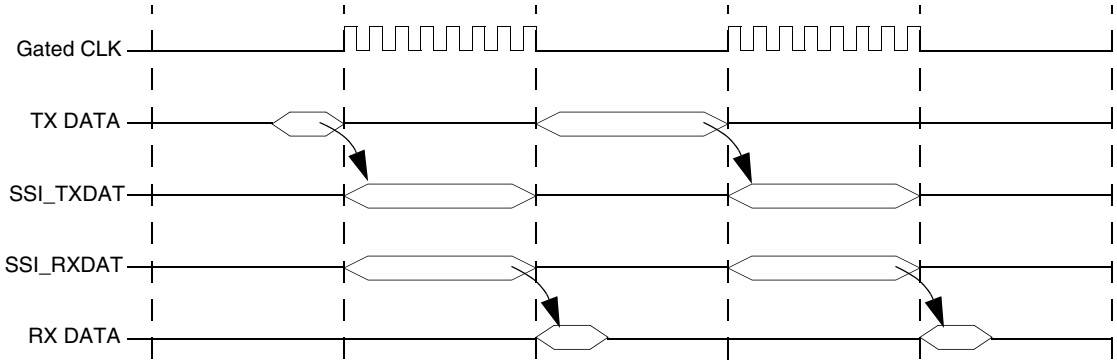


Figure 27-18. Normal Mode Timing—Gated Clock

27.5.2 Network Mode

Network mode is used for creating a Time Division Multiplexed (TDM) network, such as a TDM codec network or a network of DSPs. In continuous clock mode, a frame sync occurs at the beginning of each frame. In this mode, the frame is divided into more than one time slot. During each time slot, one data word can be transferred. Each time slot is then assigned to an appropriate codec or DSP on the network. The DSP can be a master device that controls its own private network, or a slave device that is connected to an existing TDM network and occupies a few time slots.

The frame sync signal indicates the beginning of a new data frame. Each data frame is divided into time slots and transmission and/or reception of one data word can occur in each time slot (rather than in just the frame sync time slot as in normal mode). The frame rate divider, controlled by the DC bits of the STCCR or SRCCR, selects 2 to 32 time slots per frame.

The length of the frame is determined by the fields of the STCCR or SRCCR registers:

- The period of the serial bit clock
  - for external clocking, this is based on the frequency on the SSI_TXCLK pin
  - for internal clocking, this is based on the values of the PSR bit and the PM field
- The number of bits per sample (WL)
- The number of time slots per frame (DC)

In network mode, data can be transmitted in any time slot. The distinction of the network mode is that each time slot is identified with respect to the frame sync (data word time). This time slot identification allows the option of transmitting data during the time slot by writing to the STX register or ignoring the time slot by writing to SSI Time Slot Register (STSR). The receiver is treated in the same manner, except that data is always being shifted into the RXSR and transferred to the SRX register. The MC9328MXL reads the SRX register and either uses it or discards it.

27.5.2.1 Network Mode Transmit

The transmit portion of the SSI is enabled when the SSI_EN and the TE bits in the SCSR are both set. However, for a continuous clock, when the TE bit is set, the transmitter is enabled only after detection of a new time slot (if the TE bit is set during a slot other than the first). The software must find the start of the next frame.

The normal start-up sequence for transmission is as follows:

1. Write the data to be transmitted to the STX register. This clears the TDE bit.
2. Set the TE bit to enable the transmitter on the next word boundary (for continuous clock).
3. Enable transmit interrupts.

If the programmer decides not to transmit in a time slot by writing to the SSI Time Slot Register (STSR), this clears the TDE bit, however the SSI_TXDAT pin remains disabled during the time slot. When the frame sync is detected or generated (continuous clock), the first enabled data word is transferred from the STX register to the TXSR and is shifted out (transmitted). When the STX register is empty, the TDE bit is set. When the TIE bit in the STCR is set, setting the TDE bit causes a transmitter interrupt to occur.

The software can poll the TDE bit or use interrupts to reload the STX register with new data for the next time slot, or can write to the STSR to prevent transmitting in the next time slot. Failing to write to the STX register (or the STSR) before the TXSR is finished shifting (empty) causes a transmitter underrun. This is detected by the TUE bit of the SCSR. When this happens, the SSI_TXDAT pin continuously sends the last transmitted data.

Clearing the TE bit disables the transmitter (and the SSI_TXDAT pin) after transmission of the current data word is complete. Setting the TE bit enables transmission of the next word. The TE bit is cleared after the TDE bit is set to ensure that all pending data is transmitted.

To summarize, in network mode, the transmitter can generate an interrupt during every enabled time slot or require the DSP program to poll the TDE bit. When the TDE bit is set, the MC9328MXL responds with one of the following actions:

- Write data to the STX register—enables transmission in the next time slot.
- Write to the SSI Time Slot Register (STSR)—disables transmission in the next time slot.
- Do nothing—causes a transmit underrun to occur at the beginning of the next time slot (the previous data is re-transmitted).

27.5.2.2 Network Mode Receive

The receiver portion of the SSI is enabled when both the SSI_EN and the RE bits in the SCSR are set. However, the receive enable takes place during that time slot only if the RE bit is enabled before the second to last bit of the word. When the RE bit is cleared, the receiver is disabled immediately. SSI is capable of finding the start of the next frame automatically. The word is received into the RXSR and then transferred to the SRX register when the whole word is received. The transfer to the SRX register sets the RDR bit. When the RIE bit is set, setting the RDR bit causes a receive interrupt to occur.

After the received word is transferred to the SRX register, the second data word (second time slot in the frame), begins shifting in immediately. The DSP program must read the data from the SRX register (this clears the RDR bit) before the second data word is completely received into the RXSR or a receive overrun error occurs. This is detected by the ROE bit of the SCSR.

To summarize, in network mode, the receiver can generate an interrupt during every enabled time slot or require the DSP program to poll the RDR bit. When the TDE bit is set, the MC9328MXL responds with one of the following actions:

- Read the SRX register and use the data.
- Read the SRX register and ignore the data.
- Do nothing—a receiver overrun occurs at the end of the current time slot.

NOTE:

For a continuous clock, the optional frame sync output and clock output signals are not affected even when the transmitter or receiver is disabled. The TE and RE bits do not disable the bit clock or the frame sync generation. The only way to disable the bit clock and the frame sync generation is to disable the SSI_EN bit in the SCSRregister.

Figure 27-19 on page 27-43 shows the transmitter and receiver timing for an 8-bit word with continuous clock, with the FIFO disabled, and with a frame size of three words per frame sync, operating in network mode.

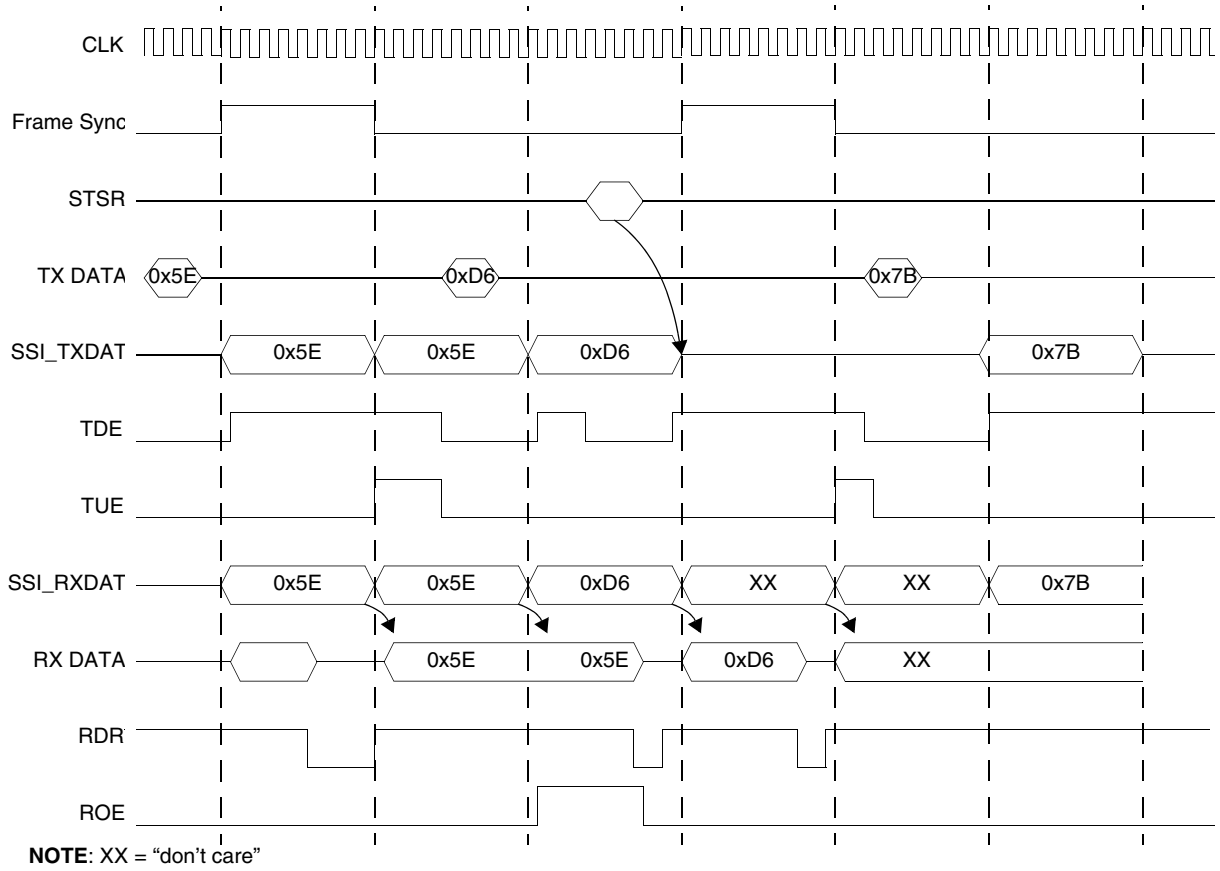


Figure 27-19. Network Mode Timing—Continuous Clock

27.6 Gated Clock Mode

The gated clock mode is often used to connect to SPI-type interfaces on Microcontroller Units (MCUs) or external peripheral chips. In gated clock mode, the presence of the clock indicates that valid data is on the SSI_TXDAT or SSI_RXDAT pins. For this reason, no frame sync is needed in this mode. When transmission of data is complete, the clock pin is tri-stated. Gated clocks are allowed for both the transmit and receive sections with either internal or external clock and in Normal mode. Gated clocks are not allowed in Network mode.

The clock runs when the TE bit or the RE bit in the SCSR are appropriately enabled. For clocks that are generated internally, all internal bit clocks, word clocks, and frame clocks continue to operate. When a valid time slot occurs, such as the first time slot in normal mode, the internal bit clock is enabled onto the appropriate clock pin. This allows data to be transferred out in periodic intervals in gated clock mode. When an external clock is used, the SSI waits for a clock signal to be received. When the clock begins, valid data is shifted in.

For gated clock operation in external clock mode, a proper clock signal must be applied to the SSI STCK for proper function. If the SSI uses a rising edge transition to clock data (TSCKP=0) and a falling edge transition to latch data(RSCKP=0), the clock must be in an active low state when idle. If the SSI uses a falling edge transition to clock data (TSCKP=1) and a rising edge transition to latch data (RSCKP=1), the clock must be in an active high state when idle. The following diagrams illustrate the different edge clocking and latching.

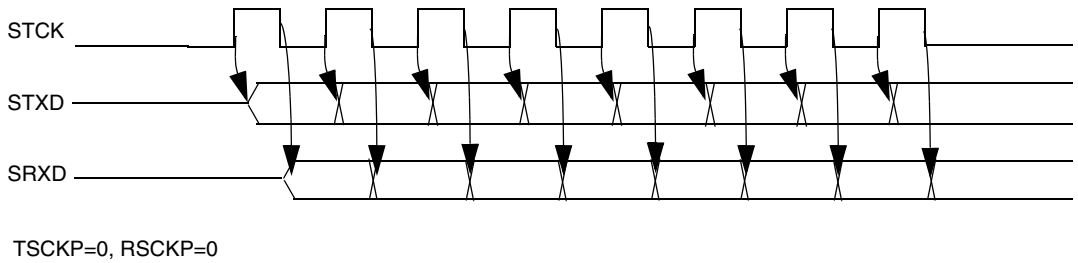


Figure 27-20. Rising Edge Clocking with Falling Edge Latching

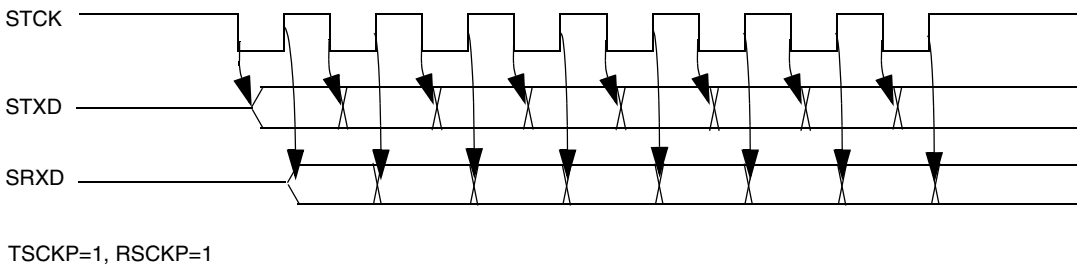


Figure 27-21. Falling Edge Clocking with Rising Edge Latching

For gated clock operation in internal clock mode, only the rising edge transition to clock data (TSCKP=0) and falling edge transition to latch data (RSCKP=0) is supported. The clock will always be in an active low state when idle. TSCKP=1 and RSCKP=1 are not supported in internal clock mode.

NOTE:

The bit clock pins must be kept free of timing glitches. A single glitch causes all subsequent transfers to lose synchronization.

When there is new data to be transmitted after the idle state has been entered, the data written to the Transmit FIFO will be transmitted immediately.

27.7 External Frame and Clock Operation

When applying external frame sync and clock signals to SSI, TE should not be set to 1 within 4 clock cycles before the rising edge of the frame sync signal and 4 clock cycles after the rising edge of the frame sync signal. The transition of SSI_TXFS or SSI_RXFS is synchronized with the rising edge of the external clock signal, SSI_TXCLK or SSI_RXCLK.

27.8 SSI Reset and Initialization Procedure

The SSI is affected by three types of reset:

- Power-on reset—The power-on reset is generated by asserting either the RESET pin or the Computer Operating Properly (COP) timer reset. The power-on reset clears the SSI_EN bit in the SCSR and disables the SSI.

- SSI reset—The SSI reset is generated when the SSI_EN bit in the SCSR is cleared. The SSI status bits are reset to the same state produced by the power-on reset. The SSI control bits are unaffected. The control bits in the top half of the SCSR are also unaffected. The SSI reset is useful for selective reset of the SSI without changing the present SSI control bits and without affecting the other peripherals.

The correct sequence to initialize the SSI is as follows:

1. Issue a power-on or SSI reset.
2. Set the SSI_EN bit in the SCSR.
3. Set all other control bits (such as TXDIR and RXDIR).
4. Set the TE and RE bits in the SCSR.

To ensure proper operation of the SSI, use the power-on or SSI reset before changing any of the control bits listed in Table 27-24. These control bits should not be changed during SSI operation.

Table 27-24. SSI Control Bits Requiring Reset Before Change

Control Register	Bit
SRCCR and STCCR	[14:13] = WL [1:0]
STCR and SRCR	[9] = TDMAE / RDMAE [6] = TFDIR / RFDIR [5] = TXDIR / RXDIR [4] = TSHFD / RSHFD [3] = TSCKP / RSCKP [2] = TFSI / RFSI [1] = TFSL / RFSL [0] = TEFS / REFS
SCSR	[15] = SYS_CLK_EN [14:13] = I2S_MODE [12] = SYN [11] = NET

NOTE:

The SSI bit clock must go low for at least one complete period to ensure proper SSI reset.

Chapter 28
CMOS Sensor Interface Module

28.1 Introduction

This chapter describes the CMOS Sensor Interface (CSI) module, and discusses the architecture, the programming model, and the software initialization sequence. The CSI module is a logic interface that enables the MC9328MXL to connect directly to external CMOS image sensors.

The features of the CSI module include:

- Configurable interface logic to support Motorola and other commonly available CMOS sensors
- 8-bit data port for YCC, YUV, or Bayer-RGB data input
- 32 × 32 FIFO for storing image data that supports ARM920T processor data reads and DMA data burst transfers to system memory
- Single interrupt source to interrupt controller from maskable sensor interrupt sources: start of frame, FIFO full, and FIFO overrun
- Configurable master clock frequency output to sensor
- Statistic data generation for auto exposure (AE) and auto white balance (AWB) control of the camera

28.2 CSI Module Architecture

The CSI module consists of a control register to configure the interface timing, a control register for statistic data generation, a status register, interface logic, a 32 × 32 image data receive FIFO, and a 16 × 32 statistic data FIFO. Figure 28-1 on page 28-2 shows a block diagram of the CSI module.

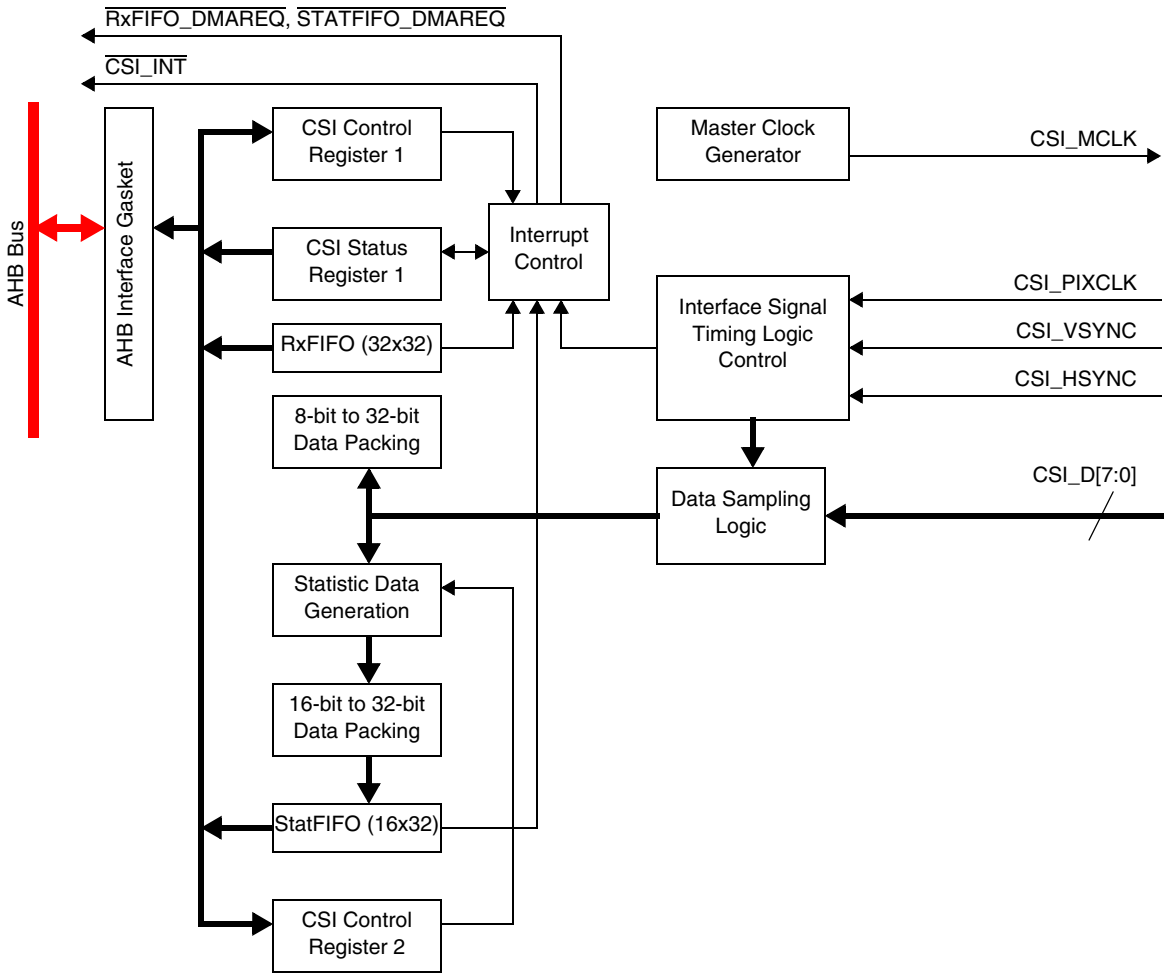


Figure 28-1. CSI Module Block Diagram

28.3 CSI Module Interface Signal Description

Table 28-1. CSI Module Interface Signal Description

Signal	Description
AHB Bus	ARM920T processor AHB bus.
$\overline{\text{CSI_INT}}$	A CSI module interrupt signal is sent to the interrupt controller. $\overline{\text{CSI_INT}}$ is asserted when any CSI interrupt event is detected and is negated when the all interrupt sources are cleared or masked.
$\overline{\text{RxFIFO_DMAREQ}}$	This DMA request signal is asserted when RxFIFO Full condition is true and is negated when RxFIFO Full condition becomes false.
$\overline{\text{STATFIFO_DMAREQ}}$	This DMA request signal is asserted when STATFIFO Full condition is true and is negated when STATFIFO Full condition becomes false.

Table 28-2. CMOS Interface Signals

CSI Signals	Direction	Description
CSI_VSYNC	IN	Vertical sync or start of frame
CSI_HSYNC	IN	Horizontal sync or horizontal window reference
CSI_D [7:0]	IN	Sensor data (Bayer or YUV)
CSI_MCLK	OUT	Sensor master clock
CSI_PIXCLK	IN	Data latch clock

28.3.1 Pin Configuration for CSI

Table 28-3 lists the pins used for the CSI module. These pins are multiplexed with other functions on the device, and must be configured for CSI operation.

NOTE:

The user must ensure that the data direction bits in the GPIO are set to the correct direction for proper operation. See Section 29.5.1, “Data Direction Registers,” on page 29-8 for details.

Table 28-3. Pin Configuration

Pin	Setting	Configuration Procedure
CSI_VSYNC	Primary function of GPIO Port A [12]	1. Clear bit 12 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 12 of Port A General Purpose Register (GPR_A)
CSI_HSYNC	Primary function of GPIO Port A [13]	1. Clear bit 13 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 13 of Port A General Purpose Register (GPR_A)
CSI_D [7:0]	Primary function of GPIO Port A [11:4]	1. Clear bits [11:4] of Port A GPIO In Use Register (GIUS_A) 2. Clear bits [11:4] of Port A General Purpose Register (GPR_A)
CSI_MCLK	Primary function of GPIO Port A [3]	1. Clear bit 3 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 3 of Port A General Purpose Register (GPR_A)
CSI_PIXCLK	Primary function of GPIO Port A [14]	1. Clear bit 14 of Port A GPIO In Use Register (GIUS_A) 2. Clear bit 14 of Port A General Purpose Register (GPR_A)

28.4 CSI Module Operation

The following sections describe the operation of the CSI module’s data FIFO and interrupts.

28.4.1 Data FIFO Operation

The CSI module features an integrated 32 × 32-bit FIFO for the image data and a 16 × 32-bit FIFO for statistic data. FIFO access is achieved by reading the CSI RxFIFO Register 1 and CSI Statistic FIFO Register 1.

The RxFIFO full level is programmed by the RxFF_LEVEL bits in CSI Control Register 1. When the 32-word RxFIFO is full, any further data writes to the RxFIFO by the external CMOS sensors are ignored and the RxFIFO overrun interrupt (when enabled) is generated. The RxFIFO and STATFIFO operate identically.

28.4.2 CSI Interrupt Operation

The CSI provides a single interrupt to the interrupt controller. For interrupt pin assignments, see Chapter 10, “Interrupt Controller (AITC).”

28.4.3 Register Access While CSI Module is Disabled

When the value of EN bit of CSI Control Register 1 is 0, nothing can be written into any CSI module register. Setting the EN bit to 0 resets all the CSI module registers.

To initialize the CSI module registers for operation, perform the following steps:

- Set the EN bit of CSI Control Register 1 to 1.
- Write values to other registers or other bits of CSI Control Register 1.
- Complete remaining user-specified procedures.

28.5 Programming Model

The CSI module includes five user-accessible 32-bit registers. Table 28-4 summarizes these registers and their addresses.

Table 28-4. CSI Module Register Memory Map

Description	Name	Address
CSI Control Register 1	CSICR1	0x00224000
CSI Control Register 2	CSICR2	0x00224004
CSI Status Register 1	CSISR	0x00224008
CSI Statistic FIFO Register 1	CSISTATR	0x0022400C
CSI RxFIFO Register 1	CSIRXR	0x00224010

28.5.1 CSI Control Register 1

The CSI Control Register 1 provides user control of the CSI module's signals timing and interrupts.

CSICR1 CSI Control Register 1 **Addr 0x00224000**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
							SF_OR_INTEN	RF_OR_INTEN	STATFF_LEVEL		STATFF_INTEN	RXFF_LEVEL		RXFF_INTEN	SOF_POL	SOF_INTEN
TYPE	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	r/w1c
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MCLKDIV						MCLKEN	FCC	BIG_ENDIAN	CLR_STATFIFO	CLR_RxFIFO	GCLK_MODE	INV_DATA	INV_PCLK	REDGE	EN
TYPE	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 28-5. CSI Control Register 1 Description

Name	Description	Settings
Reserved Bits 31–26	Reserved—These bits are reserved and should read 0.	
SF_OR_INTEN Bit 25	**STATFIFO Overrun Interrupt Enable**—Enables/Disables the STATFIFO overrun interrupt.	0 = Disable STATFIFO overrun interrupt 1 = Enable STATFIFO overrun interrupt
RF_OR_INTEN Bit 24	**RxFIFO Overrun Interrupt Enable**—Enables/Disables the RxFIFO overrun interrupt.	0 = Disable RxFIFO overrun interrupt 1 = Enable RxFIFO overrun interrupt
STATFF_LEVEL Bits 23–22	**STATFIFO Full Level**—Defines the STATFIFO full level. When the amount of data in the STATFIFO matches the defined STATFIFO full level, a FIFO full interrupt (when enabled) and a DMA request are generated.	00 = 4 words 01 = 8 words 10 = 12 words 11 = 16 words
STATFF_INTEN Bit 21	**STATFIFO Full Interrupt Enable**—Enables/Disables the STATFIFO full interrupt.	0 = Disable the STATFIFO full interrupt 1 = Enable the STATFIFO full interrupt
RXFF_LEVEL Bits 20–19	**RxFIFO Full Level**—Defines the RxFIFO full level. When the amount of data in the RxFIFO matches the defined RxFIFO full level, a FIFO full interrupt (when enabled) and a DMA request are generated.	00 = 4 words 01 = 8 words 10 = 16 words 11 = 24 words
RXFF_INTEN Bit 18	**RxFIFO Full Interrupt Enable**—Enables/Disables the RxFIFO full interrupt.	0 = Disable the RxFIFO full interrupt 1 = Enable the RxFIFO full interrupt

Table 28-5. CSI Control Register 1 Description (continued)

Name	Description	Settings
SOF_POL Bit 17	**Start Of Frame Interrupt Polarity**—Selects the edge (rising or falling) that generates the SOF interrupt.	0 = SOF interrupt generated on SOF falling edge 1 = SOF interrupt generated on SOF rising edge
SOF_INTEN Bit 16	**Start Of Frame Interrupt**—SOF interrupt status bit; set when interrupt occurs.	*Read:* 0 = No interrupt 1 = SOF interrupt occurred *Write:* 0 = No action 1 = Clears bit
MCLKDIV Bits 15–12	**MCLK Divider**—Contains the integer divider value applied to the SYSCLK input to generate the MCLK signal for the external CMOS sensors.	0000 = Divide by 2 0001 = Divide by 4 0010 = Divide by 6 … 1111 = Divide by 32
Reserved Bits 11–10	Reserved—These bits are reserved and should read 0.	
MCLKEN Bit 9	**MCLK Enable**—Enables/Disables the MCLK signal output to the external device.	0 = MCLK is not output 1 = MCLK is output
FCC Bit 8	**FIFO Clear Control**—Controls how RXFIFO and STATFIFO are cleared and STAT block is reset. Asynchronous clear: the RXFIFO is cleared when both of the following conditions are met: • A 1 is written to the CLR_RXFIFO, while the STATFIFO is cleared • The STAT block is held reset until the next SOF when a 1 is written to the CLR_STATFIFO bit. The RXFIFO is available to receive data immediately after clear. Synchronous clear: RXFIFO and STATFIFO are cleared, and STAT block is reset in 1 internal clock period when the rising edge of SOF is detected. Both the FIFOs and the STAT block are available for operation immediately.	0 = Performs an asynchronous clear 1 = Performs a synchronous clear
BIG_ENDIAN Bit 7	**Big Endian Enable**—Configures the packing of data into the RxFIFO in big endian (1) or little endian (0) format.	0 = RxFIFO data in little endian format 1 = RxFIFO data in big endian format
CLR_STATFIFO Bit 6	**Clear STATFIFO**—This bit is ignored when FCC is set. The STATFIFO is cleared when a 1 is written to this bit and when FCC is reset. This bit is auto-reset after STATFIFO is clear.	0 = No effect 1 = Clear STATFIFO
CLR_RXFIFO Bit 5	**Clear RxFIFO**—This bit is ignored when FCC is set. The RXFIFO is cleared when a 1 is written to this bit and when FCC is reset. This bit is auto-reset after RXFIFO is clear.	0 = No effect 1 = Clear STATFIFO

Table 28-5. CSI Control Register 1 Description (continued)

Name	Description	Settings
GCLK_MODE Bit 4	**Gated Clock Mode**—Controls how data is latched. When set, it latches data on selected pixel clock edge and when CSI_HSYNC is logic high. When reset, it latches data on selected pixel clock edge regardless of CSI_HSYNC.	0 = Latches data on the selected pixel clock edge 1 = Latches data on selected pixel clock edge and when CSI_HSYNC is logic high.)
INV_DATA Bit 3	**Invert Data Input**—Inverts the CSI_D [7:0] data lines before they are applied to the interface circuitry.	0 = No effect 1 = Inverts CSI_D [7:0] data lines
INV_PCLK Bit 2	**Invert PIXCLK Input**—Inverts the CSI_PIXCLK signal before they are applied to the interface circuitry.	0 = No effect 1 = Inverts CSI_PIXCLK signal
REDGE Bit 1	**Rising Edge**—Controls the edge of CSI_PIXCLK that latches data.	0 = Latches data at the falling edge of CSI_PIXCLK 1 = Latches data at the rising edge of CSI_PIXCLK
EN Bit 0	**Enable**—Enables/Disables the CMOS Sensor Interface module.	0 = Disable CSI module 1 = Enable CSI module

28.5.2 CSI Control Register 2

The CSI Control Register 2 indicates to the statistic block the live view resolution, the pixel of the Bayer pattern where the sensor starts, and the horizontal and vertical count that determines the number of pixels to skip between the 64 × 64 blocks of statistic when generating statistics on live view images that are larger than 512 × 384 pixels.

CSICR2 CSI Control Register 2 **Addr 0x00224004**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
						DRM	AFS		SCE			BTS		LVRM		
TYPE	r	r	r	r	r	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	VSC								HSC							
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 28-6. CSI Control Register 2 Description

Name	Description	Settings
Reserved Bits 31–27	Reserved—These bits are reserved and should read 0.	
DRM Bit 26	**Double Resolution Mode**—Indicates whether an array of 8 x 6 values or 8 x 12 values is produced for the statistic data. See Table 28-11 on page 28-14.	0 = Statistic grid of 8 x 6 1 = Statistic grid of 8 x 12
AFS Bits 25–24	**Auto Focus Spread**—Indicates how a group of 4 or 8 green pixels is paired up for computation of absolute difference for the auto focus statistic data.	00 = Compute absolute difference on consecutive green pixels 01 = Compute absolute difference on every third green pixels 1x = Compute absolute difference on every fifth green pixels
SCE Bit 23	**Skip Count Enable**—Enables/Disables skip counting. To enable the StatValidMask Block (refer to Figure 28-2 on page 28-13), the HSC and VSC are used for counting the number of pixels to be skipped.	0 = Skip count disable 1 = Skip count enable
Reserved Bits 22–21	Reserved—These bits are reserved and should read 0.	
BTS Bits 20–19	**Bayer Tile Start**—Indicates the pattern of the Bayer data (starting with green, followed by red and then for the next line, the pattern is blue followed by green).	00 = GR next line is BG 01 = RG next line is GB 10 = BG next line is GR 11 = GB next line is RG

Table 28-6. CSI Control Register 2 Description (continued)

Name	Description	Settings
LVRM Bits 18–16	**Live View Resolution Mode**—Indicates the size of the image range (from 288 x 216 to 512 x 384 pixels) for statistic data generation. Use SCE with VSC and HSC for image sizes larger than 512 x 384 pixels.	000 = 512 x 384 001 = 448 x 336 010 = 384 x 288 011 = 384 x 256 100 = 320 x 240 101 = 288 x 216 110 = 400 x 300
VSC Bits 15–8	**Vertical Skip Count**—Indicates the number of rows to skip during the computation of the statistic data for image sizes larger than 512 x 384 pixels. See Figure 28-3 on page 28-14. SCE must be 1 or VSC is ignored.	0x00 = Skip 1 row 0x01 = Skip 2rows 0x02 = Skip 3row ... 0xFF = Skip 256 rows
HSC Bits 7–0	**Horizontal Skip Count**—Indicates the number of horizontal pixels to skip during the computation of the statistic data for image sizes larger than 512 x 384 pixels. See Figure 28-3 on page 28-14. SCE must be 1 or HSC is ignored.	0x00 = Skip 1 row 0x01 = Skip 2rows 0x02 = Skip 3row ... 0xFF = Skip 256 rows

28.5.3 CSI Status Register 1

The CSI Status Register 1 (CSISR) contains sensor interface status.

CSISR CSI Status Register 1 **Addr 0x00224008**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
							SFF_OR_INT	RFF_OR_INT			STATFF_INT			RXFF_INT		SOF_INT
TYPE	r	r	r	r	r	r	rw	rw	r	r	rw	r	r	rw	r	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
									0x0000							

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																DRDY
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
									0x0000							

Table 28-7. CSI Status Register 1 Description

Name	Description	Settings
Reserved Bits 31–26	Reserved—These bits are reserved and should read 0.	
SFF_OR_INT Bit 25	**STATFIFO Overrun Interrupt**—Indicates whether a STATFIFO overrun is detected. Clear by writing 1.	0 = STATFIFO overrun not detected 1 = STATFIFO overrun detected
RFF_OR_INT Bit 24	**RxFIFO Overrun Interrupt**—Indicates whether an RxFIFO overrun is detected. Clear by writing 1.	0 = RxFIFO overrun not detected 1 = RxFIFO overrun detected
Reserved Bits 23–22	Reserved—These bits are reserved and should read 0.	
STATFF_INT Bit 21	**STATFIFO Full Interrupt**—Indicates whether the STATFIFO is full. Cleared automatically after a read from the STATFIFO is performed.	0 = STATFIFO is not full 1 = STATFIFO is full
Reserved Bits 20–19	Reserved—These bits are reserved and should read 0.	
RXFF_INT Bit 18	**RxFIFO Full Interrupt**—Indicates whether the RxFIFO is full. Cleared automatically after a read from the RxFIFO is performed.	0 = RxFIFO is not full 1 = RxFIFO is full
Reserved Bit 17	Reserved—This bit is reserved and should read 0.	

Table 28-7. CSI Status Register 1 Description (continued)

Name	Description	Settings
SOF_INT Bit 16	**Start Of Frame Interrupt**—Enables/Disables the SOF interrupt. When set, SOF interrupt is sampled. Clear by writing 1.	0 = Disable SOF interrupt 1 = Enable SOF interrupt
Reserved Bits 15–1	Reserved—These bits are reserved and should read 0.	
DRDY Bit 0	**Data Ready**—Indicates that there is data ready to be read out of the RxFIFO. When set, at least one data word is ready in RxFIFO.	0 = No data word is ready in the RxFIFO 1 = At least one data word is ready in the RxFIFO

28.5.4 CSI Statistic FIFO Register 1

The read-only CSI Statistic FIFO Register 1 contains statistic data. Writing to this register has no effect.

CSISTATR CSI Statistic FIFO Register 1 **Addr 0x0022400C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	STAT															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	STAT															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 28-8. CSI Statistic FIFO Register 1 Description

Name	Description
STAT Bits 31–0	Statistic Data—Contains the statistic data. For more details refer to Section 28.6.4, “Packing of Statistic Data,” on page 28-15.

28.5.5 CSI RxFIFO Register 1

The read-only CSI RxFIFO Register 1 contains received image data. Writing to this register has no effect.

Because the incoming image data is 8-bit and the FIFO size is 32-bit, the incoming data is packed into the 32-bit FIFO in the sequences provided in Table 28-9.

Table 28-9. CSI Module FIFO Register Storage Scheme

FIFO Bits	Image Data	Starting Addresses When DMA Transfer is Completed from FIFO to RAM
31:24	4th byte (LE) 1st byte (BE)	Stored in highest (Base address + 3) in little endian format Stored in lowest (Base) address in the big endian format
23:16	3rd byte (LE) 2nd byte (BE)	Stored in Base address + 2 in little endian format Stored in Base address + 1 in big endian format
15:8	2nd byte (LE) 3rd byte (BE)	Stored in Base address + 1 in little endian format Stored in Base address + 2 in big endian format
7:0	1st byte (LE) 4th byte (BE)	Stored in lowest (Base) address in little endian format Stored in highest (Base address + 3) in big endian format

When a DMA transfer is complete, regardless of big endian or little endian mode, the lowest address in the RAM always stores the first byte of the incoming data. Likewise, the highest address stores the last byte of the incoming image data.

CSIRXR CSI RxFIFO Register 1 **Addr 0x00224010**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	IMAGE															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	IMAGE															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 28-10. CSI RxFIFO Register 1 Description

Name	Description
IMAGE Bits 31–0	**Image Data**—Contains the image data from the CMOS sensor.

28.6 Statistic Data Generation

The statistic block generates a simple set of normalized totals for the red, blue, and green pixels contained in the image. The software uses these values to control the auto exposure (AE) and auto white balance (AWB) of the camera. The block also generates a sum of absolute differences of the green pixels for each block that is used to determine the relative focus of the image by creating this rough measure for contrast.

28.6.1 Statistic Block Diagram and Description

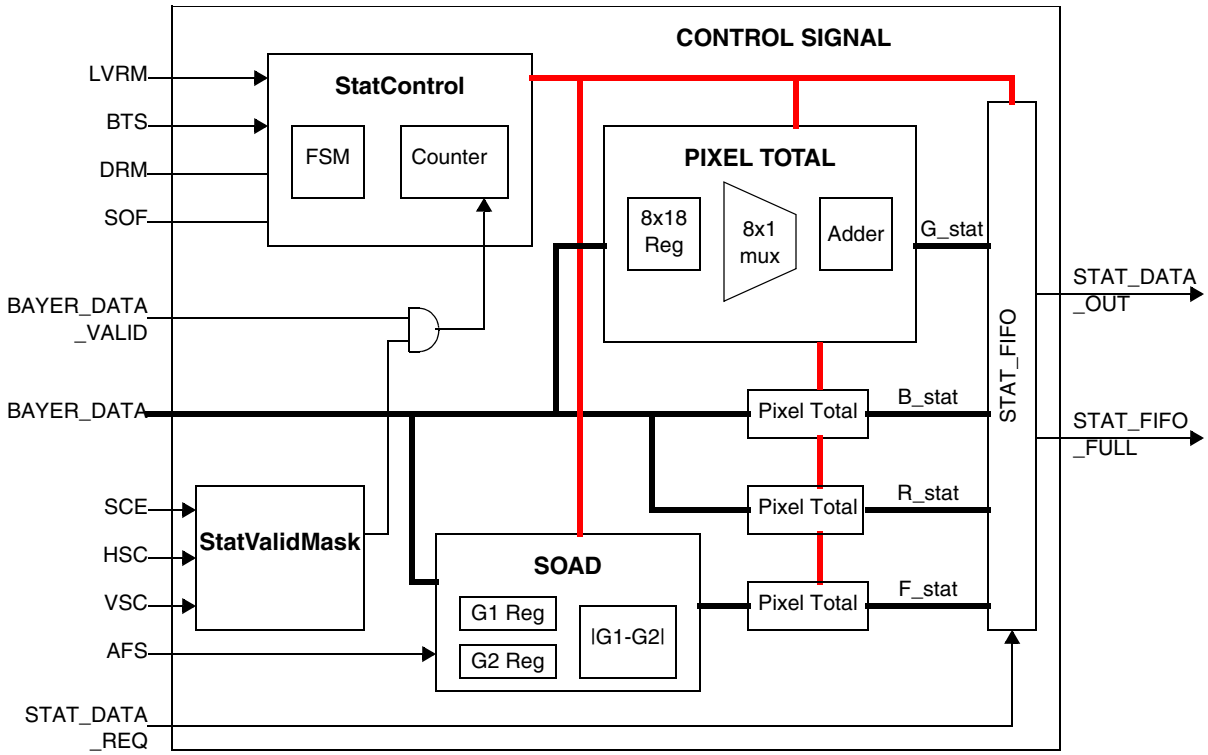


Figure 28-2. Statistic Block Diagram

28.6.2 Auto Exposure and Auto White Balance

The values are based on the supported live view image resolutions in the range of 512 × 384 to 288 × 216 pixels.

These image sizes are divided up into square blocks of varying size to produce an array of 8 × 6 values for each color in all cases. The sum of each color pixel values in each block is temporary saved in the 8 × 18 register. Depending on the DRM bit of the CSI Control Register 2, the sum value is either divided by 4 (DRM = 0) or 2 (DRM = 1) before it is stored in the FIFO. The SOAD (sum of absolute difference) data is generated by the SOAD block and temporarily stored in an 8 × 18 register. The DRM bit of the Control Register 2 determines the amount of division to be done on the SOAD data before it is stored in the FIFO. The block size for the corresponding live view LCD size is shown in Table 28-11 on page 28-14.

NOTE:

For the 384 × 256 live view, the statistic array is only 6 × 4 and uses a larger block size. This is due to the 3:2 ratio of the CCD vs. the 4:3 ratio of the rest of the CCDs.

Because there are twice as many green pixels as red or blue pixels, the value generated for the green is calculated by skipping every other line to produce results that are of the same magnitude as the blue and red pixels.

Table 28-11. Block Size for Live View LCD Size

Live View Resolution	Block Size	No. Of Each Color / Block	Statistic Array
288 x 216	36 x 36 (DRM = 0) 36 x 18 (DRM = 1)	324 (DRM = 0) 162 (DRM = 1)	8 x 6 (DRM = 0) 8 x 12 (DRM = 1)
320 x 240	40 x 40 (DRM= 0) 40 x 20 (DRM = 1)	400 (DRM = 0) 200 (DRM = 1)	8 x 6 (DRM = 0) 8 x 12 (DRM = 1)
384 x 256	64 x 64 (DRM = 0) 64 x 32 (DRM = 1)	1024 (DRM = 0) 512 (DRM = 1)	6 x 4 (DRM = 0) 6 x 8 (DRM = 1)
384 x 288	48 x 48 (DRM = 0) 48 x 24 (DRM = 1)	576 (DRM = 0) 288 (DRM = 1)	8 x 6 (DRM = 0) 8 x 12 (DRM = 1)
400 x 300	50 x 50 (DRM = 0) 50 x 25 (DRM = 1)	625 (DRM = 0) 313, 312 (DRM = 1)	8 x 6 (DRM = 0) 8 x 12 (DRM = 1)
448 x 336	56 x 56 (DRM = 0) 56 x 28 (DRM = 1)	784 (DRM = 0) 392 (DRM = 1)	8 x 6 (DRM = 0) 8 x 12 (DRM = 1)
512 x 384	64 x 64 (DRM = 0) 64 x 32 (DRM = 1)	1024 (DRM = 0) 512 (DRM = 1)	8 x 6 (DRM = 0) 8 x 12 (DRM = 1)

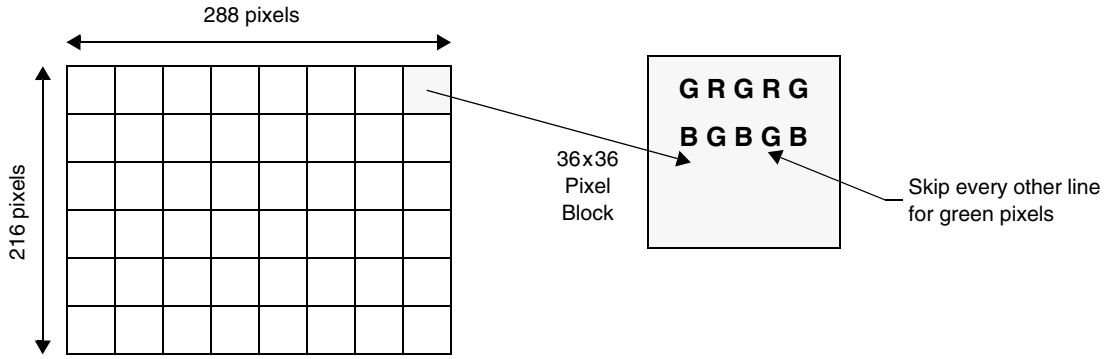


Figure 28-3. Statistic Blocks Example for 288 x 216 Pixels Image Size

For image sizes larger than 512 × 384 pixels, the SCE bit of the CSI Control Register 2 must be set to 1. The Vertical Skip Count and Horizontal Skip Count bits of the VSC and HSC bits of the CSI Control Register 2 contain the number of pixels to be skipped.

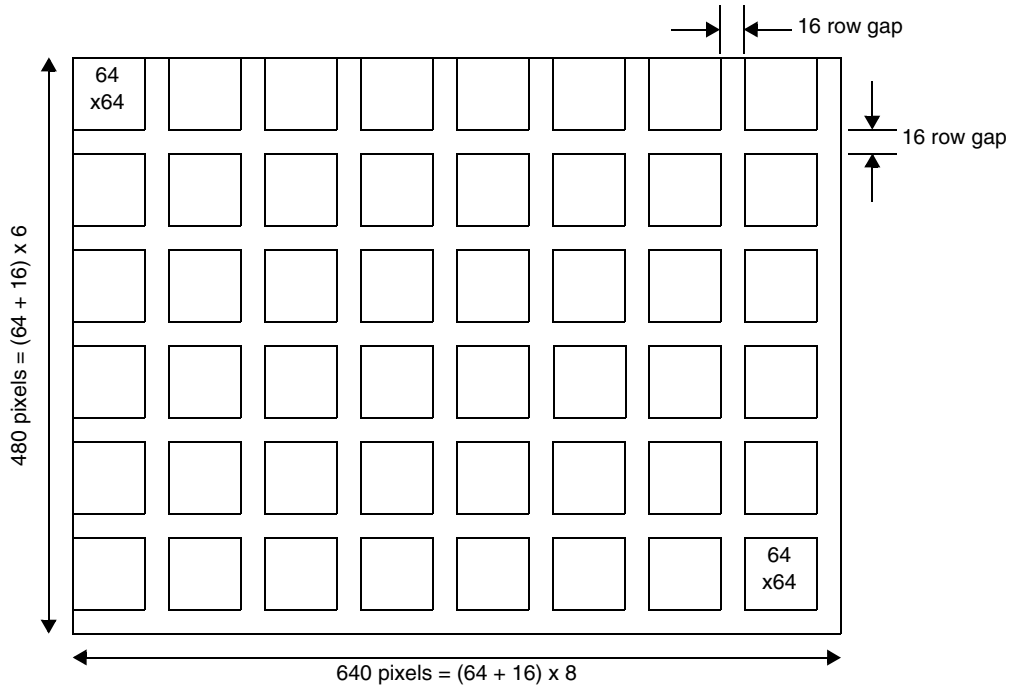


Figure 28-4. Full Resolution Statistic Example

For the example shown in Figure 28-4, the horizontal skip count is set to 15 pixels and the vertical skip count is set to 15 rows. This distributes the forty-eight (8 × 6) statistic blocks (64 × 64 pixels each) evenly over the full resolution image. The horizontal and vertical counts are not required to be identical, however for 4:3 ratio images, these counts are the same.

28.6.3 Auto Focus

The sum of absolute difference (SOAD) hardware runs parallel with the color summation hardware, however it only generates one value per statistic block based on green pixel values. The absolute difference is computed on each pair of green pixels in a row without any overlap. For the example frame in Figure 28-4 (218 × 216), there would be 72 absolute differences computed and summed per line. In the content of the 36 × 36 statistic block shown in Figure 28-3, there would be 9 absolute differences in each line of 36 pixels and 9 × 36 = 324 differences summed together for each 36 × 36 block of pixels.

28.6.4 Packing of Statistic Data

All of the statistic data, the red, green (sum of all the green pixels), blue, and the focus data, are in a 16-bit data format.

The data is packed into the 32 bits of the FIFO in the following sequence:

- (Red, Green, 1st block for big endian) or (Green, Red, 1st block for little endian) 32-bit data.
- (Blue, Focus, 1st block for big endian) or (Focus, Blue, 1st block for little endian) 32-bit data.
- (Red, Green, 2nd block for big endian) or (Green, Red, 2nd block for little endian) 32-bit data.
- (Blue, Focus, 2nd block for big endian) or (Focus, Blue, 2nd block for little endian) 32-bit data.

When a DMA operation is used to transfer the data to the RAM, the statistic data is placed in the RAM regardless if it is big endian (set the BIG_ENDIAN bit for correct data transfer) or little endian mode (default) as follows:

- **Address 0**—Red
- **Address 2**—Green
- **Address 4**—Blue
- **Address 6**— Focus

28.6.5 Sensor Interface Signals

There are two sensor interface signals used by the CSI module:

- **BAYER_DATA [7:0]**—These eight bits contain the incoming Bayer data.
- **BAYER_DATA_VALID**—This signal is used by the CMOS sensor valid bit.

28.6.6 Statistic Control Signals

The statistic control signals used by the CSI module are described in the following sections.

28.6.6.1 Start of Frame

The Start of Frame (SOF) signal is a pulse (from low to high and then a high to low) that triggers the statistic block operation.

28.6.6.2 Auto Focus Spread

For more information on Auto Focus Spread (AFS), see Section 28.5.2, “CSI Control Register 2,” on page 28-8.

Figure 28-5. Auto Focus Spread

28.6.7 Statistic Output and DMA Signals

28.6.7.1 Statistic Data Out

This is the STAT [31:0] field of CSI Statistic FIFO Register 1. The Statistic Data Out (STATS_DATA_OUT [31:0]) field contains the output data for the red, green, blue, and focus statistic. The sequence of the data is 32 bits for red (2 bytes) and green pixels (2 bytes) followed by another 4 bytes for blue and focus in the next cycle.

28.6.7.2 Statistic FIFO Full

The Statistic FIFO Full (STATS_FIFO_FULL) bit indicates that the CSI Statistic FIFO Register 1 is full and that a DMA request should be initiated.

28.6.7.3 Statistic Data Request

The Statistic Data Request (STATS_DATA_REQ) bit indicates a request to read the statistic data in CSI Statistic FIFO Register 1.

Chapter 29
GPIO Module and I/O Multiplexer (IOMUX)

29.1 General Description

This chapter describes the four GPIO ports of the MC9328MXL. All of the GPIO port pins are multiplexed with other signals.

NOTE:

See Chapter 2, "Signal Descriptions and Pin Assignments," for detailed I/O multiplexing information.

This section contains the description of the top level I/O multiplexing strategy in the MC9328MXL, which consists of two modules:

- Software controllable multiplexing performed in the GPIO module
- Hardware multiplexing performed in the IOMUX module

The I/O multiplexing strategy is designed to configure the inputs and outputs of the MC9328MXL to allow the same I/O pad to be used for peripheral functions and GPIO. The I/O multiplexing is designed to be as flexible as possible and to allow the simple and quick reuse of this system in future derivatives of the MC9328MXL. In addition to the I/O multiplexing, the IOMUX module contains the JTAG shift registers, which significantly simplify the I/O design.

There are four GPIO ports on the MC9328MXL: Port A, Port B, Port C, and Port D. Each port consists of 32 pins, however not all pins are used. The usable port pins are:

- **Port A—**pins 0–31
- **Port B—**pins 8–31
- **Port C—**pins 3–17
- **Port D—**pins 6–31

Figure 29-1 on page 29-2 depicts a top-level view of the IOMUX and GPIO modules for a single port pin. This circuitry is duplicated for each of the 97 port pins.

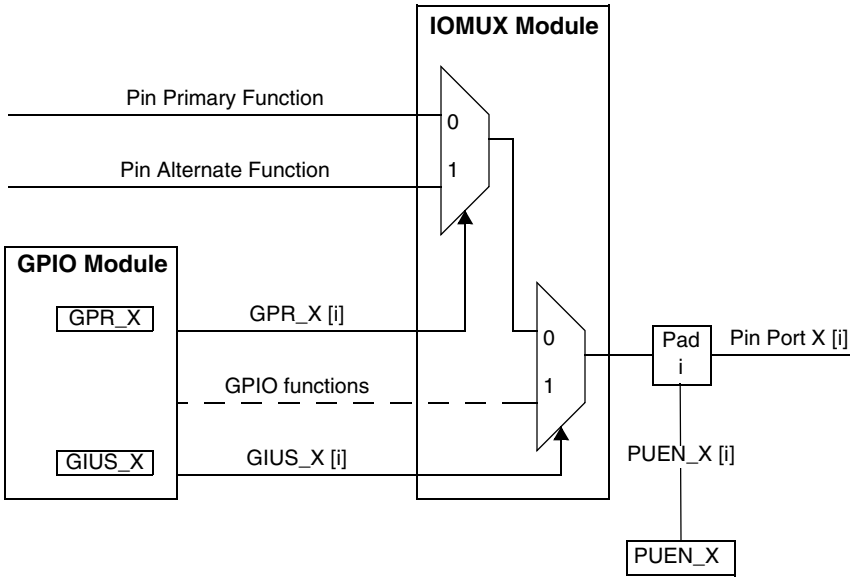


Figure 29-1. Top Level of Circuitry for Port X, Pin [i]

29.2 GPIO Module Overview

Most of this chapter focuses on the GPIO module, which provides general purpose I/O capability to the device. The GPIO module communicates with the ARM920T processor through an IPBUS interface connected as an IP peripheral. The complete module controls 97 bidirectional port pins. Software configurability allows each I/O to be configured as general purpose input (optionally routed to two different destinations) or general purpose output (from one out of four selectable sources).

29.2.1 GPIO Module Features

The GPIO module features include:

- 97 direction-configurable port pins
- Software controllable input/output selection through four 32-bit direction registers
- Software control for multiplexing one of four different sources (a data register and three peripheral modules on the MC9328MXL) for every output pin
- Software control for routing every input to other modules
- Input data sampling on each clock
- Software control of the IOMUX module through four 32-bit general purpose registers
- Configurability of each input port pin interrupt as positive edge triggered, negative edge triggered, positive level sensitive, negative level sensitive or as a masked interrupt
- Ability to logically OR each port's 32 interrupt lines to a single interrupt to the ARM920T processor
- Software reset

29.2.2 Interrupts

The interrupt block inside the GPIO module controls all GPIO interrupt signals. Inside this block, interrupts are defined as positive edge triggered, negative edge triggered, positive level sensitive, negative level sensitive, or masked. For edge triggered interrupts, the edge is detected by the GPIO interrupt block and is converted to a low level interrupt that is routed to other internal modules. When the interrupt is masked, a value of 1 is driven to the interrupt controller regardless of the GPIO input value. The interrupts waiting for service are stored in the Interrupt Status Register (ISR) for that port. Write a value of 1 to the ISR to clear the interrupt. The logical OR of all the non-masked interrupt lines is available as an output from the port. Individual interrupts for each port pin may also be output to other internal modules.

29.2.3 GPIO Signal Description

The circuitry for a single pin of the GPIO module is shown in Figure 29-2 on page 29-4. The signals shown in that figure and the signals shown in Figure 29-1 on page 29-2 are described in Table 29-1.

Table 29-1. GPIO External Pins Description

Signal Name	Direction	Description
AIN	Input	A 32-bit input from MC9328MXL. Any signal may be connected. This input may be reflected to GPIO output (GPIO-Out in Figure 29-2) by appropriate GPIO configuration.
BIN	Input	Same as AIN.
CIN	Input	Same as AIN.
AOUT	Output	A 32-bit output from the GPIO that connects external pins to internal signals in the MC9328MXL, sends an interrupt signal to an internal module, or is connected high or low.
BOUT	Output	Same as AOUT
GIUS_X	Output	Represents one of four GPIO In-Use Registers (GIUS). Connects to the IOMUX module and selects whether a pin is used for GPIO or a peripheral function.
GPR_X	Output	Represents one of the four General Purpose Registers (GPR). Connects to the IOMUX module and selects whether a pin is used for its primary or alternate function.
PUEN_X	Output	Selects whether the port pin is pulled up to a logic high.
$\overline{\text{GPIO_INT}}$	Output	The OR value of all 32 interrupt lines. Each pin [i] of the port corresponds to pin [i] of the ISR register.

29.3 GPIO Module Block Diagram

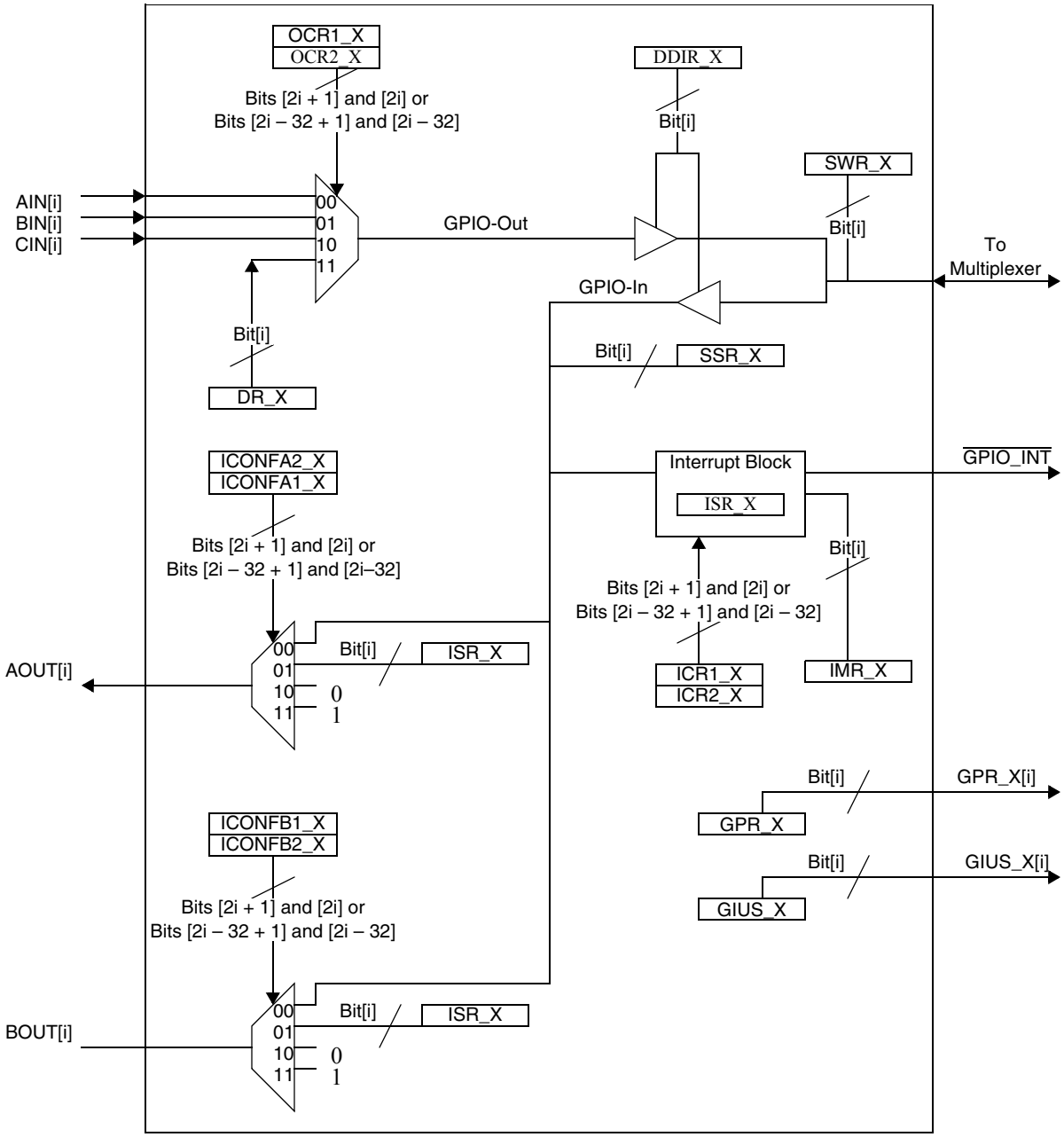


Figure 29-2. GPIO Module Block Diagram for Port X, Pin [i]

29.4 Pin Configuration for GPIO

The GPIO port pins on the MC9328MXL are multiplexed with signals from peripheral modules on the device. These pins must be configured for GPIO operation. Table 29-2 on page 29-5 shows the pin configuration.

To further expand the multiplexing features of the GPIO module, added functionality was incorporated to route the input and output signals of selected peripherals. These signals are routed to the GPIO module via AIN[i], BIN[i], CIN[i], AOUT[i], and BOUT[i], where “i” denotes any bit from bit 31 to bit 0. There are a set of these signals available for each GPIO Port (Port A, Port B, Port C, and Port D). The “IN” signals denote inputs to the GPIO module whereas the “OUT” signals are outputs from the GPIO module, as can be seen in Figure 29-2 on page 29-4. In the case of the “IN” signals, these signals are passed through the GPIO module from the selected peripheral and are output to the MC9328MXL pins. Therefore, the Data Direction Register for the desired “IN” signal must be set as an output. On the other hand, in the case of the “OUT” signals, these signals are also passed through the GPIO module, however their inputs come from the MC9328MXL pins and are output to the selected peripheral. Therefore, the Data Direction register for the desired “OUT” signal must be set as an input. For example, if the user wants to route the SPI2_RXD input via the PA1 pin, they must set GIUS_A bit 1 for GPIO function (set bit 1). The next step would then be to clear the ICONFA1_A bits 2 and 3 (to 00) to select the GPIO-IN to allow PA1 to drive AOUT[1]. Finally, the DDIR bit 1 must be set as an input (clear bit 1). Should the user need to route the SPI2_CLK output via pin PD7, they must first set bit 7 of GIUS_D. The next step is to clear bits 15 and 14 of OCR1_D to select AIN[7] as an input to the GPIO module. Finally, bit 7 of DDIR must be set to select this signal as an output from the GPIO module to the external pin. Table 29-3 shows the signals that use this expanded functionality.

Table 29-2. Pin Configuration

Pin	Configuration Procedure
Port A Pins 31–0	1. For each pin [i] that is used as a GPIO, set bit [i] in the Port A GPIO In Use Register (GIUS_A). 2. When pin [i] is used as an input: • Clear bit [i] in the Port A Data Direction Register (DDIR_A) • Read the Port A Sample Status Register (SSR_A) as needed 3. When pin [i] is used as an output: • Set bits [2i + 1] and [2i] in the Port A Output Configuration Register 1 (OCR1_A) or Set bits [2i – 32 + 1] and [2i – 32] in the Port A Output Configuration Register 2 (OCR2_A) • Write desired output value to bit [i] of the Port A Data Register (DR_A) • Set bit [i] in the Port A Data Direction Register (DDIR_A)
Port B Pins 31–8	1. For each pin [i] that is used as a GPIO, set bit [i] in thePort B GPIO In Use Register (GIUS_B). 2. When pin [i] is used as an input: • Clear bit [i] in the Port B Data Direction Register (DDIR_B) • Read the Port B Sample Status Register (SSR_B) as needed 3. When pin [i] is used as an output: • Set bits [2i + 1] and [2i] in the Port B Output Configuration Register 1 (OCR1_B) or Set bits [2i – 32 + 1] and [2i – 32] in the Port B Output Configuration Register 2 (OCR2_B) • Write desired output value to bit [i] of the Port B Data Register (DR_B) • Set bit [i] in the Port B Data Direction Register (DDIR_B)

Table 29-2. Pin Configuration (continued)

Pin	Configuration Procedure
Port C Pins 17–3	1. For each pin [i] that is used as a GPIO, set bit [i] in the Port C GPIO In Use Register (GIUS_C). 2. When pin [i] is used as an input: • Clear bit [i] in the Port C Data Direction Register (DDIR_C) • Read the Port C Sample Status Register (SSR_C) as needed 3. When pin [i] is used as an output: • Set bits [2i + 1] and [2i] in the Port C Output Configuration Register 1 (OCR1_C) or Set bits [2i – 32 + 1] and [2i – 32] in the Port C Output Configuration Register 2 (OCR2_C) • Write desired output value to bit [i] of the Port C Data Register (DR_C) • Set bit [i] in the Port C Data Direction Register (DDIR_C)
Port D Pins 31–6	1. For each pin [i] that is used as a GPIO, set bit [i] in the Port D GPIO In Use Register (GIUS_D). 2. When pin [i] is used as an input: • Clear bit [i] in the Port D Data Direction Register (DDIR_D) • Read the Port D Sample Status Register (SSR_D) as needed 3. When pin [i] is used as an output: • Set bits [2i + 1] and [2i] in the Port D Output Configuration Register 1 (OCR1_D) or Set bits [2i – 32 + 1] and [2i – 32] in the Port D Output Configuration Register 2 (OCR2_D) • Write desired output value to bit [i] of the Port D Data Register (DR_D) • Set bit [i] in the Port D Data Direction Register (DDIR_D)

29.5 Programming Model

There are four sets of control registers corresponding to the four GPIO ports. Each port has 17 registers associated with it, located at sequential addresses in the memory map. Table 29-3 summarizes these registers and their addresses.

The starting address of each register set is known as the *base address* and is referred to by the term *$BA*. The base addresses for the four GPIO ports are as follows:

- GPIO Port A $BA = 0x0021C000
- GPIO Port B $BA = 0x0021C100
- GPIO Port C $BA = 0x0021C200
- GPIO Port D $BA = 0x0021C300

Table 29-3. GPIO Module Register Memory Map

Description[1]	Name[1]	Address
Port X Data Direction Register	DDIR_X	$BA + $000
Port X Output Configuration Register 1	OCR1_X	$BA + $004
Port X Output Configuration Register 2	OCR2_X	$BA + $008
Port X Input Configuration Register A1	ICONFA1_X	$BA + $00C

Table 29-3. GPIO Module Register Memory Map (continued)

Description[1]	Name[1]	Address
Port X Input Configuration Register A2	ICONFA2_X	$BA + $010
Port X Input Configuration Register B1	ICONFB1_X	$BA + $014
Port X Input Configuration Register B2	ICONFB2_X	$BA + $018
Port X Data Register	DR_X	$BA + $01C
Port X GPIO In Use Register	GIUS_X	$BA + $020
Port X Sample Status Register	SSR_X	$BA + $024
Port X Interrupt Configuration Register 1	ICR1_X	$BA + $028
Port X Interrupt Configuration Register 2	ICR2_X	$BA + $02C
Port X Interrupt Mask Register	IMR_X	$BA + $030
Port X Interrupt Status Register	ISR_X	$BA + $034
Port X General Purpose Register	GPR_X	$BA + $038
Port X Software Reset Register	SWR_X	$BA + $03C
Port X Pull_Up Enable Register	PUEN_X	$BA + $040

1. X is a variable representing A, B, C or D. The actual register names include the port.

29.5.1 Data Direction Registers

The data direction registers specify whether each pin of the port is an input or an output pin.

There are four distinct Data Direction Registers; each holds the data for one of the four GPIO ports (Port A, Port B, Port C, and Port D).

		Addr
DDIR_A	Port A Data Direction Register	**0x0021C000**
DDIR_B	Port B Data Direction Register	**0x0021C100**
DDIR_C	Port C Data Direction Register	**0x0021C200**
DDIR_D	Port D Data Direction Register	**0x0021C300**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DDIR															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DDIR															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 29-4. Data Direction Registers Description

Name	Description	Settings
DDIR [i] Bits 31–0	**Data Direction**—Controls the direction of the pins.	0 = Pin [i] is an input 1 = Pin [i] is an output

29.5.2 Output Configuration Registers

The output configuration registers specify the output signal for each pin. There are two bits in the output configuration registers for each port pin.

There are two output configuration registers for each port; each holds the data for one of the four GPIO ports (Port A, Port B, Port C, and Port D).

29.5.2.1 Output Configuration Register 1

		Addr
OCR1_A	Port A Output Configuration Register 1	**0x0021C004**
OCR1_B	Port B Output Configuration Register 1	**0x0021C104**
OCR1_C	Port C Output Configuration Register 1	**0x0021C204**
OCR1_D	Port D Output Configuration Register 1	**0x0021C304**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	OCR1															
	pin 15		pin 14		pin 13		pin 12		pin 11		pin 10		pin 9		pin 8	
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	OCR1															
	pin 7		pin 6		pin 5		pin 4		pin 3		pin 2		pin 1		pin 0	
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 29-5. Output Configuration Register 1 Description

Name	Description	Settings		
OCR1 [i] Bits 31–0	**Output Configuration**—Corresponds to pins 0–15 of the port and defines which one of the four options is selected as the output signal GPIO-Out. Each port pin [i] (i = 0 through 15) requires two OCR1 bits to determine the output signal.	**OCR1 [2i + 1]**	**OCR1 [2i]**	**Output Selected**
		0	0	External input AIN [i]
		0	1	External input BIN [i]
		1	0	External input CIN [i]
		1	1	Data Register [i]

29.5.2.2 Output Configuration Register 2

		Addr
OCR2_A	Port A Output Configuration Register 2	**0x0021C008**
OCR2_B	Port B Output Configuration Register 2	**0x0021C108**
OCR2_C	Port C Output Configuration Register 2	**0x0021C208**
OCR2_D	Port D Output Configuration Register 2	**0x0021C308**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	OCR2															
	pin 31		pin 30		pin 29		pin 28		pin 27		pin 26		pin 25		pin 24	
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	OCR2															
	pin 23		pin 22		pin 21		pin 20		pin 19		pin 18		pin 17		pin 16	
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 29-6. Output Configuration Register 2 Description

Name	Description	Settings		
OCR2 [i] Bits 31–0	**Output Configuration**—Corresponds to pins 16–31 of the port and defines which one of the four options is selected as the output signal GPIO-Out. Each port pin [i] (i = 16 through 31) requires two OCR2 bits to determine the output signal.	**OCR2 [2i–32 + 1]**	**OCR2 [2i–32]**	**Output Selected**
		0	0	External input AIN [i]
		0	1	External input BIN [i]
		1	0	External input CIN [i]
		1	1	Data Register [i]

29.5.3 Input Configuration Registers

The input configuration registers ICONFA1 and ICONFA2 specify the signal or value driven to the AOUT bus. The input configuration registers ICONFB1 and ICONFB2 specify the signal or value driven to the BOUT bus. There are two bits in the input configuration registers for each port pin.

There are four distinct input configuration registers for each port; each holds the data for one of the four GPIO ports (Port A, Port B, Port C, and Port D).

29.5.3.1 Input Configuration Register A1

		Addr
ICONFA1_A	Port A Input Configuration Register A1	**0x0021C00C**
ICONFA1_B	Port B Input Configuration Register A1	**0x0021C10C**
ICONFA1_C	Port C Input Configuration Register A1	**0x0021C20C**
ICONFA1_D	Port D Input Configuration Register A1	**0x0021C30C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ICONFA1															
	pin 15		pin 14		pin 13		pin 12		pin 11		pin 10		pin 9		pin 8	
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0xFFFF															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ICONFA1															
	pin 7		pin 6		pin 5		pin 4		pin 3		pin 2		pin 1		pin 0	
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0xFFFF															

Table 29-7. Input Configuration Register A1 Description

Name	Description	Settings		
ICONFA1 [i] Bits 31–0	**Input Configuration**—Corresponds to pins 0–15 of the port and defines which one of the four options is driven to AOUT [i]. Each port pin [i] (i = 0 through 15) requires two ICONFA1 bits to determine the input value.	**ICONFA1 [2i + 1]**	**ICONFA1 [2i]**	**Input Selected**
		0	0	GPIO-In [i]
		0	1	Interrupt Status Register [i]
		1	0	0
		1	1	1

29.5.3.2 Input Configuration Register A2

		Addr
ICONFA2_A	Port A Input Configuration Register A2	**0x0021C010**
ICONFA2_B	Port B Input Configuration Register A2	**0x0021C110**
ICONFA2_C	Port C Input Configuration Register A2	**0x0021C210**
ICONFA2_D	Port D Input Configuration Register A2	**0x0021C310**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ICONFA2															
	pin 31		pin 30		pin 29		pin 28		pin 27		pin 26		pin 25		pin 24	
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0xFFFF															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ICONFA2															
	pin 23		pin 22		pin 21		pin 20		pin 19		pin 18		pin 17		pin 16	
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0xFFFF															

Table 29-8. Input Configuration Register A2 Description

Name	Description	Settings		
		ICONFA2 [2i – 32 + 1]	**ICONFA2 [2i – 32]**	**Input Selected**
ICONFA2 [i] Bits 31–0	**Input Configuration**—Corresponds to pins 16–31 of the port and defines which one of the four options is driven to AOUT [i]. Each port pin [i] (i = 16 through 31) requires two ICONFA2 bits to determine the input value.	0	0	GPIO-In [i]
		0	1	Interrupt Status Register [i]
		1	0	0
		1	1	1

29.5.3.3 Input Configuration Register B1

		Addr
ICONFB1_A	Port A Input Configuration Register B1	**0x0021C014**
ICONFB1_B	Port B Input Configuration Register B1	**0x0021C114**
ICONFB1_C	Port C Input Configuration Register B1	**0x0021C214**
ICONFB1_D	Port D Input Configuration Register B1	**0x0021C314**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ICONFB1															
	pin 15		pin 14		pin 13		pin 12		pin 11		pin 10		pin 9		pin 8	
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0xFFFF															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ICONFB1															
	pin 7		pin 6		pin 5		pin 4		pin 3		pin 2		pin 1		pin 0	
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0xFFFF															

Table 29-9. Input Configuration Register B1 Description

Name	Description	Settings		
ICONFB1 [i] Bits 31–0	**Input Configuration**—Corresponds to pins 0–15 of the port and defines which one of the four options is driven to BOUT [i]. Each port pin [i] (i = 0 through 15) requires two ICONFB1 bits to determine the input value.	**ICONFB1 [2i + 1]**	**ICONFB1 [2i]**	**Input Selected**
		0	0	GPIO-In [i]
		0	1	Interrupt Status Register [i]
		1	0	0
		1	1	1

29.5.3.4 Input Configuration Register B2

		Addr
ICONFB2_A	Port A Input Configuration Register B2	**0x0021C018**
ICONFB2_B	Port B Input Configuration Register B2	**0x0021C118**
ICONFB2_C	Port C Input Configuration Register B2	**0x0021C218**
ICONFB2_D	Port D Input Configuration Register B2	**0x0021C318**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ICONFB2															
	pin 31		pin 30		pin 29		pin 28		pin 27		pin 26		pin 25		pin 24	
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0xFFFF															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ICONFB2															
	pin 23		pin 22		pin 21		pin 20		pin 19		pin 18		pin 17		pin 16	
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0xFFFF															

Table 29-10. Input Configuration Register B2 Description

Name	Description	Settings		
ICONFB2 [i] Bits 31–0	**Input Configuration**—Corresponds to pins 16–31 of the port and defines which one of the four options is driven to BOUT [i]. Each port pin [i] (i = 16 through 31) requires two ICONFB2 bits to determine the input value.	**ICONFB2 [2i – 32 + 1]**	**ICONFB2 [2i – 32]**	**Input Selected**
		0	0	GPIO-In [i]
		0	1	Interrupt Status Register [i]
		1	0	0
		1	1	1

29.5.4 Data Registers

The Port X Data Register holds the value to be output from port X when a pin is configured as an output and the Data Register is chosen in the Output Configuration Register 1 and Output Configuration Register 2.

There are four distinct Data Registers; each holds the data for one of the four GPIO ports (Port A, Port B, Port C, and Port D).

		Addr
DR_A	Port A Data Register	**0x0021C01C**
DR_B	Port B Data Register	**0x0021C11C**
DR_C	Port C Data Register	**0x0021C21C**
DR_D	Port D Data Register	**0x0021C31C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DR															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DR															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 29-11. Data Register Description

Name	Description	Settings
DR [i] Bits 31–0	**Data**—Contains the GPIO output values when the Output Configuration Registers select the Data Register as the output for the pin (selection 11).	0 = Drives the output signal low 1 = Drives the output signal high

29.5.5 GPIO In Use Registers

The GPIO In Use Registers control a multiplexer in the IOMUX module. The settings in these registers choose whether a pin is used for a peripheral function or for its GPIO function (see Figure 29-1 on page 29-2).

There are four distinct GPIO In Use Registers; each holds the data for one of the four GPIO ports (Port A, Port B, Port C, and Port D).

		Addr
GIUS_A	Port A GPIO In Use Register	**0x0021C020**
GIUS_B	Port B GPIO In Use Register	**0x0021C120**
GIUS_C	Port C GPIO In Use Register	**0x0021C220**
GIUS_D	Port D GPIO In Use Register	**0x0021C320**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	GIUS															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	Reset Value A = INUSE_RESET_SEL [31:16] = 0x00C3 Reset Value B = INUSE_RESET_SEL [31:16] = 0xFFFF Reset Value C = INUSE_RESET_SEL [31:16] = 0x0007 Reset Value D = INUSE_RESET_SEL [31:16] = 0xFFFF															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	GIUS															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	Reset Value A = INUSE_RESET_SEL [15:0] = 0xFFFE Reset Value B = INUSE_RESET_SEL [15:0] = 0xFFFF Reset Value C = INUSE_RESET_SEL [15:0] = 0xFFFF Reset Value D = INUSE_RESET_SEL [15:0] = 0xFFFF															

Table 29-12. GPIO In Use Register Description

Name	Description	Settings
GIUS [i] Bits 31–0	**GPIO In Use**—Informs the IOMUX module whether the port pin in used for its GPIO function. When the pin is used for its GPIO function, the multiplexed functions are not available. The reset value of this register is determined by the input value of the signal INUSE_RESET_SEL [31:0].	0 = Pin used for multiplexed function 1 = Pin used for GPIO function

29.5.6 Sample Status Registers

The read-only Sample Status Registers contain the value of the GPIO pins for the port. The register is updated on every clock. The contents are used as a status value when the pins are configured as inputs.

There are four distinct Sample Status Registers; each holds the data for one of the four GPIO ports (Port A, Port B, Port C, and Port D).

		Addr
SSR_A	Port A Sample Status Register	**0x0021C024**
SSR_B	Port B Sample Status Register	**0x0021C124**
SSR_C	Port C Sample Status Register	**0x0021C224**
SSR_D	Port D Sample Status Register	**0x0021C324**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SSR															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SSR															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 29-13. Sample Status Register Description

Name	Description	Settings
SSR [i] Bits 31–0	**Sample Status**—Contains the value of the GPIO pin [i]. It is sampled on every clock.	0 = Pin value is low 1 = Pin value is high

29.5.7 Interrupt Configuration Registers

These registers specify the external interrupt configuration for each of the 32 interrupts. There are two bits in the interrupt configuration registers for each port pin.

There are two interrupt configuration registers for each port; each holds the data for one of the four GPIO ports (Port A, Port B, Port C, and Port D).

29.5.7.1 Interrupt Configuration Register 1

		Addr
ICR1_A	Port A Interrupt Configuration Register 1	**0x0021C028**
ICR1_B	Port B Interrupt Configuration Register 1	**0x0021C128**
ICR1_C	Port C Interrupt Configuration Register 1	**0x0021C228**
ICR1_D	Port D Interrupt Configuration Register 1	**0x0021C328**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ICR1															
	pin 15		pin 14		pin 13		pin 12		pin 11		pin 10		pin 9		pin 8	
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ICR1															
	pin 7		pin 6		pin 5		pin 4		pin 3		pin 2		pin 1		pin 0	
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 29-14. Interrupt Configuration Register 1 Description

Name	Description	Settings		
ICR1 [i] Bits 31–0	**Interrupt Configuration**—Corresponds to interrupts 0–15 of the port and defines which one of the four options is the sensitivity of the interrupt. Each interrupt [i] (i = 0 through 15) requires two ICR1 bits to determine the sensitivity.	**ICR1 [2i + 1]**	**ICR1 [2i]**	**Sensitivity Selected**
		0	0	Positive edge sensitive
		0	1	Negative edge sensitive
		1	0	Positive level sensitive
		1	1	Negative level sensitive

29.5.7.2 Interrupt Configuration Register 2

		Addr
ICR2_A	Port A Interrupt Configuration Register 2	**0x0021C02C**
ICR2_B	Port B Interrupt Configuration Register 2	**0x0021C12C**
ICR2_C	Port C Interrupt Configuration Register 2	**0x0021C22C**
ICR2_D	Port D Interrupt Configuration Register 2	**0x0021C32C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ICR2															
	pin 31		pin 30		pin 29		pin 28		pin 27		pin 26		pin 25		pin 24	
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ICR2															
	pin 23		pin 22		pin 21		pin 20		pin 19		pin 18		pin 17		pin 16	
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 29-15. Interrupt Configuration Register 2 Description

Name	Description	Settings		
ICR2 [i] Bits 31–0	**Interrupt Configuration**—Corresponds to interrupts 16–31of the port and defines which one of the four options is the sensitivity of the interrupt. Each interrupt [i] (i = 16 through 31) requires two ICR2 bits to determine the sensitivity.	**ICR2 [2i – 32 + 1]**	**ICR2 [2i – 32]**	**Sensitivity Selected**
		0	0	Positive edge sensitive
		0	1	Negative edge sensitive
		1	0	Positive level sensitive
		1	1	Negative level sensitive

29.5.8 Interrupt Mask Registers

The Interrupt Mask Registers determine whether an interrupt is active. When an interrupt event occurs and the bit in this register is set (active), then the corresponding bit in the Interrupt Status Register (ISR) is set.

There are four distinct Interrupt Mask Registers; each holds the data for one of the four GPIO ports (Port A, Port B, Port C, and Port D).

		Addr
IMR_A	Port A Interrupt Mask Register	**0x0021C030**
IMR_B	Port B Interrupt Mask Register	**0x0021C130**
IMR_C	Port C Interrupt Mask Register	**0x0021C230**
IMR_D	Port D Interrupt Mask Register	**0x0021C330**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	IMR															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	IMR															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 29-16. Interrupt Mask Register Description

Name	Description	Settings
IMR [i] Bits 31–0	**Interrupt Mask**—Masks the interrupts for this module.	0 = Interrupt is masked 1 = Interrupt is not masked

29.5.9 Interrupt Status Registers

The Interrupt Status Registers indicate whether an interrupt has occurred. When an interrupt event occurs and the corresponding bit in the Interrupt Mask Register (IMR) is set (interrupt is active), then the bit in this register is set.

There are four distinct Interrupt Status Registers; each holds the data for one of the four GPIO ports (Port A, Port B, Port C, and Port D).

		Addr
ISR_A	Port A Interrupt Status Register	**0x0021C034**
ISR_B	Port B Interrupt Status Register	**0x0021C134**
ISR_C	Port C Interrupt Status Register	**0x0021C234**
ISR_D	Port D Interrupt Status Register	**0x0021C334**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ISR															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ISR															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 29-17. Interrupt Status Register Description

Name	Description	Settings
ISR [i] Bits 31–0	**Interrupt Status**—Indicates whether the interrupt [i] has occurred for this module. Write 1 to clear.	0 = Interrupt has not occurred 1 = Interrupt has occurred

29.5.10 General Purpose Registers

The General Purpose Registers control a multiplexer in the IOMUX module. The settings in these registers choose whether a pin is used for its primary peripheral function or for its alternate peripheral function (see Figure 29-1 on page 29-2).

There are four distinct General Purpose Registers; each holds the data for one of the four GPIO ports (Port A, Port B, Port C, and Port D).

		Addr
GPR_A	Port A General Purpose Register	**0x0021C038**
GPR_B	Port B General Purpose Register	**0x0021C138**
GPR_C	Port C General Purpose Register	**0x0021C238**
GPR_D	Port D General Purpose Register	**0x0021C338**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	GPR															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	GPR															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0x0000															

Table 29-18. General Purpose Register Description

Name	Description	Settings
GPR [i] Bits 31–0	**General Purpose**—Selects between the primary and alternate functions of the pin. When the associated bit in the GIUS register is set, this bit has no meaning. **Note:** Ensure that this bit is cleared when there is no alternate function for a particular pin.	0 = Select primary pin function 1 = Select alternate pin function

29.5.11 Software Reset Registers

The Software Reset Registers control the reset of the GPIO module. When the SWR bit of the Port X Software Reset Register is set, the GPIO circuitry for Port X resets immediately.

There are four distinct Software Reset Registers; each holds the data for one of the four GPIO ports (Port A, Port B, Port C, and Port D).

		Addr
SWR_A	Port A Software Reset Register	**0x0021C03C**
SWR_B	Port B Software Reset Register	**0x0021C13C**
SWR_C	Port C Software Reset Register	**0x0021C23C**
SWR_D	Port D Software Reset Register	**0x0021C33C**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																SWR
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x0000								

Table 29-19. Software Reset Register Description

Name	Description	Settings
Reserved Bits 31–1	Reserved—These bits are reserved and should read 0.	
SWR Bit 0	**Software Reset**—Controls software reset of the port. The reset signal is active for 3 system clock cycles and then it is released automatically.	0 = No effect 1 = GPIO circuitry for Port X reset

29.5.12 Pull_Up Enable Registers

The Pull_Up Enable Registers enable or disable the pull-up on each port pin. When the pull-up is disabled, then the pin is tri-stated when not driven.

There are four distinct Pull_Up Enable Registers; each holds the data for one of the four GPIO ports (Port A, Port B, Port C, and Port D).

NOTE:

The Port C Pull_Up Enable Register has a different reset value than the other Port A, B or D Pull_Up Enable Registers.

		Addr
PUEN_A	Port A Pull_Up Enable Register	**0x0021C040**
PUEN_B	Port B Pull_Up Enable Register	**0x0021C140**
PUEN_C	Port C Pull_Up Enable Register	**0x0021C240**
PUEN_D	Port D Pull_Up Enable Register	**0x0021C340**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PUEN															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	1	1	1	1	1	1[1]	1[1]	1	1[1]	1[1]	1[1]	1	1[1]	1[1]	1[1]	1[1]
	0xFFFF/0xF910[1]															

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PUEN															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0xFFFF															

1. The reset value for PUEN_C is 0xF910FFFF.

Table 29-20. Pull_Up Enable Register Description

Name	Description	Settings
PUEN [i] Bits 31–0	**Pull_Up Enable**—Determines whether the port pin [i] is pulled up[1] to a logic high. When the pin is configured as an input, clearing this bit causes the signal to be tri-stated when not driven by an external source. When the pin is configured as an output, clearing this bit causes the signal to be tri-stated when it is not enabled.	0 = Pin [i] is tri-stated when not driven internally or externally 1 = Pin [i] is pulled high[1] when not driven internally or externally

1. PUEN_B bit 11 is pulled down when the bit is set.

Index

Numerics

32 kHz Clock signal, *see* CLK32 signal

A

ACCU COUNTER field, 15-10
AIPI
    data access to 16-bit peripherals, 7-16
    data access to 32-bit peripherals, 7-17
    data access to 8-bit peripherals, 7-15
    features, 7-1
    general information, 7-1
    non-natural size access, 7-18
    overview, 7-1
    programming example, 7-15
    programming model, 7-8
AIPI 1 & 2 peripheral size register 0, *see* PSR0_*n*,
AIPI 1 & 2 peripheral size register 1, *see* PSR1_*n*,
AIPI module register memory map, 7-8
AIPI peripheral address MODULE_EN numbers, 7-8
AIPI registers
    PAR_*n*, 7-12
    PCR_*n*, 7-13
    PSR0_*n*, 7-10
    PSR1_*n*, 7-11
    TSR_*n*, 7-14
AITC
    assigning interrupt sources, 10-33
    enabling interrupt sources, 10-33
    features, 10-1
    input signals, 10-3
    interrupt assignments, 10-3
    introduction, 10-1
    operation, 10-2-10-3
    prioritization of interrupt sources, 10-33
    programming model, 10-4
    register field summary, 10-5
    register memory map, 10-4
    writing reentrant normal interrupt routines, 10-35
ALRM_HM register
    HOURS field, 21-9
    MINUTES field, 21-9
ALRM_HM register, 21-9
ALRM_SEC register
    SECONDS field, 21-10
ALRM_SEC register, 21-10
ARGH register, 18-29
ARGH register, ARGUMENT HIGH field, 18-30
ARGL register
    ARGUMENT LOW field, 18-30
ARGL register, 18-30
ARM Thumb
    instruction set, 4-7
ARM920T processor
    ARMv4T Architecture, 4-4
    cache lock-down, 4-3
    caches, 4-3
    conditional execution, 4-5
    control coprocessor, 4-4
    exception types, 4-5
    instruction classes, 4-5-4-7
    instruction set, 4-7
    introduction, 4-1
    macrocell, 4-2
    MMUs, 4-3
    modes and exception handling, 4-4
    modes and registers, 4-9
    PATAG RAM, 4-3
    prioritization of exception sources, 10-33
    Registers, 4-4
    reset, 6-2
    status registers, 4-5
    system controller, 4-3
    typical interrupt entry sequences, 10-34
    write buffer, 4-3
ARMMCMSEL field, 15-27
Atomic read/modify/write sequence, 10-10
AUCR3_1 register
    WAKEN bit, 24-33

B

BIPR*x*_*x*
    INCPI field, 24-49
BIPR*x*_*x*, 24-49
BITSEL field, 15-15
BLK_LEN register
    BLOCK LENGTH field, 18-23
BLK_LEN register, 18-23
BLR*x* register
    BL bit, 13-26
BLR*x* register, 13-26
BMPR*x*_*x* register
    MODI field, 24-50
BMPR*x*_*x* register, 24-50
Bootstrap mode
    bootloader flowchart, 9-7
    B-record examples, 9-3
    changing communication speed, 9-3
    entering, 9-1
    instruction buffer, 9-3
    operation of, 9-1
    programming notes, 9-7
    read/write examples, 9-5
    record format, 9-2

register usage, 9-2
setting up RS-232 terminal, 9-3
BUCR*x* register
CCNT field, 13-29
BUCR*x* register, 13-29
BUFFER_ACCESS register
FIFO CONTENT field, 18-32
BUFFER_ACCESS register, 18-32
BURST COUNT bits, 15-14

C

CACHE CLR bit, 15-9
CACHE EN bit, 15-9
CAP field, 23-4
CAPT bit, 23-9
Capture edge field, *see* CAP field
Capture edge, *see* CAP bit
Capture event, *see* CAPT bit
CAPTURE VALUE field, 23-7
CCR*x* register
CEN bit, 13-24
DMOD field, 13-22
DSIZ field, 13-23
FRC bit, 13-23
MDIR bit, 13-23
MSEL bit, 13-23
REN bit, 13-23
RPT bit, 13-23
SMOD field, 13-23
SSIZ field, 13-23
CCR*x* register, 13-22
Channel *x* burst length register, *see* BLR*x* register 13-26
Channel *x* bus utilization control register, *see* BUCR*x* register
Channel *x* control register, *see* CCR*x* register
Channel *x* count register, *see* CNTR*x* register
Channel *x* destination address register, *see* DAR*x* register
Channel *x* request source select register, *see* RSSR*x* register
Channel *x* request time-out register, *see* RTOR*x* register
Channel *x* source address register, *see* SAR*x* register
Chip select 0 control registers *see* CSOU and CSOL registers
Chip select 1 - 5 control registers *see* CS1U - CS5U and CS1L - CS5L registers
CLK_RATE register
CLK RATE field, 18-19
PRESCALER field, 18-18
CLK_RATE register, 18-18
CLK32 signal, 6-3
CLKSOURCE field, 23-5
Clock source control register, *see* CSCR register
Clock source, *see* CLKSOURCE field
CMD register
COMMAND NUMBER field, 18-29
CMD register, 18-29
CMD_DAT_CONT register
BUS_WIDTH field, 18-20
BUSY bit, 18-20
CMD_RESP_LONG_OFF bit, 18-19
DATA_ENABLE bit, 18-20
FORMAT_OF_RESPONSE field, 18-20
INIT bit, 18-20
START_READWAIT bit, 18-20
STOP_READWAIT bit, 18-19
STREAM_BLOCK bit, 18-20
WRITE_READ bit, 18-20
CMD_DAT_CONT register, 18-19
CMOS sensor
auto exposure, 28-13
auto focus, 28-15
data FIFO operation, 28-3
interrupt operation, 28-4
introduction, 28-1
module architecture, 28-1
operation, 28-3
pin configuration, 28-3
programming model, 28-4
signal description, 28-2
statistic data packing, 28-15
CNTR*x* register
CNT field, 13-21
CNTR*x* register, 13-21
COLUMN field, 15-19, 15-23
COMP bit, 23-9
Compare event, *see* COMP bit
COMPARE VALUE field, 23-6
CONTROLREG1 register, 16-8
CONTROLREG2 register, 16-8
CONTROLREG*x* register
BIT_COUNT field, 16-9
DATARATE field, 16-8
DRCTL field, 16-8
MODE bit, 16-8
PHA bit, 16-9
POL bit, 16-9
SPIEN bit, 16-9
SSCTL bit, 16-9
SSPOL bit, 16-9
XCH bit, 16-9
Core test reset, *see* $\overline{\text{CORE_TRST}}$ signal
$\overline{\text{CORE_TRST}}$ signal, 6-3
COUNT field, 23-8
Counter value, *see* COUNT field
CP15, *see* ARM920T processor, control coprocessor,
CPOS register
CC field, 17-29
CXP field, 17-29
CYP field, 17-29

OP bit, 17-29
CPOS register, 17-29
CS0L register, 11-12
CS0U register, 11-11
CS1L register, 11-13
CS2L register, 11-13
CS3L register, 11-13
CS4L register, 11-13
CS5L register, 11-13
CSCR register
BCLK_DIV field, 12-7
CLKO_SEL field, 12-6
MPEN bit, 12-7
MPLL_RESTART bit, 12-7
OSC_EN bit, 12-7
PRESC bit, 12-7
SD_CNT field, 12-6
SPEN bit, 12-7
SPLL_RESTART bit, 12-7
System_SEL bit, 12-7
USB_DIV field, 12-6
CSCR register, 12-6
CSI Control Register 1, *see* CSICR1 register
CSI control register 2, *see* CSICR2 register
CSI RxFIFO register 1, *see* CSIRXR register
CSI statistic FIFO register 1, *see* CSISTATR register
CSI status register 1, *see* CSISR register
CSICR1 register
BIG_ENDIAN bit, 28-6
CLR_RXFIFO bit, 28-6
CLR_STATFIFO bit, 28-6
EN bit, 28-7
FCC bit, 28-6
GCLK_MODE bit, 28-7
INV_DATA bit, 28-7
INV_PCLK bit, 28-7
MCLKDIV field, 28-6
MCLKEN bit, 28-6
REDGE bit, 28-7
RF_OR_INTEN bit, 28-5
RXFF_INTEN bit, 28-5
RXFF_LEVEL field, 28-5
SF_OR_INTEN bit, 28-5
SOF_INTEN bit, 28-6
SOF_POL bit, 28-6
STATFF_INTEN bit, 28-5
STATFF_LEVEL field, 28-5
CSICR1 register, 28-5
CSICR2 register
AFS field, 28-8
BTS field, 28-8
DRM bit, 28-8
HSC field, 28-9
LVRM field, 28-9
SCE bit, 28-8
VSC field, 28-9
CSICR2 register, 28-8
CSIRXR register
IMAGE field, 28-12
CSIRXR register, 28-12
CSISR register
DRDY bit, 28-11
RFF_OR_INT bit, 28-10
RXFF_INT bit, 28-10
SFF_OR_INT bit, 28-10
SOF_INT bit, 28-11
STATFF_INT bit, 28-10
CSISR register, 28-10
CSISTATR register
STAT field, 28-11
CSISTATR register, 28-11
CTS
de-assertion, programmable, 24-8

D

DAR*x* register
DA field, 13-20
DAR*x* register, 13-20
DATA field, 15-34
DATAINSHIFT field, 15-26
DAYALARM register
DAYSAL field, 21-8
DAYALARM register, 21-8
DAYR register
DAYS field, 21-5
DAYR register, 21-5
DBOSR register
bits CH10 through CH0, 13-14
DBOSR register, 13-14
DBTOCR register
CNT field, 13-15
EN bit, 13-15
DBTOCR register, 13-15
DBTOSR register
bits CH10 through CH0, 13-11
DBTOSR register, 13-11
DCR register
DEN bit, 13-8
DRST bit, 13-8
DCR register, 13-8
DCT ENA bit, 15-27
DCT/IDCT bit, 15-27
DCT_DES_ADDR field, 15-30
DCT_SRC_ADDR field, 15-30
DCTBYPASS bit, 15-27
DCTCLKEN bit, 15-26
DCTCOMP bit, 15-28, 15-29

DCTXPOSE bit, 15-26
DDIR_A register, 29-8
DDIR_B register, 29-8
DDIR_C register, 29-8
DDIR_D register, 29-8
DDIR_*x*
    DDIR field, 29-8
DIDEN bit, 15-28
DIIEN bit, 15-28
DIMR register
    bits CH10 through CH0, 13-10
DININTR bit, 15-29
DISNUM field, 10-10
DISR register, 13-9
DMA buffer overflow status register, *see* DBOSR register
DMA burst time-out control register, *see* DBTOCR register
DMA burst time-out status register, *see* DBTOSR register
DMA control register, *see* DCR register
DMA control register, *see* DMACR register
DMA interrupt mask register, *see* DIMR register
DMA interrupt status register, *see* DISR register
DMA request time-out status register, *see* DRTOSR register
DMA transfer error status register, *see* DSESR register
DMAC
    2D memory registers (A & B), 13-16
    big endian/little endian byte-ordering, 13-4
    block diagram, 13-2
    channel registers, 13-18
    DMA request table, 13-29
    features, 13-1
    general registers, 13-8
    programming model, 13-4
    signal description, 13-3
DMACR register
    BURST bit, 17-36
    HM field, 17-36
    TM field, 17-36
DMACR register, 17-36
DMAREG1 register, 16-14
DMAREG2 register, 16-14
DMAREG*x* register
    RFDEN bit, 16-14
    RHDEN bit, 16-15
    RHDMA bit, 16-15
    TEDEN bit, 16-14
    TEDMA bit, 16-15
    THDEN bit, 16-14
    THDMA bit, 16-15
DODEN bit, 15-28
DOIEN bit, 15-28
DOUTINTR bit, 15-29
DR_A register, 29-15
DR_B register, 29-15
DR_C register, 29-15
DR_D register, 29-15
DR_*x* register
    DR field, 29-15
DRAM reset, *see* $\overline{\text{RESET_DRAM}}$ signal
DRTOSR register
    bits CH10 through CH0, 13-12
DRTOSR register, 13-12
DSESR register
    bits CH10 through CH0, 13-13
DSESR register, 13-13
DTR edge triggered interrupt, 24-7

E

EIM
    address bus, 11-1
    burst mode signals, 11-3
    chip select outputs, 11-2
    control signals, 11-2
    data bus, 11-1
    I O signals, 11-1
    overview, 11-1
    pin configuration, 11-3
    programming model, 11-10
    read and write signals, 11-1
    system connections, typical, 11-5
EIM configuration register *see* EIM register
EIM functionality
    burst clock divisor, 11-8
    burst clock start, 11-9
    burst mode operation, 11-8
    configurable bus sizing, 11-8
    error conditions, 11-9
    page mode emulation, 11-9
    programmable output generation, 11-8
EIM register, 11-21
Embedded Trace Macrocell, *see* ETM
Endpoint*x* data register, *see* USB_EP*x*_FDAT register
Endpoint*x* FIFO alarm register, *see* USB_EP*x*_FALRM register
Endpoint*x* FIFO write pointer register, *see* USB_EP*x*_FWRP register
Endpoint*x* interrupt mask register, *see* USB_EP*x*_MASK register
Endpoint*x* interrupt status register, *see* USB_EP*x*_INTR register
Endpoint*x* last read frame pointer register, *see* USB_EP*x*_LRFP register
Endpoint*x* last write frame pointer register, *see* USB_EP*x*_LWFP register
Endpoint*x*FIFO read pointer register, *see* USB_EP*x*_FRDP register,
Endpoint*x*FIFO Status Register, *see* USB_EP*x*_FSTAT register

ENNUM field, 10-9
ERR INTR bit, 15-29
ERRINTREN bit, 15-28
ETM
block diagram, 5-1
introduction, 5-1
pin configuration, 5-2
registers, programming and reading, 5-2
EXR bit, 6-4
External Interface Module *see* EIM module
External reset bit, *see* EXR bit

F

Fast interrupt
arbiter disable, *see* FIAD bit
pending bit, *see* FIPEND field
pending register high, *see* FIPNDH register
pending register low, *see* FIPNDL register
vector and status register, *see* FIVECSR register
vector, *see* FIVECTOR field
Fast interrupt pending bit, *see* FIPEND field
FIFO REGISTER bits, 15-13
FIFO COUNT bits, 15-14
FIFO EMP bit, 15-29
FIFO EMPT bit, 15-11, 15-12, 15-14
FIFO FULL bit, 15-11, 15-12, 15-14, 15-29
FIFO HALF bit, 15-11, 15-12, 15-14
FIPEND field, 10-31, 10-32
FIPNDH register, 10-31
FIPNDL register, 10-32
FIVECSR register, 10-24
FIVECTOR field, 10-24
FORCE field, 10-27, 10-28
Four bits/pixel grayscale mode
GPM field, 17-39
Free-run/restart, *see* FRR bit
FRR bit, 23-4
Function multiplexing control register, *see* FMCR register

G

General purpose timers
programming model, 23-3
GIUS_A register, 29-16
GIUS_B register, 29-16
GIUS_C register, 29-16
GIUS_D register, 29-16
GIUS_*x* register
GIUS field, 29-16
Global peripheral control register, *see* GPCR register
GPIO
data direction registers, 29-8
data registers, 29-15
general description, 29-1
general purpose registers, 29-22
in use registers, 29-16
Input configuration registers, 29-11
interrupt configuration registers, 29-18
interrupt mask registers, 29-20
interrupt status registers, 29-21
interrupts, 29-3
module block diagram, 29-4
module features, 29-2
module overview, 29-2
output configuration registers, 29-9
pin configuration, 29-4
programming model, 29-5
pull_up enable registers, 29-24
sample status registers, 29-17
signal description, 29-3
software reset registers, 29-23
GPR_A register, 29-22
GPR_B register, 29-22
GPR_C register, 29-22
GPR_D register, 29-22
GPR_*x* register
GPR field, 29-22

H

Hard asynchronous reset, *see* $\overline{\text{HARD_ASYN_RESET}}$ signal
Hard reset signal, *see* $\overline{\text{HARD_RESET}}$ signal
$\overline{\text{HARD_ASYN_RESET}}$ signal, 6-3
HCR register
H_WAIT_1 field, 17-26
H_WAIT_2 field, 17-26
H_WIDTH field, 17-26
HCR register, 17-26
Horizontal configuration register, *see* HCR register
HOURMIN register
HOURS field, 21-6
HOURMIN register, 21-6
HOURMIN register, MINUTES field, 21-6
$\overline{\text{HRESET}}$ signal, 6-3

I

I^2C
Address Register, *see* IADR register
arbitration procedure, 26-4, 26-5
clock stretching, 26-5
clock synchronization, 26-4
handshaking, 26-5
initialization sequence, 26-14
interface features, 26-1
lost arbitration, 26-15
module, *see* I^2C

overview, 26-1
pin configuration, 26-5
programming
examples, 26-13
model, 26-6
protocol, communication, 26-3
repeated START generation, 26-15
slave mode, 26-15
software response, post transfer, 26-14
START signal, generation, 26-14
STOP generation, 26-15
I^2C control register, *see* I2CR register
I^2C data I/O register, *see* I2DR register
I^2C frequency divider register, *see* IFDR register
I^2C status register, *see* I2CSR register
I2CR register
IEN bit, 26-10
IIEN bit, 26-10
MSTA bit, 26-10
MTX bit, 26-11
RSTA bit, 26-11
TXAK bit, 26-11
I2CR register, 26-10
I2CSR register
IAAS bit, 26-12
IAL bit, 26-12
IBB bit, 26-12
ICF bit, 26-12
IIF bit, 26-12
RXAK bit, 26-12
SRW bit, 26-12
I2CSR register, 26-11
I2DR register
D field, 26-13
I2DR register, 26-13
IADR register
ADR field, 26-7
IADR register, 26-7
ICONFA1_A register, 29-11
ICONFA1_C register, 29-11
ICONFA1_D register, 29-11
ICONFA1_*x* register
ICONFA1 field, 29-11
ICONFA2_A register, 29-12
ICONFA2_B register, 29-12
ICONFA2_C register, 29-12
ICONFA2_D register, 29-12
ICONFA2_*x* register
ICONFA2 field, 29-12
ICONFB1_A register, 29-13
ICONFB1_B register, 29-13
ICONFB1_C register, 29-13
ICONFB1_D register, 29-13
ICONFB1_*x* register
ICONFB1 field, 29-13
ICONFB2_A register, 29-14
ICONFB2_B register, 29-14
ICONFB2_C register, 29-14
ICONFB2_D register, 29-14
ICONFB2_*x* register
ICONFB2 field, 29-14
ICR1_A register, 29-18
ICR1_B register, 29-18
ICR1_C register, 29-18
ICR1_D register, 29-18
ICR1_*x* register
ICR1 field, 29-18
ICR2_A register, 29-19
ICR2_B register, 29-19
ICR2_C register, 29-19
ICR2_D register, 29-19
ICR2_*x* register
ICR2 field, 29-19
IFDR register
IC field, 26-8
IFDR register, 26-8
IMR_A register, 29-20
IMR_B register, 29-20
IMR_C register, 29-20
IMR_D register, 29-20
IMR_*x* register
IMR field, 29-20
INT_MASK register
AUTO_CARD_DETECT bit, 18-26
BUF_READY bit, 18-27
DAT0_EN bit, 18-26
DATA_TRAN bit, 18-27
END_CMD_RES bit, 18-27
SDIO bit, 18-26
WRITE_OP_DONE bit, 18-27
INT_MASK register, 18-26
INTCNTL register
FIAD bit, 10-7
NIAD bit, 10-7
INTCNTL register, 8-6, 10-6
INTDISNUM register, 10-10
INTENABLE field, 10-11, 10-12
INTENABLEH register, 10-11
INTENABLEL register, 10-12
INTENNUM register, 10-9
Interrupt configuration register, *see* LCDICR register
Interrupt control register, *see* INTCNTL register
Interrupt controller, *see* AITC
Interrupt disable number register, *see* INTDISNUM register
Interrupt disable number, *see* DISNUM field
Interrupt enable number register, *see* INTENNUM register
Interrupt enable number, *see* ENNUM field
Interrupt enable register high, *see* INTENABLEH register

Interrupt enable register low, *see* INTENABLEL register
Interrupt enable, *see* INTENABLE field
Interrupt Force Register High, *see* INTFRCH register
Interrupt force register low, *see* INTFRCL register
Interrupt request enable, *see* IRQEN bit
Interrupt source force request, *see* FORCE field
Interrupt source register high, *see* INTSRCH register
Interrupt source register low, *see* INTSRCL register
Interrupt source, *see* INTIN field
Interrupt source, *see* INTIN field
Interrupt status register, *see* LCDISR register
Interrupt type register high, *see* INTTYPEH register
Interrupt type register low, *see* INTTYPEL register
Interrupt type, *see* INTTYPE field
INTFRCH register, 10-27
INTFRCL register, 10-28
INTIN field, 10-25, 10-26
INTREG1 register, 16-10
INTREG2 register, 16-10
INTREG*x* register
    BO bit, 16-11
    BOEN bit, 16-10
    RF bit, 16-11
    RFEN bit, 16-10
    RH bit, 16-11
    RHEN bit, 16-10
    RO bit, 16-11
    ROEN bit, 16-10
    RR bit, 16-11
    RREN bit, 16-10
    TE bit, 16-11
    TEEN bit, 16-11
    TF bit, 16-11
    TFEN bit, 16-10
    TH bit, 16-11
    THEN bit, 16-10
INTSRCH register, 10-25
INTSRCL register, 10-26
INTTYPE field, 10-13, 10-14
INTTYPEH register, 10-13
INTTYPEL register, 10-14
IRQEN bit, 23-4
ISR_A register, 29-21
ISR_B register, 29-21
ISR_C register, 29-21
ISR_D register, 29-21
ISR_*x* register
    ISR field, 29-21

L

LCD color cursor mapping register, *see* LCHCC register
LCD cursor position, *see* CPOS register
LCD cursor width height and blink register, *see* LCWHB register
LCD gray palette mapping register, *see* LSCR1 register
LCDC
    active matrix panel interface signals
        "digital CRT", 17-14
    active matrix panel interface signals, 17-14
    active panel interface timing, 17-16
    black-and-white operation, 17-7
    color generation, 17-8
    display data mapping, 17-3
    features, 17-1
    frame rate modulation control, 17-10
    gray scale operation, 17-7
    introduction, 17-1
    mapping RAM registers, 17-39
    operation, 17-2
    panel interface signals, 17-11
    panning
        typical panning algorithm, 17-3
    panning, 17-3
    passive panel interface timing, 17-13
    pin configuration
        setting data direction, 17-11
    pin configuration, 17-11
    programming model, 17-18
    screen format
        maximum page width, 17-3
    screen format, 17-2
    using VPH for boundary checks, 17-3
LCDICR register
    INTCON bit, 17-37
    INTSYN bit, 17-37
LCDICR register, 17-37
LCDISR register
    BOF bit, 17-38
    EOF bit, 17-38
    ERR_RES bit, 17-38
    UDR_ERR bit, 17-38
LCDISR register, 17-38
LCHCC register
    CUR_COL_B register, 17-31
    CUR_COL_G register, 17-31
    CUR_COL_R register, 17-31
LCHCC register, 17-31
LCWHB register
    BD field, 17-30
    BK_EN bit, 17-30
    CH field, 17-30
    CW field, 17-30
LCWHB register, 17-30
LENGTH field, 15-19, 15-23
LGPMR register, *see* LSCR1 register
LSCR1 register

CLS_RISE_DELAY field, 17-32
GRAY 1 register, 17-32
GRAY 2 register, 17-32
PS_RISE_DELAY field, 17-32
REV_TOGGLE_DELAY field, 17-32

M

Mapping RAM registers
eight bits/pixel active matrix color mode register, 17-41
eight bits/pixel passive matrix color mode, 17-40
four bits/pixel active matrix color mode, 17-41
four bits/pixel gray-scale mode, 17-39
four bits/pixel passive matrix color mode, 17-40
one bit/pixel mono mode, 17-39
twelve bits/pixel active matrix color mode, 17-42
MC9328MX1
features of, ??-1-10
internal registers, 3-6
MCU PLL and system clock control register 1, *see* MPCTL1 register
MCU PLL control register 0, *see* MPCTL0 register
Memory space
double map image, 3-5
external memory, 3-5
internal register space, 3-5
MCU physical memory map 3-2
memory map description, 3-1
Memory space, 3-1
Memory stick
auto command register, *see* MSACD register
command register, *see* MSCMD register
control 2 register, *see* MSC2 register
control/status register, *see* MSCS register
DMA request control register, *see* MSDRQC register
FIFO access error control/status register, *see* MSFAECS register
host controller, *see* MSHC
parallel port control/data register, *see* MSPPCD register
receive FIFO data register, *see* MSRDATA register
serial clock divider register, *see* MSCLKD register
transmit FIFO data register, *see* MSTDATA register
Memory stick interrupt control/status register, *see* MSICS register
MISCELLANEOUS register
OMA bit, 22-17
RMA0 bit, 22-17
MISCELLANEOUS register, 22-17
MMC/SC
program flow operation, 18-33
protection management, 18-47
MMC/SD
block read example, 18-42
block read, 18-41
card detection, 18-8
card identification state, 18-33
card internal write protection, 18-48
card registry differences MMC and SD, 18-37
clock control register, 18-13
code example
block write with DMA, 18-40
code listing
card detect, 18-33
send cmd wait response, 18-33
command formats, 18-58
command number register, *see* CMD register
command types, 18-57
commands and arguments, 18-28
commands and responses, 18-56
commands, 18-58
commands, two types, 18-32
DMA burst request, 18-6
DMA interface, 18-6
features, 18-1
functional description, 18-5
interrupt handling, 18-9
introduction, 18-1
logic/command interpreter, 18-9
mechanical write protect switch, 18-48
memory controller, 18-8
MMC erase, 18-46
password protect, 18-48
pin configuration, 18-5
programming model, 18-12
reset types, 18-34
response formats, 18-62
signal description, 18-4
specific and general commands, 18-57
stream write, 18-44
system clock controller, 18-11
types of cards, 18-3
voltage validation, 18-34
wide bus selection/de-selection, 18-47
write-error detection, 18-7
MMC/SD block length register, *see* BLK_LEN register
MMC/SD buffer access register, *see* BUFFER_ACCESS register
MMC/SD clock control register, 18-14
MMC/SD clock rate register, *see* CLK_RATE register
MMC/SD command and data control register, *see* CMD_DAT_CONT register
MMC/SD higher argument register, *see* ARGH register
MMC/SD interrupt mask register, *see* INT_MASK register
MMC/SD lower argument register, *see* ARGL register
MMC/SD number of blocks register, *see* NOB register
MMC/SD read time out register, *see* READ_TO register
MMC/SD response fifo register, *see* RES_FIFO register
MMC/SD response time out register, *see* RES_TO register

MMC/SD revision number register, *see* REV_NO register
MMC/SD status register, *see* STATUS register
MOD ENAB bit, 15-7
MPCTL0 register
    MFD field, 12-10
    MFI field, 12-10
    MFN field, 12-10
    PD field, 12-10
MPCTL0 register, 12-9
MPCTL1 register
    BRMO bit, 12-11
MPCTL1 register, 12-11
MSACD register
    ADATASIZE field, 19-21
    APID field, 19-21
MSACD register, 19-20
MSC2 register
    ACD bit, 19-20
    LEND bit, 19-20
    MSCEN bit, 19-20
    RED bit, 19-20
MSC2 register, 19-20
MSCLKD register
    DIV field, 19-23
    SRC bit, 19-22
MSCLKD register, 19-22
MSCMD register
    DATA SIZE field, 19-13
    PID field, 19-13
MSCMD register, 19-13
MSCS register
    BSYCNT field, 19-14
    DAKEN bit, 19-14
    DRQ bit, 19-15
    INT bit, 19-15
    NOCRC bit, 19-14
    PWS bit, 19-14
    RBE bit, 19-15
    RBF bit, 19-15
    RST bit, 19-14
    SIEN bit, 19-14
    TBE bit, 19-15
    TBF bit, 19-15
MSCS register, 19-14
MSDRQC register
    DRQEN bit, 19-23
    RFF bit, 19-24
    TFE bit, 19-24
MSDRQC register, 19-23
MSFAECS register
    FAEEN bit, 19-21
    RUN bit, 19-22
    TOV bit, 19-22
MSFAECS register, 19-21
MSHC
    auto command function, 19-9
    block diagram and description, 19-1
    bus state control operation, 19-5
    data FIFO operation, 19-4
    interrupt sources, 19-5
    memory stick interface, 19-2
    operation, 19-4
    overview, 19-1
    pin configuration, 19-3
    power save mode operation, 19-8
    programmer's reference, 19-24
    programming model, 19-12
    protocol error, 19-28
    protocol, 19-26
    reset operation, 19-7
    serial clock divider operation, 19-11
    serial interface overview, 19-24
    signal description, 19-3
    signal timing, 19-31
    system-level DMA transfer operation, 19-11
    transfer protocol command (TPC), 19-27
MSICS register
    CRC bit, 19-18
    DRQ bit, 19-18
    DRQSL bit, 19-17
    FAE bit, 19-18
    INTEN bit, 19-17
    PIN bit, 19-18
    PINEN bit, 19-17
    RDY bit, 19-17
    SIF bit, 19-18
    TOE bit, 19-18
MSICS register, 19-17
MSPPCD register
    PIEN0 bit, 19-19
    PIEN1 bit, 19-19
    XPIN0 bit, 19-19
    XPIN1 bit, 19-19
MSPPCD register, 19-19
MSRDATA register
    RX DATA BUFFER field, 19-16
MSRDATA register, 19-16
MSTDATA register
    TX DATA BUFFER field, 19-16
MSTDATA register, 19-15
MULT COUNTER field, 15-10
Multimedia card/secure digital host controller, *see* MMC/SD

N

NIMASK field, 10-8
NIMASK register, 10-8
NIPEND field, 10-29, 10-30

NIPNDH register, 10-29
NIPNDL register, 10-30
NIPR0 though NIPR64 fields, 10-15-10-22
NIPRILVL field, 10-23
NIVECSR register, 10-23
NIVECTOR field, 10-23
NOB register
    NOB field, 18-24
NOB register, 18-24
Normal interrupt
    arbiter disable, *see* NIAD bit
    mask, *see* NIMASK register
    pending bit, *see* NIPEND field
    pending register high, *see* NIPNDH register
    priority level register 0, *see* NIPRIORITY0 register
    priority level register 1, *see* NIPRIORITY1 register
    priority level register 2, *see* NIPRIORITY2 register
    priority level register 3, *see* NIPRIORITY3 register
    priority level register 4, *see* NIPRIORITY4 register
    priority level register 5, *see* NIPRIORITY5 register
    priority level register 6, *see* NIPRIORITY6 register
    priority level register 7, *see* NIPRIORITY7 register
    priority level, *see* NIPRILVL field
    vector, *see* NIVECTOR field
Normal interrupt mask, *see* NIMASK field
Normal interrupt pending bit, *see* NIPEND field
Normal interrupt priority level fields, *see* NIPR*xx*
Normal interrupt priority level registers
    description of, 10-14
Normal interrupt vector and status register, *see* NIVECSR register

O

OCR1_A register, 29-9
OCR1_B register, 29-9
OCR1_C register, 29-9
OCR1_D register, 29-9
OCR1_*x* register
    OCR1 field, 29-9
OCR2_A register, 29-10
OCR2_B register, 29-10
OCR2_C register, 29-10
OCR2_D register, 29-10
OCR2_*x* register
    OCR2 field, 29-10
OM bit, 23-4
OP EN bit, 15-9
OP END bit, 15-11, 15-12
Output mode, *see* OM bit

P

Panel Configuration Register, *see* PCR register
Panning offset register, *see* POS register,
PAR_*n* register
    ACCESS field, 7-13
PAR_*n* register, 7-12
Passive matrix panel
    interface signals
        LD bus, 17-12
passive matrix panel interface signals, 17-12
PCDR register
    PCLK_DIV1 field, 12-8
    PCLK_DIV2 field, 12-8
    PCLK_DIV3 field, 12-8
PCDR register, 12-8
PCR register
    ACD field, 17-24
    ACDSEL bit, 17-24
    BPIX field, 17-23
    CLKPOL bit, 17-24
    COLOR bit, 17-23
    END_SEL bit, 17-24
    FLMPOL bit, 17-24
    LPPOL bit, 17-24
    OEPOL bit, 17-24
    PBSIZ field, 17-23
    PCD field, 17-25
    PIXPOL bit, 17-23
    REV_VS bit, 17-24
    SCLKIDLE bit, 17-24
    SCLKSEL bit, 17-24
    SHARP bit, 17-24
    TFT bit, 17-23
PCR register, 17-23
PCR_*n* register
    ACCESS_MODE field, 7-14
PCR_*n* register, 7-13
PERIODREG1 register, 16-13
PERIODREG2 register, 16-13
PERIODREG*x* register
    CSRC bit, 16-13
    WAIT field, 16-14
peripheral access registers, *see* PAR_*n*,
Peripheral clock divider register, *see* PCDR register
peripheral control registers, *see* PCR_*n*,
peripheral size registers, *see* PSR*n*_*n* registers
Phase-locked loop and clock control, *see* PLL and clock control
PLL and clock control
    ARM920T processor low power modes, 12-4
    clock sources, 12-1
    DPLL output frequency calculation, 12-3
    DPLL phase and frequency jitter, 12-3
    generation of 48 MHz clocks, 12-11
    high frequency clock source, 12-2
    introduction, 12-1

low frequency clock source, 12-1
PLL operation at power-up, 12-4
PLL operation at wake-up, 12-4
power management in clock controller, 12-4
programming digital phase locked loops, 12-9
programming model, 12-5
SDRAM power modes, 12-4
POR signal, 6-3
Port A data direction register, *see* DDIR_A register
Port A data register, *see* DR_A register
Port A general purpose register, *see* GPR_A register
Port A GPIO in use register, *see* GIUS_A register
Port A input configuration register A1, *see* ICONFA1_A register
Port A input configuration register A2, *see* ICONFA2_A register
Port A input configuration register B1, *see* ICONFB1_A register
Port A input configuration register B2, *see* ICONFB2_A register
Port A interrupt configuration register 1, *see* ICR1_A register
Port A interrupt configuration register 2, *see* ICR2_A register
Port A interrupt mask register, *see* IMR_A register
Port A interrupt status register, *see* ISR_A register
Port A output configuration register 1, *see* OCR1_A register,
Port A output configuration register 2, *see* OCR1_A register
Port A pull_up enable register, *see* PUEN_A register
Port A sample status register, *see* SSR_A register
Port A software reset register, *see* SWR_A register
Port B data direction register, *see* DDIR_B register
Port B data register, *see* DR_B register
Port B general purpose register, *see* GPR_B register
Port B GPIO in use register, *see* GIUS_B register
Port B input configuration register A1, *see* ICONFA1_B register
Port B input configuration register A2, *see* ICONFA2_B register
Port B input configuration register B1, *see* ICONFB1_B register
Port B input configuration register B2, *see* ICONFB2_B register
Port B interrupt configuration register 1, *see* ICR1_B register
Port B interrupt configuration register 2, *see* ICR2_B register
Port B interrupt mask register, *see* IMR_B register
Port B interrupt status register, *see* ISR_B register
Port B output configuration register 1, *see* OCR1_B register
Port B output configuration register 2, *see* OCR1_B register
Port B pull_up enable register, *see* PUEN_B register
Port B sample status register, *see* SSR_B register
Port B software reset register *see* SWR_B register
Port C data direction register, *see* DDIR_C register
Port C data register, *see* DR_C register
Port C general purpose register, *see* GPR_C register
Port C GPIO in use register, *see* GIUS_C register
Port C input configuration register A1, *see* ICONFA1_C register
Port C input configuration register A2 *see* ICONFA2_C register
Port C input configuration register B1, *see* ICONFB1_C register
Port C input configuration register B2, *see* ICONFB2_C register
Port C interrupt configuration register 1, *see* ICR1_C register
Port C interrupt configuration register 2, *see* ICR2_C register
Port C interrupt mask register, *see* IMR_C register
Port C interrupt status register, *see* ISR_C register
Port C output configuration register 1, *see* OCR1_C register
Port C output configuration register 2, *see* OCR1_C register
Port C pull_up enable register, *see* PUEN_C register
Port C sample status register, *see* SSR_C register
Port C software reset register, *see* SWR_C register
Port D data direction register, *see* DDIR_D register
Port D data register, *see* DR_D register
Port D general purpose register, *see* GPR_D register
Port D GPIO in use register, *see* GIUS_D register
Port D input configuration register A1, *see* ICONFA1_D register
Port D input configuration register A2, *see* ICONFA2_D register
Port D input configuration register B1, *see* ICONFB1_D register
Port D input configuration register B2, *see* ICONFB2_D register
Port D interrupt configuration register 1, *see* ICR1_D register
Port D interrupt configuration register 2, *see* ICR2_D register
Port D interrupt mask register, *see* IMR_D register
Port D interrupt status register, *see* ISR_D register
Port D output configuration register 1, *see* OCR1_D register
Port D output configuration register 2, *see* OCR1_D register
Port D pull_up enable register, *see* PUEN_D register
Port D sample status register, *see* SSR_D register
Port D software reset register, *see* SWR_D register
POS register
POS bit, 17-28
POS register, 17-28
Power-on reset, *see* POR signal
PRESCALER field, 23-5
Prescaler, *see* PRESCALER field
Programming model

AITC, 10-4
CMOS sensor 28-4
DMAC, 13-4
EIM, 11-10
GP timers, 23-3
GPIO, 29-5
I^2C, 26-6
LCDC, 17-18
MMC/SD, 18-12
MSHC, 19-12
PLL and clock control, 12-5
PWM, 20-3
reset module, 6-3
RTC, 21-4
SDRAM memory controller, 22-8
SPI, 16-5
SSI, 27-7
system control, 8-1
UART, 24-21
USB device port 25-6
PSR0_*n* register
MOD_EN_L field, 7-10
PSR0_*n* register, 7-10
PSR1_*n* register
MOD_EN_U field, 7-11
PSR1_*n* register, 7-11
PUEN_A register, 29-24
PUEN_B register, 29-24
PUEN_C register, 29-24
PUEN_D register, 29-24
PUEN_*x* register
PUEN field, 29-24
Pulse-width modulator, *see* PWM
PWM
clock signals, 20-1
clock source selection, 20-1
D/A mode, 20-3
digital-to analog converter mode, 20-3
introduction, 20-1
operation, 20-2
period frequency, calculating, 20-7
playback mode
digital sample values, 20-3
maskable interrupt generation, 20-3
variable pulse width, 20-2
playback mode, 20-2
programming model, 20-3
tone mode, 20-3
PWM contrast control register, *see* PWMR register
PWM counter register, *see* PWMCNT register
PWM Period Register, *see* PWMP register
PWM sample register, *see* PWMS register
PWMCNT register, 20-8
PWMCNT register, COUNT field, 20-8
PWMP register
PERIOD field, 20-7
PWMP register, 20-7
PWMR register
CC_EN bit, 17-34
LDMSK bit, 17-34
PW field, 17-34
SCR field, 17-34
PWMR register, 17-34
PWMS register
SAMPLE field, 20-7
PWMS register, 20-6

R

RCCTL register
EN bit, 21-11
SWR bit, 21-11
XTL field, 21-11
RCCTL register, 21-11
READ_TO register
DATA READ TIME OUT field, 18-22
READ_TO register, 18-22
Refresh mode control register, *see* RMCR register,
RES_FIFO register
RESPONSE CONTENT field, 18-31
RES_FIFO register, 18-31
RES_TO register
RESPONSE TIME OUT field, 18-21
RES_TO register, 18-21
RESERREG*x* register
START bit, 16-16
Reset module
functional description, 6-1
$\overline{\text{RESET}}$ signal, 6-3
Reset signal, *see* $\overline{\text{RESET}}$ signal
Reset source register, *see* RSR register
$\overline{\text{RESET_DRAM}}$ signal, 6-3
RESETREG1 register, 16-15
RESETREG2 register, 16-15
REV_NO register
REVISION NUMBER field, 18-25
REV_NO register, 18-25
RMCR register
LCDC_EN bit, 17-35
SELF_REF bit, 17-35
RMCR register, 17-35
RSR register, 6-3
RSSR*x* register
RSS field, 13-25
RSSR*x* register, 13-25
RST bit, 15-7
RTC
alarm, 21-3

minute stopwatch, 21-3
operation, 21-2
prescaler and counter, 21-2
programming model, 21-4
sampling timer, 21-3
RTC control register, *see* RCCTL register
RTC day alarm register, *see* DAYALARM register
RTC days counter register, *see* DAYR register
RTC hours and minutes alarm register, *see* ALRM_HM register
RTC hours and minutes counter register, *see* HOURMIN register
RTC interrupt enable register, *see* RTCIENR register
RTC interrupt status register, *see* RTCISR register
RTC seconds alarm register, *see* ALRM_SEC register
RTC seconds counter register, *see* SECONDS register
RTCIENR register
1HZ bit, 21-15
2HZ bit, 21-15
ALM bit, 21-15
DAY bit, 21-15
HR bit, 21-15
MIN bit, 21-15
SAM0 bit, 21-15
SAM1 bit, 21-15
SAM2 bit, 21-14
SAM3 bit, 21-14
SAM4 bit, 21-14
SAM5 bit, 21-14
SAM6 bit, 21-14
SAM7 bit, 21-14
SW bit, 21-15
RTCIENR register, 21-14
RTCISR register
1HZ bit, 21-13
2HZ bit, 21-13
ALM bit, 21-13
DAY bit, 21-13
HR bit, 21-13
MIN bit, 21-13
SAM0 bit, 21-13
SAM1 bit, 21-13
SAM2 bit, 21-13
SAM3 bit, 21-13
SAM4 bit, 21-12
SAM5 bit, 21-12
SAM6 bit, 21-12
SAM7 bit, 21-12
SW bit, 21-13
RTCISR register, 21-12
RTOR*x* register
CLK bit, 13-28
CNT field, 13-28
EN bit, 13-27
PSC bit, 13-28
RTOR*x* register, 13-27
RT*x* BRM incremental register, *see* UBIR_*x* register
RXDATAREG1 register, 16-6
RXDATAREG2 register, 16-6
RXDATAREG*x* register
DATA field, 16-6

S

SAR*x* register
SA field, 13-19
SAR*x* register, 13-19
Screen start address register, *see* SSA register
SCSR register, 27-15
SD I/O
description, 18-54
interrupts, 18-54
IRQ readwait feature, 18-8
IRQ service handling, 18-8
readwait, 18-55
suspend/resume, 18-55
SDCTL0 register, 22-9
SDCTL1 register, 22-9
SDCTL*x* register
CLKST field, 22-12
COL field, 22-10
DSIZ field, 22-11
IAM bit, 22-11
ROW field, 22-10
SCL field, 22-12
SDE bit, 22-9
SMODE field, 22-9
SP bit, 22-10
SRC field, 22-12
SRCD field, 22-12
SREFR field, 22-11
SRP bit, 22-12
SDHC
card clock control, 18-12
SDRAM
memory refresh, 22-67
SDRAM 0 control register, *see* SDCTL0 register
SDRAM 1 control register, *see* SDCTL1 register
SDRAM control registers, 22-9
SDRAM memory controller, *see* SDRAMC
SDRAM operation, 22-37
SDRAM reset register, *see* SDRST register
SDRAM/SyncFlash command encoding, 22-18
SDRAMC
address multiplexing, 22-29
auto-refresh mode, 22-25
bank addresses, 22-31
block diagram, 22-2

clock suspend low power mode, 22-35
column address strobe, 22-6
command controller, 22-3
configuration registers, 22-3
configuring controller for memory array, 22-38
data aligner/multiplexer, 22-3
data bus (internal), 22-5
data qualifier mask, 22-6
features, 22-1
functional overview, 22-3
general operation, 22-28
memory configuration examples, 22-39
miscellaneous register, *see* miscellaneous register
mode register programming example, 22-62
multiplexed address bus, 22-5
non-multiplexed address bus, 22-6
normal read/write mode, 22-19
operating modes, 22-18
page and bank address comparators, 22-3
pin configuration, 22-7
powerdown operation in reset and low-power
modes, 22-33
powerdown timer, 22-4
powerdown, 22-35
precharge command mode, 22-24
programming example
mode register
bit assignments, 22-64
converting to an address, 22-63
example 1, 22-62
example 2, 22-64
programming model 22-8
refresh request counter, 22-3
refreshing SDRAM, 22-32
reset initialization, 22-57
reset/powerdown, 22-6
row address strobe, 22-6
row/column address multiplexer, 22-3
SDCLK SDRAM clock, 22-5
SDRAM chip select, 22-5
SDRAM clock enable, 22-5
SDRAM selection, 22-37
self-refresh during low power mode, 22-33
self-refresh, 22-33
set mode register mode, 22-25
syncflash
booting from, 22-68
clock suspend timer, 22-72
configuration, 22-68
deep powerdown operation, 22-72
mode register programming, 22-68
operation, 22-67
powerdown operation, 22-72
programming, 22-71
reset initialization, 22-67
syncflash load command mode, 22-26
syncflash programming, 22-27
write enable, 22-6
SDRST register
RST field, 22-16
SDRST register, 22-16
SECONDS register
SECONDS field, 21-7
SECONDS register, 21-7
SEMAEN bit, 15-26
SFCSR register, 27-31
Sharp configuration 1 register, *see* LSCR1 register
SIDR register
SID field, 8-2
SIDR register, 8-2
Silicon ID register, *see* SIDR register
SIZE register
XMAX field, 17-21
YMAX field, 17-21
SIZE register, 17-21
Size register, *see* SIZE register
SKIP_ADDR field, 15-33
Software reset, *see* SWR bit
SOR register, 27-34
SPCTL0 register
MFD field, 12-13
MFI field, 12-13
MFN field, 12-13
PD field, 12-12
SPCTL0 register, 12-12
SPCTL1 register
BRMO bit, 12-14
LF bit, 12-14
SPCTL1 register, 12-13
SPI
block diagram, 16-1
control registers, 16-8
DMA control registers, 16-14
Interrupt control/status registers, 16-10
operation, 16-2
phase/polarity configurations, 16-2
pin configuration SPI1 and SPI2, 16-3
programming model 16-5
receive (RX) data registers, 16-6
sample period control registers, 16-13
signals, 16-3
soft reset registers, 16-15
test registers, 16-12
transmit (TX) data registers, 16-7
SPI 1 control register, *see* CONTROLREG1 register
SPI 1 DMA control register, *see* DMAREG1 register
SPI 1 interrupt control/status register, *see* INTREG1 register
SPI 1 Rx data register, *see* RXDATAREG1

SPI 1 sample period control register, *see* PERIODREG1 register
SPI 1 soft reset register, *see* RESETREG1 register
SPI 1 test register, *see* TESTREG1 register
SPI 1 Tx data register, *see* TXDATAREG1 register
SPI 2 control register, *see* CONTROLREG2 register
SPI 2 DMA control register, *see* DMAREG2 register
SPI 2 interrupt control/status register, *see* INTREG2 register
SPI 2 Rx data register, *see* RXDATAREG2 register
SPI 2 sample period control register, *see* PERIODREG2 register
SPI 2 soft reset register, *see* RESETREG2 register
SPI 2 test register, *see* TESTREG2 register
SPI 2 Tx data register, *see* TXDATAREG2 register
SRCR register, 27-24
SSA register
    SSA field, 17-20
SSA register, 17-20
SSI
    architecture, 27-2
    calculating bit clock from input clock, 27-28
    clock and frame sync generation, 27-4
    clocking, 27-4
    data and control pins
        SSI_RXCLK pin, 27-35
        SSI_RXDAT pin, 27-35
        SSI_RXFS pin, 27-36
        SSI_TXCLK pin, 27-35
        SSI_TXDAT pin, 27-35
        SSI_TXFS pin, 27-36
    data and control pins, 27-35
    data bit shifting configuration, 27-10
    external frame and clock operation, 27-44
    I^2S mode selection, 27-19
    introduction, 27-1
    operating modes
        gated clock mode, 27-43
        network mode, 27-41
        normal mode, 27-39
    operating modes, 27-38
    pin config software example, 27-7
    pin configuration, 27-5
    programming model, 27-7
    receive data register, 27-12
    receive FIFO register, 27-12
    receive shift register, 27-13
    reset and initialization procedure, 27-44
    transmit FIFO register, 27-9
    transmit shift register, 27-9
SSI clocking
    master / synchronous mode, 27-4
    normal input mode, 27-4
SSI control/status register, *see* SCSR register
SSI FIFO control/status register, *see* SFCSR register
SSI option register, *see* SOR register
SSI receive configuration register, *see* SRCR register
SSI time slot register, *see* STSR register
SSI transmit configuration register, *see* STCR register
SSR_A register, 29-17
SSR_B register, 29-17
SSR_C register, 29-17
SSR_D register, 29-17
SSR_*x* register
    SSR field, 29-17
Statistic data generation, 28-13
STATUS register
    APPL_BUFF_FE bit, 18-16
    APPL_BUFF_FF bit, 18-16
    CARD_BUS_CLK_RUN bit, 18-16
    CARD_PRESENCE bit, 18-15
    CRC_READ_ERR bit, 18-17
    CRC_WRITE_ERR bit, 18-17
    DATA_TRANS_DONE bit, 18-16
    END_CMD_RESP bit, 18-16
    RESP_CRC_ERR bit, 18-16
    SDIO_INT_ACTIVE bit, 18-15
    TIME_OUT_READ bit, 18-17
    TIME_OUT_RESP bit, 18-17
    WR_CRC_ERROR_CODE field, 18-16
    WRITE_OP_DONE bit, 18-16
STATUS register, 18-15
STCR register, 27-21
stem PLL control register 0, *see* SPCTL0 register
STOP_CLK bit, 18-15
Stopwatch Minutes Register *see* CNT register
STPWCH register
    CNT field, 21-16
STPWCH register, 21-15
STR_STP_CLK register
    ENDIAN bit, 18-14
    MMCSD_ENABLE bit, 18-14
    MMCSD_RESET field, 18-14
    START_CLK bit, 18-14
STR_STP_CLK register, 18-14
STSR register, 27-30
SWR bit, 23-4
SWR_A register, 29-23
SWR_B register, 29-23
SWR_C register, 29-23
SWR_D register, 29-23
SWR_*x* register
    SWR bit, 29-23
SWRST bit, 15-26
System control
    programming, 8-1
    system boot mode selection, 8-7
System PLL control register 1, *see* SPCTL1 register

T

TCMP1 register, 23-6
TCMP2 register, 23-6
TCN1 register, 23-8
TCN2 register, 23-8
TCR1 register, 23-7
TCR2 register, 23-7
TCTL1 register, 23-4
TCTL2 register, 23-4
TEN bit, 23-5
Test reset pin signal, *see* $\overline{\text{TRST}}$ signal
TESTREG1 register, 16-12
TESTREG2 register, 16-12
TESTREG*x* register
    LBC bit, 16-12
    RXCNT field, 16-12
    SSTATUS field, 16-12
    TXCNT field, 16-13
time-out status registers, *see* TSR_*n*,
Timer 1 control register, *see* TCTL1 register
Timer 2 control register, *see* TCTL2 register
Timer capture register, *see* TCRx register
Timer compare register, *see* TCMPx register
Timer control registers, *see* TCTL*x* register
Timer counter register, *see* TCNx register
Timer enable, *see* TEN bit
Timer prescaler registers, *see* TPRERx register
Timer status register, *see* TSTATx register
TPRER1 register, 23-5
TPRER2 register, 23-5
$\overline{\text{TRST}}$ signal, 6-3
TSR_*n* register
    ADDR field, 7-15
    BE1 bit, 7-15
    BE2 bit, 7-15
    BE3 bit, 7-15
    BE4 bit, 7-15
    MODULE_EN field, 7-15
    RW bit, 7-15
    TO bit, 7-15
TSR_*n* register, 7-14
TSTAT1 register, 23-9
TSTAT2 register, 23-9
TUSR1_*x* register
    RDY bit, 24-40
TXDATAREG*x* register
    DATA field, 16-7

U

UART
    baud rate
        automatic detection logic, 24-17
    binary rate multiplier (BRM)
        programming examples, 24-15
        updating, 24-15
    binary rate multiplier (BRM), 24-15
    definitions
        clear to send, 24-7
        data carrier detect, 24-7
        data set ready, 24-7
        data terminal ready, 24-6
        request to send, 24-5
        ring indicator, 24-7
        UART receive, 24-8
        UART transmit, 24-8
    DMA requests, 24-4
    escape sequence detection, 24-19
    features, 24-1
    general definitions, 24-4
    infrared interface
        description, 24-20
        receiving, 24-20
        serial infrared mode (SIR), 24-20
        transmitting, 24-20
    interrupts, 24-4
    introduction, 24-1
    module interface signals, 24-2
    operation
        in low-power system states, 24-52
    pin configuration, 24-3
    programming model, 24-21
    receiver
        description, 24-12
        idle line detect, 24-13
        receiving a BREAK condition, 24-14
        vote logic
            operation, 24-15
            special case operation, 24-15
        vote logic, 24-14
        wake, 24-14
    RTS edge
        triggered interrupt, 24-6
    sub-block description, 24-9
    transmitter
        description, 24-10
        FIFO empty interrupt suppression, 24-11
UART 1, *see* UART
UART 2, *see* UART
UART 3, *see* UART
UART FIFO Control registers, 24-38
UART *x* escape timer register, *see* UTIM_*x* register
UART1 control register 3, *see* UCR3_1 register
UART2 control register 3, *see* UCR3_2 register
UART*x* baud rate count register, *see* UBRC_*x* register
UART*x* BRM incremental preset registers 1-4, *see* BIPR*x*_*x* register

UART*x* BRM modulator preset registers 1-4, *see* BMPR*xx* register
UART*x* BRM modulator register, *see* UBMR_*x* register
UART*x* Control register 1, *see* UCR1_*x* register
UART*x* Control register 2, *see* UCR2_*x* register
UART*x* Control register 4, *see* UCR4_*x* register
UART*x* escape character register, *see* UESC_*x* register
UART*x* FIFO control register, *see* UFCR_*x* register
UART*x* Receiver register n, *see* URXDn_*x* register
UART*x* status register 1, *see* USR1_*x* register
UART*x* Status register 2, *see* USR2_*x* register
UART*x* Test register 1, *see* UTS_*x* register
UART*x* transmitter register n, *see* UTXnD_*x* register
UBIR_*x* register
    INC field, 24-46
UBIR_*x* register, 24-46
UBMR_*x* register
    MOD field, 24-47
UBMR_*x* register, 24-47
UBRC_*x* register
    BCNT field, 24-48
UBRC_*x* register, 24-48
UCR1_*x* register
    ADBR bit, 24-26
    ADEN bit, 24-26
    DOZE bit, 24-28
    ICD field, 24-27
    IDEN bit, 24-26
    IREN bit, 24-27
    RDMAEN bit, 24-27
    RRDYEN bit, 24-27
    RTSDEN bit, 24-27
    SNDBRK bit, 24-27
    TDMAEN bit, 24-27
    TRDYEN bit, 24-26
    TXMPTYEN bit, 24-27
    UARTCLKEN bit, 24-27
    UARTEN bit, 24-28
UCR1_*x* register, 24-26
UCR2_*x* register
    CTS bit, 24-30
    CTSC bit, 24-29
    ESCEN bit, 24-30
    ESCI bit, 24-29
    IRTS bit, 24-29
    PREN bit, 24-30
    PROE bit, 24-30
    RTEC field, 24-30
    RTSEN bit, 24-30
    RXEN bit, 24-31
    SRST bit, 24-31
    STPB bit, 24-30
    TXEN bit, 24-30
    WS bit, 24-30
UCR2_*x* register, 24-29
UCR3_1 register
    AIRINTEN bit, 24-32
    BPEN bit, 24-33
    FRAERREN bit, 24-32
    INVT bit, 24-33
    PARERREN bit, 24-32
    REF25 bit, 24-33
    REF30 bit, 24-33
    RXDSEN bit, 24-32
UCR3_1 register, 24-32
UCR3_2 register
    AIRINTEN bit, 24-35
    AWAKEN bit, 24-35
    BPEN bit, 24-35
    DCD bit, 24-34
    DPEC field, 24-34
    DSR bit, 24-34
    DTREN bit, 24-34
    FRAERREN bit, 24-34
    INVT bit, 24-35
    PARERREN bit, 24-34
    REF25 bit, 24-35
    REF30 bit, 24-35
    RI bit, 24-34
    RXDSEN bit, 24-35
UCR3_2 register, 24-34
UCR4_*x* register
    BKEN bit, 24-37
    CTSTL field, 24-36
    DREN bit, 24-37
    ENIRI bit, 24-36
    INVR bit, 24-36
    IRSC bit, 24-37
    OREN bit, 24-37
    REF16 bit, 24-36
    TCEN bit, 24-37
UCR4_*x* register, 24-36
UESC_*x* register
    ESC_CHAR field, 24-44
UESC_*x* register, 24-44
UFCR_*x* register
    RFDIV field, 24-38
    RXTL field, 24-39
    TXTL field, 24-38
UFCR_*x* register, 24-38
universal asynchronous receiver/transmitter, *see* UART
universal serial bus, *see* USB device
URXDn_*x* register
    BRK bit, 24-24
    CHARRDY bit, 24-23
    ERR bit, 24-23
    FRMERR bit, 24-24
    OVRRUN bit, 24-23

PRERR bit, 24-24
RX_DATA field, 24-24
URXDn_*x* register, 24-23
USB control register, *see* USB_CTRL register
USB descriptor RAM address register, *see* USB_DADR register
USB descriptor RAM/endpoint buffer data register, *see* USB_DDAT register
USB device port
aborted device request, 25-38
bulk traffic, 25-42
catastrophic error, 25-39
configuration download, 25-36
control logic, 25-5
control transfers, 25-42
data transfer operations, 25-39
device initialization, 25-35
endpoint FIFO architecture, 25-4
endpoint interrupts, 25-45
exception handling, 25-38
features, 25-1
general interrupts, 25-44
hard reset, 25-48
interrupt services, 25-44
interrupt traffic, 25-43
introduction, 25-1
isochronous operations, 25-43
module components, 25-2
pin configuration, 25-6
programmer's reference, 25-34
programming model, 25-6
reset operation, 25-47
reset signaling, 25-48
signal description, 25-5
software reset, 25-48
synchronization and transaction decode, 25-4
transceiver interface, 25-5
UDC reset, 25-48
unable to complete device request, 25-38
USB packets, 25-39
USB transfers, 25-41
USB enable register, *see* USB_ENAB register
USB Frame Number and Match Register, *see* USB_FRAME register
USB interrupt mask register, *see* USB_MASK register
USB interrupt status register, *see* USB_INTR register
USB Specification and Release Number Register, *see* USB_SPEC register
USB Status Register, *see* USB_STAT register
USB_CTRL register
AFE_ENA bit, 25-12
CMD_ERROR bit, 25-11
CMD_OVER bit, 25-11
RESUME bit, 25-12
UDC_RST bit, 25-12
USB_ENA bit, 25-11
USB_SPD bit, 25-11
USB_CTRL register, 25-11
USB_DADR register
BSY bit, 25-13
CFG bit, 25-13
DADR field, 25-13
USB_DADR register, 25-13
USB_DDAT register
DDAT field, 25-14
USB_DDAT register, 25-14
USB_EP *x* _STAT register
DIR bit, 25-19
USB_EP0*x*_MASK register
DEVREQ bit, 25-24
EOF bit, 25-24
EOT bit, 25-24
FIFO_EMPTY bit, 25-23
FIFO_ERROR bit, 25-23
FIFO_FULL bit, 25-23
FIFO_HIGH bit, 25-23
FIFO_LOW bit, 25-23
MDEVREQ bit, 25-24
USB_EP0*x*_MASK register, 25-23
USB_EP1_STAT register, 25-19
USB_EP2_STAT register, 25-19
USB_EP3_STAT register, 25-19
USB_EP4_STAT register, 25-19
USB_EP5_STAT register, 25-19
USB_EP*x* _STAT register
BYTE_COUNT field, 25-19
FLUSH bit, 25-20
FORCE_STALL bit, 25-20
MAX bit, 25-20
SIP bit, 25-19
TYP field, 25-20
ZLPS bit, 25-20
USB_EP*x*_FALRM register
ALRM field, 25-32
USB_EP*x*_FALRM register, 25-32
USB_EP*x*_FDAT register
RXDATA register, 25-25
TXDATA field, 25-25
USB_EP*x*_FDAT register, 25-25
USB_EP*x*_FRDP register
RP field, 25-33
USB_EP*x*_FRDP register, 25-33
USB_EP*x*_FSTAT register
ALARM bit, 25-27
EMPTY bit, 25-27
ERROR bit, 25-27
FR bit, 25-27
FRAME bit, 25-29

FRAME0 bit, 25-26
FRAME1 bit, 25-26
FRAME2 bit, 25-26
FRAME3 bit, 25-27
FULL bit, 25-27
GR field, 25-29
OF bit, 25-27
UF bit, 25-27
WFR bit, 25-28
USB_EP*x*_FSTAT register, 25-26
USB_EP*x*_FWRP register
WP field, 25-34
USB_EP*x*_FWRP register, 25-34
USB_EP*x*_INTR register
DEVREQ bit, 25-22
EOF bit, 25-22
EOT bit, 25-22
FIFO_EMPTY bit, 25-21
FIFO_ERROR bit, 25-21
FIFO_FULL bit, 25-21
FIFO_HIGH bit, 25-21
FIFO_LOW bit, 25-22
MDEVREQ bit, 25-22
USB_EP*x*_INTR register, 25-21
USB_EP*x*_LRFP register
LRFP field, 25-30
USB_EP*x*_LRFP register, 25-30
USB_EP*x*_LWFP register
LWFP field, 25-31
USB_EP*x*_LWFP register, 25-31
USB_FRAME register
FRAME field, 25-8
MATCH field, 25-8
USB_FRAME register, 25-8
USB_INTR register
CFG_CHG bit, 25-16
FRAME_MATCH bit, 25-16
MSOF bit, 25-15
RES bit, 25-16
RESET_START bit, 25-16
RESET_STOP bit, 25-16
SOF bit, 25-15
SUSP bit, 25-16
WAKEUP bit, 25-15
USB_INTR register, 25-15
USB_MASK register
CFG_CHG bit, 25-17
FRAME_MATCH bit, 25-17
MSOF bit, 25-17
RES bit, 25-17
RESET_START bit, 25-17
RESET_STO bit, 25-17
SOF bit, 25-17
SUSP bit, 25-17
WAKEUP bit, 25-17
USB_MASK register, 25-17
USB_MCTL register
ENAB bit, 25-18
RST bit, 25-18
SUSPEND bit, 25-18
USB_SPEC register
SPEC field, 25-9
USB_SPEC register, 25-9
USB_STAT register
ALTSET field, 25-10
CFG field, 25-10
INTF field, 25-10
RST bit, 25-10
SUSP bit, 25-10
USB_STAT register, 25-10
USR1_*x* register
AIRINT bit, 24-41
AWAKE bit, 24-41
ESCF bit, 24-41
FRAMERR bit, 24-41
PARITYERR bit, 24-40
RRDY bit, 24-41
RTSD bit, 24-40
RTSS bit, 24-40
RXDS bit, 24-41
USR1_*x* register, 24-40
USR2_*x* register
ADET bit, 24-42
BRCD bit, 24-43
DTRF bit, 24-42
IDLE bit, 24-42
IRINT bit, 24-43
ORE bit, 24-43
RDR bit, 24-43
RTSF bit, 24-43
TXDC bit, 24-43
TXFE bit, 24-42
WAKE bit, 24-43
USR2_*x* register, 24-42
UTIM_*x* register
TIM field, 24-45
UTIM_*x* register, 24-45
UTS_*x* register
FRCPERR bit, 24-51
LOOP bit, 24-51
RXEMPTY bit, 24-51
RXFULL bit, 24-51
SOFTRST bit, 24-52
TXEMPTY bit, 24-51
TXFULL bit, 24-51
UTS_*x* register, 24-51
UTXnD_*x* register
TX_DATA field, 24-25

UTXnD_*x* register, 24-25

V

VCR register
    V_WAIT_1 field, 17-27
    V_WAIT_2 field, 17-27
    V_WIDTH field, 17-27
VCR register, 17-27
VERSION NUMBER field, 15-27
Vertical configuration register, *see* VCR register
Virtual page width register, *see* VPW register
VPW register
    VPW field, 17-22
VPW register, 17-22

W

$\overline{\text{WAT_RESET}}$ signal, 6-3
Watchdog reset bit, *see* WDR bit
Watchdog timer reset, *see* $\overline{\text{WAT_RESET}}$ signal
WDR bit, 6-4
W-size register A, *see* WSRA register
W-size register B, *see* WSRB register
WSRA register, 13-16
WSRB register, 13-16
WSR*x* register
    WS field, 13-16
WUCR4_*x* register
    KEN bit, 24-36

X

X INDEX INCR bit, 15-8
X INDEX LOAD bit, 15-8
X SIGN ALT bit, 15-8
X SIGN INI bit, 15-8
X_DATA_SEL bit, 15-8
XBASE field, 15-17
X-COUNT field, 15-32
XCOUNT field, 15-21
XINCR field, 15-20
XINDEX field, 15-18
XMODIFY field, 15-19
X-OFFSET field, 15-31
X-size register A, *see* XSRA register
X-size register B, *see* XSRB register
XSRA register, 13-17
XSRB register, 13-17
XSR*x* register
    XS field, 13-17

Y

Y INDEX INCR bit, 15-8
Y INDEX LOAD bit, 15-8
Y SIGN ALT bit, 15-9
Y SIGN INI bit, 15-9
Y SIGNED bit, 15-9
Y_DATA_SEL bit, 15-8
YBASE field, 15-22
Y-COUNT field, 15-32
YCOUNT field, 15-25
YINCR field, 15-25
YINDEX field, 15-23
YMODIFY field, 15-24
Y-OFFSET field, 15-32
Y-size register A, *see* YSRA register
Y-size register B, *see* YSRB register
YSRA register, 13-18
YSRB register
    YS field, 13-18
YSRB register, 13-18